AN INTEGRATED COMPUTATIONAL APPROACH FOR AI-ASSISTED BIM ANALYTICS AND DIGITAL TWIN DEVELOPMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

NİLSU ŞAHİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF DATA INFORMATICS

JANUARY 2026

Approval of the thesis:

AN INTEGRATED COMPUTATIONAL APPROACH FOR AI-ASSISTED BIM ANALYTICS AND DIGITAL TWIN DEVELOPMENT

Submitted by NİLSU ŞAHİN in partial fulfilment of the requirements for the degree of **Master of Science in Data Informatics Department, Middle East Technical University** by,

Date: *14.01.2026*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Nilsu Şahin

Signature : _____

ABSTRACT

AN INTEGRATED COMPUTATIONAL APPROACH FOR AI-ASSISTED BIM ANALYTICS AND DIGITAL TWIN DEVELOPMENT

Şahin, Nilsu

MSc., Department of Data Informatics

Supervisor: Assoc. Prof. Elif Sürer

January 2026, 113 pages

Building Information Modeling (BIM) integrates geometric representations with structured spatial and functional data, enabling reuse in interactive and operational digital twin applications beyond design and construction. Existing BIM models can be incorporated into computational pipelines to support indoor navigation and emergency evacuation in complex facilities. This thesis introduces an integrated framework for AI-assisted BIM analytics and digital twin development. It shows how Revit-based BIM data can be transformed into a unified pipeline for spatial analysis, navigation, and human-centred interaction within a Unity-based environment. The workflow separates geometric representations from structured spatial data to enable real-time indoor pathfinding with the A* algorithm, while accounting for key physical constraints relevant to evacuation. A large language model (LLM) serves as an interaction layer, translating computed routes into natural-language navigation instructions and supporting context-aware user queries. The system provides real-time visualization and interaction via a Unity-based digital twin, enabling interactive testing and demonstrations. The framework supports both algorithmic and human-centred evaluation. Thirty participants completed user evaluations using the System Usability Scale (SUS) and the Usefulness, Satisfaction, and Ease of Use (USE) Questionnaire. Overall, the proposed approach offers a scalable solution for integrating BIM-based navigation, AI-driven interaction, and immersive digital twins for indoor and emergency-oriented applications.

Keywords: Building Information Modeling (BIM), Digital Twin, Large Language Model (LLM), AI-assisted Spatial Analysis, Indoor Navigation and Emergency Evacuation

ÖZ

YAPAY ZEKÂ DESTEKLI BIM ANALITIĞI VE DIJITAL İKIZ GELIŞTIRMEYE YÖNELIK ENTEGRE HESAPLAMALI YAKLAŞIM

Şahin, Nilsu
Yüksek Lisans, Veri Bilişimi Bölümü
Tez Yöneticisi: Doç. Dr. Elif Sürer

Ocak 2026, 113 sayfa

Yapı Bilgi Modellemesi (BIM), geometrik gösterimleri yapılandırılmış mekânsal ve işlevsel verilerle entegre ederek, tasarım ve inşaat süreçlerinin ötesinde etkileşimli ve işletmeye yönelik dijital ikiz uygulamalarında yeniden kullanım olanağı sağlar. Mevcut BIM modelleri, karmaşık tesislerde iç mekân navigasyonu ve acil durum tahliyesini desteklemek amacıyla hesaplamalı iş akışlarına dâhil edilebilir.Bu tez, yapay zekâ destekli BIM analitiği ve dijital ikiz geliştirmeye yönelik bütünleşik bir yaklaşım sunmaktadır. Revit tabanlı BIM verilerinin, Unity tabanlı bir ortamda mekânsal analiz, navigasyon ve insan merkezli etkileşimi destekleyen birleşik bir iş akışına nasıl dönüştürülebileceğini göstermektedir. İş akışı, geometrik gösterimleri yapılandırılmış mekânsal verilerden ayırarak, tahliye senaryoları için kritik olan temel fiziksel kısıtları dikkate alan A* algoritması ile gerçek zamanlı iç mekân yol bulmayı mümkün kılmaktadır.Büyük dil modeli (LLM), etkileşim katmanı olarak görev yaparak hesaplanan rotaları doğal dilde navigasyon talimatlarına dönüştürmekte ve bağlama duyarlı kullanıcı sorgularını desteklemektedir. Sistem, Unity tabanlı bir dijital ikiz aracılığıyla gerçek zamanlı görselleştirme ve etkileşim sunarak etkileşimli test ve gösterimlere olanak tanımaktadır.Yaklaşım, hem algoritmik hem de insan merkezli değerlendirmeyi desteklemektedir. Otuz katılımcı, Sistem Kullanılabilirlik Ölçeği (SUS) ile Kullanışlılık, Memnuniyet ve Kullanım Kolaylığı (USE) Anketi kullanılarak yapılan kullanıcı değerlendirmelerine katılmıştır. Genel olarak, önerilen yaklaşım, iç mekân ve acil durum odaklı uygulamalar için BIM tabanlı navigasyon, yapay zekâ destekli etkileşim ve sürükleyici dijital ikizlerin entegrasyonuna yönelik ölçeklenebilir bir çözüm sunmaktadır.

Anahtar Sözcükler: Yapı Bilgi Modellemesi (BIM), Dijital İkiz, Büyük Dil Modeli (LLM), Yapay Zekâ Destekli Mekânsal Analiz, İç Mekân Navigasyonu ve Acil Durum Tahliyesi

To My Dearest Family

ACKNOWLEDGMENTS

I would like to begin by expressing my gratitude to my supervisor, Assoc. Prof. Elif Sürer for her unwavering support and wisdom from the very first day of this journey. The opportunity to work with her was one of the main reasons I wanted to pursue a master's degree in this department. Her support throughout my academic process and during my thesis period, her constant encouragement, understanding, cheerful demeanour, and belief that we could overcome anything together always motivated me and brought me to where I am today. Knowing that our work will continue and having had the opportunity to meet her so early makes me feel very fortunate.

The user testing process, which constitutes a significant and important part of this thesis, was crucial to the study's success. I would like to thank all the participants who patiently took part in this process and generously shared their time and ideas. This work would not have been possible without them.

I would also like to thank my dear family for never losing faith in me. Their patience, understanding, constant support, and never letting me feel alone were among the most important pillars of this journey.

I would also like to express my gratitude to my manager, Merve Özdemir Öğüt. Her support for my academic studies and her understanding during my efforts to balance my professional and academic life have greatly contributed to my success. I also thank her for consistently sharing her industry knowledge on BIM processes, which are a significant part of this thesis, and for enabling me to develop professionally in this field.

Finally, I would like to thank all my friends and colleagues who supported me at every stage of the process, motivated me, broadened my vision through the exchange of ideas, and never made me feel alone. I especially want to thank Başak Sungar and Orhun Kılınç for their patience and friendship, for helping me get back on my feet whenever I fell during this challenging period, and for giving me the motivation to try again.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ v
DEDICATION vi
ACKNOWLEDGMENTS vii
TABLE OF CONTENTS viii
LIST OF TABLES xii
LIST OF FIGURES xiii
LIST OF ABBREVIATIONS xv
CHAPTERS
INTRODUCTION 1
1.1 Problem Statement and Motivation 1
1.2 Research Questions and Objectives 2
1.3 Scope and Limitations 3
1.4 Contributions 5
1.5 Thesis Structure 6
LITERATURE REVIEW 7
2.1. Large Language Models in Emergency Management and Navigation 7
2.2. BIM and Digital Twins by Game Engines 7
2.3. BIM Data Conversion to Game Engines 8
2.4. Digital Twins for Emergency Evacuation 9
2.5. Game Engines in Evacuation Simulation 9
2.6. LLM Integration with Spatial Reasoning 10
2.7. Pathfinding and Navigation Algorithms 10
2.8. Human Behaviour Modeling in Evacuations 11
2.9. Real-Time Data Integration in Digital Twins 11
2.10. Evaluation Metrics for Navigation Systems 12

2.11. Voice Interaction in Emergency Systems 12
2.12. Gaps in Current Research 12
SYSTEM ARCHITECTURE AND COMPONENTS 15
3.1. Overall Architecture Design 15
3.2. Data Model and Building Representation 19
3.3. Occupancy Grid Generation Process 24
3.4. Door Leaf Blocking Implementation 26
3.5. Pathfinding Algorithm Details 27
3.5.1. Heuristic Function 27
3.5.2. Neighbour Generation 27
3.5.3. Cost and Evaluation Function 28
3.5.4. Path Reconstruction 28
3.5.5. Performance Optimizations 28
3.5.6. Path Simplification 28
3.6. Baseline Command Generation 31
3.7. LLM Integration Architecture 32
3.8. Command Simulation System 34
3.9. Evaluation Metrics and Statistical Analysis 35
3.9.1. Success Rate 35
3.9.2. Collision Rate 35
3.9.3. Path Ratio 35
3.9.4. Success weighted by Path Length (SPL) 36
3.10. Question-Answering Evaluation Methodology 36
3.10.1. Appropriate Answer Rate 37
3.10.2. Contains Expected Information Rate 37
3.10.3. Rejects Irrelevant Rate 37
3.10.4. Overall QA Performance Assessment 38
3.11. HTTP Server Architecture 38
3.12. Game Engine Client Architecture 39
3.13. Voice Interaction System 40
3.14. Performance Optimization 40

3.15. Unity Integration and Automation of Digital Twin Generation 41
3.16. User Interface and Interaction 42
USER EVALUATION 49
4.1. Ethics 49
4.2. Participants 49
4.3. Procedure 50
4.3.1. Scenario A: Baseline Navigation (Without AI Assistance) 50
4.3.2. Scenario B: AI-Assisted Navigation (With LLM) 51
4.3.3. Time Measurement Protocol 53
4.3.4. Data Collection Methods 53
4.3.5. Time Measurement Comparison 53
4.3.6. Usability Questionnaires 53
4.3.7. Questionnaire Administration 54
RESULTS 57
5.1. LLM Benchmark Results 57
5.1.1. Success Rate 58
5.1.2. Collision Rate 59
5.1.3. Path Ratio 60
5.1.4. SPL (Success weighted by Path Length) 60
5.1.5. Average Commands 61
5.1.6. Average Latency 62
5.2. Question-Answering Results 62
5.2.3. Rejects Irrelevant Rate 64
5.2.4. QA Latency 65
5.3. User Evaluation Results 66
5.3.1. System Usability Scale 66
5.3.2. Usefulness, Satisfaction, and Ease of Use (USE) questionnaire 68
5.3.3. Correlation Analysis Between SUS and USE Scores 73
5.3.4. User Feedback Results 75
5.3.5. Time Measurement Results 76
DISCUSSION 77

6.1 Discussion of LLM Benchmark Results 77
6.1.1. Navigation Performance Analysis 77
6.1.2. Latency Analysis 79
6.1.3. Question-Answering Performance Analysis 79
6.1.4. Model Comparison and Selection Considerations 80
6.2 Discussion of User Evaluation Results 81
6.2.1. System Usability Scale (SUS) Results Discussion 81
6.2.2. USE Scale Results Discussion 82
6.2.3. User Feedback Analysis Results 83
6.2.4. Correlation Analysis Between SUS and USE Scores Discussion 84
6.2.5. Time Measurement Results Discussion 85
6.3 Practical Implications 85
CONCLUSION 87
7.1 Conclusion 87
7.2 Limitations and Future Work 88
DISCLAIMER 91
REFERENCES 93
APPENDICES 101
APPENDIX A 101
APPENDIX B 103
APPENDIX C 107
APPENDIX D 111
APPENDIX E 113

LIST OF TABLES

Table 1: Descriptive Statistics of SUS Scores 66
Table 2: Category Distribution of the SUS Scores 66
Table 3: USE Questionnaire Dimensions.....69

LIST OF FIGURES

Figure 1: Overall System Architecture Diagram. 1
Figure 2: 3D view of the Revit model of building. 19
Figure 3: FBX export of the building in 3Ds Max Interface. 20
Figure 4: Revit plan view of the building showing the locked door. 23
Figure 5: Occupancy grid representation of the building, highlighting the location of the locked door. 24
Figure 6: User Interface of the Digital Twin via Unity Game Engine. 44
Figure 7: Door interaction UI elements. 45
Figure 8: Voice interaction UI elements. 46
Figure 9: Recorded user question located at the top left corner. 46
Figure 10: LLM response located at the top left corner. 47
Figure 11: Loading screen of the simulation. 48
Figure 12: Exit succeed screen. 48
Figure 13: Revit plan view of the building Scenario A. 51
Figure 14: Revit plan view of the building Scenario B. 52
Figure 15: Command Simulation Across Three Evacuation Scenarios. 58
Figure 16: Success rate results of the baseline model and four LLMs across 500 trials. 59
Figure 17: Collision rate results of the baseline model and four LLMs across 500 trials. 59
Figure 18: Path ratio results of the baseline model and four LLMs across 500 trials. 60
Figure 19: SPL results of baseline model and four LLM models across 500 trials. 61
Figure 20: Average number of commands results of baseline model and four LLM models across 500 trials. 61
Figure 21: Average latency results of four LLMs. 62
Figure 22: Appropriate answer rate results of four LLMs. 63
Figure 23: Contains info rate results of four LLMs. 64
Figure 24: Rejects irrelevant rate results of four LLMs. 65
Figure 25: Average latency results of four LLMs. 65
Figure 26: SUS score box plot. 67
Figure 27: Pie chart of the SUS score interpretation. 68
Figure 28: Usefulness score box plot. 70
Figure 29: Ease of Use score box plot. 71
Figure 30: Ease of Learning score box plot. 71
Figure 31: Satisfaction score box plot. 72
Figure 32: Overall USE score box plot. 72
Figure 33: Correlation between SUS scores and Usefulness scores. 73

Figure 34: Correlation between SUS scores and Ease of Use scores.....74
Figure 35: Correlation between SUS scores and Ease of Learning scores. 74
Figure 36: Correlation between SUS scores and Satisfaction scores.....75
Figure 37: Frequency Mapping of most used words about system's strengths. 75
Figure 38: Frequency Mapping of most used words about system's weaknesses. 76
Figure 39: Frequency Mapping of most used words about users' suggestions for improvement.....76

LIST OF ABBREVIATIONS

3D	Three-dimensional
AI	Artificial Intelligence
API	Application Programming Interface
BIM	Building Information Modeling
CORS	Cross-Origin Resource Sharing
CSV	Comma Separated Value
DT	Digital Twin
FBX	Filmbox
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LLM	Large Language Model
RAD	Rapid Application Development
REST	Representational State Transfer
STT	Speech to Text
TTS	Text to Speech
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivation

Building Information Modeling (BIM) has become an important and widely adopted methodology for representing an asset's spatial and semantic information in the construction industry, not only during the construction phase but also during the facility management period. However, extracting valuable information from BIM data for decision-making, especially transforming it into real-time analysis and an interactive digital twin environment, presents significant challenges. Traditional BIM procedures focus on design and documentation. However, BIM data has significant potential for dynamic digital twins, simulations, and artificial intelligence-guided decision-making.

Transforming BIM data to digital twin models is technically complicated. Since BIM data is produced for design and documentation processes, it needs adaptation for real-time visualization, spatial analysis, and user interaction. During this transformation, geometric relationships should be preserved, losing critical semantic relationships should be avoided, and enabling this data to be analysed is important. Additionally, integrating AI to BIM-based digital twins needs to be understood correctly, buildings's geometry and spatial setup. Thus, this process requires a careful modelling approach.

Large Language Models (LLMs) can understand natural language and contextual inferences as well as limited spatial reasoning. Thanks to these features, LLMs can be used as a smart interface for BIM analytics. They can enable users to ask building-related questions naturally, improving understanding and navigation. On the other hand, using LLMs in BIM analytics brings difficulties as well. Rather than natural language tasks, BIM analytics has additional requirements such as precise geometric computations, consideration of physical constraints, as well as consistent and predictable results that can be applied safely in real-world situations.

The integration of AI into BIM data and digital twin environments is limited both in the literature and in application. For example, although pathfinding algorithms such as A* can optimize the complex indoor paths. However, presenting these outputs in logical, user-friendly commands and with smart analytics requires advanced natural language processing and generation capabilities. In these procedures, the asset's geometry, spatial relationships, and physical constraints such as doors and obstacles

should be considered. Additionally, especially in safety-critical scenarios, reliability and accuracy must be the main concerns.

Current studies mainly focus on BIM data management, digital twin development, and AI applications individually. Studies that consider the integration of these aspects for advanced analysis and the generation of decision-making systems are quite limited.

This thesis proposes an integrated computational framework for AI-assisted BIM analytics and digital twin development. This framework processes BIM data into digital twin environments. This approach not only adopts advanced geometric modelling and spatial reasoning but also considers some constraints to represent the scenarios more accurately. The framework integrates LLMs into digital twins to enable natural language interaction and AI-assisted guidance. The practical value of the study is shown with a case study based on a theoretical evacuation scenario. Then, this system is also evaluated through comprehensive user testing. The contribution of this thesis is clearly reported.

1.2 Research Questions and Objectives

This thesis focuses on several fundamental research questions while proposing a computational framework for AI-assisted BIM analytics and digital twin development. Firstly, it examines how BIM data can be processed and analysed to be usable within digital twin environments. The primary goal is to use AI's spatial reasoning and pathfinding capacities to make these environments more powerful. Another research question addresses how LLMs can work with BIM-based digital twins. The aim is to improve natural language interaction and spatial analysis capabilities that allow users to ask questions about the building and been navigated with AI assistance. Furthermore, the thesis investigates the technical processes that enable the production of digital twin representations of buildings by examining computational methods and algorithms that transforming BIM data into game engine environments. Technical challenges related with data transformation, geometric representation problems, and real-time visualization requirements are discussed in detail. Furthermore, the study focuses on how AI-assistance can make digital twins more functional, especially in situations such as evacuation scenarios. In this context, the aim is to provide empowered navigation support by the integration of LLM features and wayfinding algorithms. Finally, the thesis considers evaluation metrics and frameworks for measuring both technical performance and user experience. The aim is to develop a comprehensive evaluation approach that considers both system usability and technical capabilities.

By the consideration of these goals and observations, this thesis aims to address the following questions:

RQ1: What are the optimal ways to process BIM data and develop digital twin environments through game engines? What are the technical requirements and challenges of these processes?

RQ2: How can LLMs be used with BIM-based digital twins? How can smart analysis, spatial reasoning, and natural language interaction be provided?

RQ3: Which computational approaches and algorithms can be chosen to transform BIM data into digital twins?

RQ4: How can AI-assisted analytics improve the functionality of digital twins for evacuation scenarios?

RQ5: Which evaluation frameworks and metrics can be selected to evaluate the technical performance of AI-assisted BIM analytics and user experience with the digital twin interface?

RQ6: How can different LLMs and their integration methodologies be compared based on their usage in BIM-based digital twin environments?

This thesis aims to develop a computational framework that processes BIM data into interactive digital twin environments. This framework enables AI-assisted analyses within the digital twin environments for spatial reasoning and natural language interaction. It also adopts methodologies to evaluate both technical performance and user experiences. It also investigates how AI can be integrated into BIM analytics and digital twin development systematically. Additionally, demonstrates insights into computational approaches for the transformation of BIM data into digital twins. Finally, it offers an emergency evacuation case study that demonstrates how the proposed approach is implemented in practice. This case study highlights the AI-assisted navigation and analytics features of the study.

1.3 Scope and Limitations

Although this research aims to create an integrated computational method for AI-assisted BIM analytics and digital twin development, it has several limitations. The study focuses on interior building environments. The research encompasses two-dimensional spatial representation. Three-dimensional features are supported by the system but are not explicitly examined.

The study uses static building plans. Dynamic changes and real-time sensor data are not considered. Additionally, the research considers individual user interaction only. Multi-user coordination and crowd dynamics are not addressed. Finally, user interfaces are available only in English. Specific types of office buildings were used in the research.

Benchmark tests used only a single building plan. This may limit how the system can be applied to other building types. Different types of assets such as high-rise buildings or complex structures might have different challenges. Additional considerations may be required, especially for AI-assisted routing. The current system only operates in a two-dimensional plane. It does not address multi-story environments where stairs, elevators, or inter-floor access are important. The system could be extended to three-

dimensional routing in the future. However, this study only examines two-dimensional cases because in most multi-story buildings, each floor typically has one access to evacuation routes, such as evacuation stairs. Testing the single-storey system can be considered as a starting point. This approach aims to validate the framework before including additional complexity. Additional components can be included in the future, such as vertical movement or multiple exits. Static building plans assume that the building structure will not change. Walls, blocks, and obstacles remain in their original positions in the BIM data and do not change during the operation. Thus, the system cannot handle changing conditions of the building structure.

Individual user interaction means that the system allows only one user interaction within the digital twin environment. The system does not consider situations where users interact with each other or use the system all together. Therefore, some important elements are overlooked. For example, simultaneous access by multiple users to the system is not considered. Joint analysis processes are not evaluated. In other words, the ways in which users influence each other while using the same digital twin are also not examined.

Since all prompts, instructions, and LLM responses are in English, the system is suitable for English-speaking users. This can create problems when support in other languages is needed. Since the aim of digital twin applications is to reach diverse user groups, adding multilingual support should be considered in the future.

This research does not cover outdoor environments; it focuses only on indoor evacuation. Large-scale scenarios such as multiple buildings or extensive open spaces, like cities, have also not been examined. Dynamic environments, for example, moving obstacles or changing conditions, have not been considered either. Real-time sensor updates and building management system integration are also not included in this study. Multi-user coordination and crowd dynamics have not been included because such scenarios require different evaluation methods rather than individual user tests. Real-time sensors and building management system integration can make digital twins more dynamic and realistic. However, these features are not part of this study. Accessibility is considered, but in terms of basic physical constraints. Special accessibility options have not been examined in detail. Furthermore, advanced psychological factors influencing user interaction with digital twins have not been modelled.

This thesis acknowledges these limitations and discusses them as areas for further development in future research. The scope of the study has been intentionally limited. This allows for a more detailed focus on the selected area. Furthermore, this approach provides a foundation for progressing application areas of AI-powered BIM analytics and digital twin development.

1.4 Contributions

This thesis offers several contributions to the field of AI-assisted BIM analytics and digital twin development. First, it proposes a computational framework that transforms BIM data into interactive digital twin environments. This framework uses architectural data from standard data formats used in the industry. It then processes both geometric and semantic information. The resulting data is converted into formats suitable for real-time visualization and spatial reasoning. [1]

The second contribution is AI-assisted analytics that combine LLMs with digital twin environments. This approach allows users to query building information using natural language. It also provides spatial navigation support. Thus, it is shown that AI can make digital twins more useful through intelligent interfaces.

The third contribution is that this thesis includes a comprehensive evaluation mechanism both for technical and user aspects. For the technical side, this framework enables statistically reliable evaluations using multiple trials. Multiple evaluation metrics are used to measure performance. These include success rate, collision rate, path efficiency, and combined performance scores. Additionally, user evaluation methodology is conducted, and the results are examined by user evaluation scales. This thesis also investigates the relationship between technical performance and user satisfaction.

The fourth contribution is, this study uses a building model representation which is quite close to real world constraints, such as human radius. System precisely detects whether any collision occurred during the evacuation. Additionally, this study proposes a new approach that considers door leaves as obstacles as well, since people cannot pass through them during the evacuation. The opening directions and widths are modelled accurately as well. This approach the reliability of the evacuation routes as well.

The fifth contribution is the production of real-time integration architecture. This architecture uses HTTP-based communication between game engine clients and analytics backends. This approach enables interactive simulation and visualization. This structure provides real-time path updates within the game environment.

The sixth contribution is the voice interaction feature. This feature allows users to communicate with LLM based chatbot within the digital twin environment. This system is designed by adopting speech-to-text technology. Thus, users can ask questions about building information verbally while moving around the game environment. This feature enables hands-free interaction.

The seventh contribution is the proposed analytical framework that enables researchers to compare different language models under the same conditions. This framework includes a deterministic basis system for fair comparison and avoids vendor lock-in that can be adaptable to any LLM client. Additionally, while new LLMs developed in the industry, those can be included to the system easily.

The eighth and final significant contribution is that this research brings together the existing literature. This study combines different research areas such as BIM analytics, digital twin development, and AI-assisted spatial reasoning and evaluation methods in single approach. This contribution is demonstrated and supported by findings from the case study on emergency evacuation designed for this thesis.

1.5 Thesis Structure

This thesis consists of seven chapters. The content of each chapter is summarized below:

Chapter 1 describes the scope of the research. It presents basic background information related to the field. It introduces the aim of the thesis, its main contributions, and its general structure.

Chapter 2 includes a literature review. This chapter examines the use of LLMs in emergency management and wayfinding. BIM and digital twins are also discussed. The conversion of BIM data into game engines, digital twins for emergency evacuation, and the use of game engines in evacuation simulation are also examined. In addition, the integration of LLMs with spatial reasoning, pathfinding, and navigation algorithms is evaluated.

Chapter 3 describes the system architecture and its components. It defines the overall system design. The data model and building representation are explained. The process of creating an occupancy grid is also described. Pathfinding algorithms, LLM integration, and command simulation are also covered in this chapter. Evaluation criteria and Unity integration are also explained.

Chapter 4 describes user evaluation methods. Ethical issues are addressed. The experimental process is explained. Data collection methods and usability surveys are described.

Chapter 5 presents the results for each evaluation criterion. Both LLM benchmark results and user evaluation results are included in this chapter.

Chapter 6 discusses the collected data and explains the results.

Chapter 7 concludes the study and offers suggestions for future research in this field.

Additionally, Disclaimer chapter clearly explains the limited AI usage in this thesis. Kindly check the p.91 for further information.

CHAPTER 2

LITERATURE REVIEW

2.1. Large Language Models in Emergency Management and Navigation

Recent studies show that Large Language Models (LLMs) have strong potential in the fields of emergency management and navigation. Lei et al. [1] conducted a literature review examining how LLMs are used in different phases of the disaster management. Their study showed that natural language interfaces have significant applications in emergency situations. Their research shows that LLMs can process large amounts of information. They also indicate that they can provide useful directions during emergencies. These features provide a significant advantage for evacuation guidance systems.

Chen et al. [2] combined LLMs with behavioral theories to gather and predict how people make evacuation decisions during wildfires. Their proposed framework, called "FLARE", utilizes psychological theories related to decision-making under uncertain situations. This study demonstrates that LLMs, along with theoretical information, can reflect the complexity of human evacuation decisions. This shows that LLMs are not only limited by the patterns within the data but can also model the thinking processes behind evacuation decisions.

In a recent study by El Fekih Zguir et al. [3], a framework called "RoadMind" was developed. This framework aims to enable LLMs to better understand and process spatial information using OpenStreetMap data. The study shows that LLMs can process road networks, distances, and directions. This feature makes them useful for disaster response. The developed framework uses LLMs as geospatial AI experts that are capable of understanding geographic space and reasoning about spatial relationships. This capability is particularly important in emergency situations. [1]

2.2. BIM and Digital Twins by Game Engines

BIM has changed significantly in recent years, particularly with its increasing integration with digital twin systems. Deng et al. [4] examined how BIM evolved towards digital twins. They also assessed how this evolution impacted the Architecture, Engineering, Construction, and Facility Management (AEC-FM) sector. The study highlights some key challenges and opportunities during the process of converting BIM data into digital twin models. This is particularly important for evacuation simulation systems that require accurate building details.

Saddiq et al. [5] investigated how BIM and game engines work together to create digital twins for construction. Their research shows that BIM data can be transformed into interactive digital twin environments by using game engine technology. This allows for real-time visualization and simulation. This study is particularly important from a technical point of view because it addresses the problem of converting BIM geometry and data into formats compatible with real-time game engines.

Alexander et al. [6] recently investigated how ideas from video game design can contribute to the development of digital twins of buildings. Their research shows that interactive building models can be developed using game design methods. This approach improves user experience and makes the system easier to use. This is particularly important for evacuation systems, as user interaction with the system and active participation in the process are important in such systems.

Clausen et al. [7] evaluated the potential of game engines to create digital twins of photovoltaic systems, the systems that convert sunlight directly into electricity. Their work shows that game engine technologies are not limited to the traditional game development field. It has been shown that these technologies have large application areas. The research also demonstrates that game engines can handle complex physical systems and support real-time data integration. These features are important capabilities for evacuation simulation systems.

2.3. BIM Data Conversion to Game Engines

Researchers are actively investigating the process of translating BIM data into formats that can be used with game engines. Rundel and De Amicis [8] investigated the visualization of game engines with digital twins and their visualization capabilities. Their work made it possible to process and display complex particle data in real time. The research also includes options for converting data formats. In addition, how to optimize large change models is also being examined.

Crampen et al. [9] developed three-dimensional urban digital twins using Unreal Engine 5, another commonly used game engine in the industry, for traffic monitoring purposes. Their work demonstrates how large-scale urban models can be managed in game engines. Advanced methods such as optimizing the level of detail and processing data in real-time streaming were addressed. These methods can also be used in building evacuation simulations, as efficient visualization of large and complex models is important in such applications.

Aksu and Demirel [10] developed a process for transferring BIM models to game engines. Their work explains how BIM data can be automatically transferred to a game engine. They also demonstrate how to create 3D navigation networks. The A* algorithm is used to analyze indoor accessibility in the study. Different scenarios aretested in public buildings. This method forms an important basis for digital twins and smart city projects.

2.4. Digital Twins for Emergency Evacuation

The usage of digital twins has been shown to be useful in emergency evacuation scenarios. Lin et al. [11] investigated how digital twins of buildings and users within buildings can be used in evacuation scenarios. The study focuses particularly on the advantages provided by real-time data integration. The research shows that digital twins can utilize user behavior models, environmental sensor data, and hazard information. This allows for the generation of more realistic evacuation simulations.

Han et al. [12] present a digital twin-based routing system for pedestrian evacuations in narrow buildings. This system uses cellular automata models, the mathematical and computational models used to simulate complex systems by using simple rules applied on a grid of cells. These models were created to determine evacuation routes by recording real-time pedestrian locations and building layouts. Simulations show that this method increases evacuation capability. It also demonstrates that people in emergency situations can be evacuated more safely with this methodology.

Von Danwitz et al. [13] developed a digital twin framework to quickly predict the spread of pollutants in the atmosphere. This system aims to support decision-making processes in emergencies. The developed framework uses finite element meshes to model buildings. It also includes pedestrian crowd models to decide safe evacuation routes. This method allows for fast and reliable predictions to support evacuation planning and help to protect critical infrastructure in emergencies.

2.5. Game Engines in Evacuation Simulation

Game engines are commonly used in evacuation simulations since they offer powerful visualization and have real-time features. Cantrell et al. [14] developed a physics-based crowd evacuation model using the Unity game engine. This model simulates how people behave and move in emergency situations, thus helping researchers to better understand evacuation processes. The study demonstrates that game engines can provide realistic and interactive environments for testing different evacuation strategies and studying crowd behaviour.

Intini et al. [15] developed a platform called “WUI-NITY” with the Unity 3D game engine to simulate evacuations during wildfires at the wildland–urban interface. This platform creates realistic evacuation scenarios by combining fire spread, human reactions, and traffic movements. Their system has a modular structure similar to the structure proposed in this thesis, which allows for the easy addition of different sub-models. This study demonstrates that game engines are effective tools for modeling and evaluating evacuation performance.

Recent research [16] shows that the use of virtual reality together with game engines for evacuation training is promising. These systems provide the user with an immersive training environment. This allows people to better prepare for evacuation

situations and make more accurate decisions. Combining realistic building models with VR technology provides valuable training experiences for both emergency teams and building users.

2.6. LLM Integration with Spatial Reasoning

Recent advances in spatial reasoning by LLMs have created new possibilities for navigation tools. Spatial-RAG [17] combines spatial databases with LLM-based similarity methods. This approach aims to improve the spatial reasoning capabilities of language models. This method combines spatial retrieval techniques and language model features to solve challenges related to spatial data access and interpretation. As a result, LLMs become more successful in real-world spatial reasoning tasks. They can also provide more effective answers to questions about spatial relationships.

Recent studies have examined how large language models (LLMs) can be integrated with geographic information systems (GIS) [18]. These systems allow users to ask questions about spatial data using natural language. This approach demonstrates that LLMs can fill the gap between human language and geometric information. This feature is particularly important for evacuation guidance systems, as people may want to ask about their location or possible evacuation routes in emergency situations, which is also valid for the proposed approach of this thesis.

Wu et al. [19] proposed a prompting method called "Visualization-of-Thought (VoT)". This method aims to improve the spatial reasoning of LLMs. The VoT approach helps the model visually represent the reasoning steps. Thus, the model is guided to solve the problem by following each step. This method has been tested on tasks such as wayfinding with natural language and visual tiling in two-dimensional grid worlds. The results show that the VoT method significantly improves the spatial reasoning ability of LLMs. It also outperformed other multimodal large language models. [1]

2.7. Pathfinding and Navigation Algorithms

Milin et al. [20] compared the performance of Dijkstra and A* algorithms for sea routes and offshore wind farm logistics. According to the research results, both algorithms are successful in improving route planning and logistics processes. However, the A* algorithm runs faster. It also produces routes with fewer turns, especially in large and complex sea scenarios. These findings indicate that the A* algorithm can be considered as more suitable option for route planning and logistics in sea operations.

Lu [21] investigated heuristic path-finding algorithms used in video games. The study analyzes how these algorithms have evolved and how well they perform. The research covers classical algorithms such as A* as well as newer methods. It also explains how these algorithms are used in real games through case studies. The study emphasizes

the need to maintain the balance between speed and path quality. Furthermore, it offers suggestions on how future game AI navigation systems can be improved. [1]

Recent studies have examined the use of machine learning to improve pathfinding processes [22]. These methods learn to predict the best routes faster using simulation data. Thus, real-time pathfinding becomes possible for many agents in complex environments.

2.8. Human Behaviour Modeling in Evacuations

Kwak et al. [23] investigated helping behavior during emergency evacuations. The study used the volunteer's dilemma game theory. The researchers combined game theory with the social force model. This allowed them to examine how people behave and interact with each other in such situations. They observed different group behavior patterns by varying the cost of helping. [1]

Recent advances in behavioral modeling have significantly improved agent-based evacuation models [24, 25]. These models now combine individual characteristics, decision-making processes, and group dynamics by reflecting real crowd behavior more accurately. When they are also combined with game engines, these models help visualize and analyze complex behavioral patterns. This approach supports designers and policymakers, as it allows for both real-time analysis and design-phase evaluation for emergency evacuations. [1]

2.9. Real-Time Data Integration in Digital Twins

Recent research focuses on the integration of real-time data with digital twin systems. Message Queuing Telemetry Transport (MQTT), which is a lightweight messaging protocol designed for efficient communication between devices, and other IoT protocols provide data flow by transferring sensor data to these environments [26]. This allows building conditions, user locations, and hazard information to be updated in real time. This approach helps in creating dynamic evacuation scenarios. [1]

Cloud-based digital twin platforms now support real-time data integration on a large scale [27]. These platforms can manage data from thousands of sensors simultaneously. Yet they can keep latency low for visualization and simulation. This feature is particularly important for evacuation systems in large buildings. [1]

Researchers have investigated the use of edge computing with digital twins to reduce latency in critical applications [28]. In these systems, data is processed locally before being sent to the cloud. This approach enables faster responses, especially in emergencies where every millisecond is significant.

[1] Kindly check the DISCLAIMER (p. 91)

2.10. Evaluation Metrics for Navigation Systems

Recent studies have improved the methods for evaluating wayfinding and evacuation systems. The Success weighted by Path Length (SPL) metric has now been extended to account for multiple agents and different conditions [29]. These updates help to evaluate evacuation systems in more complex scenarios.

Seong et al. [30] developed a navigation benchmark system called “CostNav”. This system measures costs and considers the collision rate as a key factor. Research results show that high collision rates increase operating costs. Therefore, the collision rate is considered an important metric for navigation systems, both in terms of safety and economic evaluation. [1]

Human factors play an important role in the evaluation of wayfinding systems [31]. These criteria have been considered in terms of cognitive load, stress, and usability. These factors are particularly important for emergency wayfinding systems, since users may experience high levels of stress in such situations.

2.11. Voice Interaction in Emergency Systems

Recent advances in speech recognition and text-to-speech technologies have made voice interaction possible in emergency systems. Modern systems like “Whisper”, which is an open-source speech recognition model developed by Open-AI, demonstrate high accuracy rates even in noisy environments [32]. This is particularly important for emergencies, because alarms sound and crowded environments can make speech recognition difficult. [1]

2.12. Gaps in Current Research

Although progress has been made in related fields, there are still some gaps within current research. In particular, the use of LLMs in conjunction with detailed physical constraint modeling for evacuation routing has not been sufficiently explored. These constraints include elements such as door leaf blocking, human radius, and accurate geometric pathfinding. Most current studies focus either on high-level decision support systems or on low-level pathfinding algorithms. However, much research overlooks a critical step: translating calculated routes into explicit and actionable routing instructions that also consider physical constraints.

Researchers have investigated the conversion of BIM data into game engines for evacuation simulations. However, how LLM-based instruction generation is integrated into these systems has not yet been sufficiently explored. This is an important field of study and presents a new opportunity. With this approach, the geometric accuracy of BIM can be improved by leveraging the natural language capabilities of LLMs.

[1] Kindly check the DISCLAIMER (p. 91)

Real-time voice interaction has been studied in evacuation systems. However, how LLM-based routing command generation can be integrated into these systems has not been sufficiently evaluated. This approach could provide hands-free routing capabilities. It could also help create a system that can answer user questions and provide helpful guidance.

There has been an evaluation of how well LLMs perform within security-critical routing tasks. Most existing benchmark studies focus on general language capabilities. However, these studies frequently fail to adequately address important requirements such as geometric accuracy, security, and real-time performance. [1]

This thesis aims to fill these research gaps. Thus, it develops a framework that combines BIM data processing, game engine visualization, LLM-based instruction generation, and powerful evaluation methods. This framework delivers practical tools and important insights to support the development of emergency response systems.

[1] Kindly check the DISCLAIMER (p. 91)

CHAPTER 3

SYSTEM ARCHITECTURE AND COMPONENTS

3.1. Overall Architecture Design

The proposes system is designed by adopting has a modular architecture approach. In this approach, the system is divided into separate components that can work independently and communicate with each other. This modular structure makes the development and updating easier and enables future expansion of the system. The first advantage of this architecture is ease of maintainability. Maintainability refers that, each component can be modified independently from the others [33]. Therefore, when an error occurs in one component, only that part is repaired, and the risk of other components being affected in this maintenance procedure is reduced. For example, when the pathfinding algorithm is needs to be repaired, only that module is changed; other sections, such as visualization, are not affected. This separation makes the system more reliable and manageable in the long term. Another important advantage is testability. Developers can test each component individually. Mock interfaces, which are simplified versions of other components, are used in this process [34]. This method makes it possible to check the correctness of the components without testing the entire system. For example, the component that generates path commands can be tested with sample waypoint data. Thus, errors can be detected and corrected in the early stages of the development process. Modular architecture also offers significant advantages in terms of scalability and extensibility. Each component can be scaled independently according to the processing load that needed [35]. For example, when the HTTP server needs to handle more requests, only that component can be extended; no changes need to be made to the pathfinding infrastructure. Similarly, new components can be integrated into the system without changing existing ones [36]. This flexibility is especially important for research systems where new models and methods are constantly emerging similar with this case. Both AI and BIM research fields are constantly evolving and changing. Overall, the system is designed by adopting modular architecture approach to be able to keep up with technological advances.

This system consists of three main components that work together. The first component is a Python-based backend system. This section performs basic computational operations such as pathfinding and LLM integration. The backend is developed as a library and supports two different use cases. The first use case is batch benchmarking. In this scenario, the system collects performance data by running a large number of trials offline. The goal is to compare the performance of different LLM models by running a high number of trials in a short time. The second use case is real-time server integration. In this case, the backend system supports an HTTP server, and low latency is targeted for interactive applications. Instant feedback for

users is critical in this scenario. Since these two use cases have different requirements, the same backend code is run with different configurations. The first task of the backend system is to load and validate building data. It checks whether the data types are correct, whether the values are within the allowed ranges, and whether the data references are consistent. For example, if a door includes room data that does not exist, that error is detected early. This irrelevant data can affect all the procedure significantly. Thus, this validation process prevents incorrect data from affecting the pathfinding algorithm. After that, the system creates the occupancy grid structure which is used for pathfinding. This structure is a two-dimensional array which each cell is marked as walkable or blocked [37]. In large buildings, generating it for the first time may take longer, but it is only performed once and can be reused afterwards. Thus, following path calculations for the same floor are much faster if more than one trial is needed. The backend then uses the A* pathfinding algorithm to determine the most suitable evacuation route inside the building [38]. That algorithm is used to find the shortest path between the starting and exit points, by avoiding passing through obstacles. In the A* calculations, 8 neighbours are used to enable direct and diagonal movements, and the Euclidean distance is selected as the heuristic function to calculate the distance. Since this calculation process is completed in milliseconds in most cases or the trials, the system can be considered as suitable for real-time uses. The final routes are then simplified using the path simplification method. By this way, unnecessary waypoints are removed, and line-of-sight control is used to test whether two points can be directly connected. This allows to produce shorter and human applicable navigation commands while maintaining safety at the same time. The backend also generates baseline commands for analysis purposes. These commands used as reference for evaluating the commands generated by the LLM. Model performance is considered more successful when the LLM outputs are closer to the baseline results. In addition, the system uses LLM clients for communication with different providers. By this interface, a new LLM provider can be integrated into the system without changing the existing architecture. After the commands are generated, the system performs validation checks. The command format, turn angles, and distances are investigated as well as logical consistency is checked. Then, the commands are tested in a virtual environment using simulation. This simulation considers the limitations of human movement (such as step size, turn radius, and collision control). This simulation checks whether the generated commands are realistic and applicable. Finally, the backend system performs statistical analysis and reporting. Evaluation metrics such as success rate and collision rate are calculated. The results are reported in CSV files and graphs. This automated process is an efficient method which enables to evaluate models over numerous trials. Some libraries are used in the backend system. Pydantic is used for data validation operations and checking the data structures [39]. NumPy accelerates numerical operations, providing high performance, especially in grid calculations. Shapely offers reliable geometric operations and spatial queries [40]. Matplotlib is used to create graphs and visualizations for analysis and reporting [41].

The HTTP server provides a REST API for communication with the game engine client [42]. The server is developed using FastAPI, which is a Python web framework [43]. FastAPI generates automated API documentation via Swagger UI and ReDoc and automatically checks the correctness of incoming requests by using Pydantic models. Furthermore, with asynchronous request management, the server can process many requests simultaneously and consequently, reduces the latency. The server stores building data, occupancy grid, and LLM clients in a shared memory. This way eliminates the need to reload this data with each request. However, thread safety and resource cleanup must be carefully considered during the procedure. The API design follows the RESTful principles and uses the standard HTTP methods and status codes. For example, GET is used to retrieve data, and POST is used to send data as general. 200 indicates successful operations, and 404 indicates unavailable resources, errors. CORS support also allows the Unity client to access the server from a different origin [44]. The system also includes error handling and graceful degradation mechanisms. For example, if an LLM API call fails, the system does not completely stop. An error message is sent to the user, and other processes continue to run. Additionally, with lazy loading feature, LLM clients are only started when needed so that shortening the server's startup time.

The coordinate system transformation between Unity and Python is also addressed in the system. In Unity, x represents east, y represents altitude, and z represents north, while in Python, x represents east, y represents north, and z represents altitude. Furthermore, Unity uses a left-handed coordinate system, while Python uses a right-handed coordinate system. This transformation ensures that both systems correctly interpret position data. The game engine client is developed using C# scripts within Unity. Unity is a widely used game engine offering powerful visualization tools [45]. The client application supports real-time visualization, player position tracking, and voice interaction features. The client communicates with the HTTP server for route requests or user questions, and the results from the server are displayed on the game interface. One of the system's key features is real-time player position tracking. As the player moves inside the building, the system updates the paths and recalculates the directions depends on the current position. By this way, users receive updated results which is significant for evacuation. Path visualizations also allows users to see the path they need to follow with lines in the game environment. In addition, voice input allows users to as their questions without writing them as long texts while they are using their hands for movement. Users can ask questions verbally, and the results are displayed on the screen. Loading screens and status indicators improve the user experience by showing the system that it is running. Unity is the chosen game engine to create this digital twin environment in this study, because it offers powerful 3D visualization tools and it is commonly used in the game industry [45]. The game engine's physics and graphics systems make movements more realistic and allow a better understanding of evacuation scenarios. With real-time interaction, researchers and users can observe how routes are updated by walking around the building and better evaluate the system's real-world performance.

The overall system architecture is summarized in Figure 1.

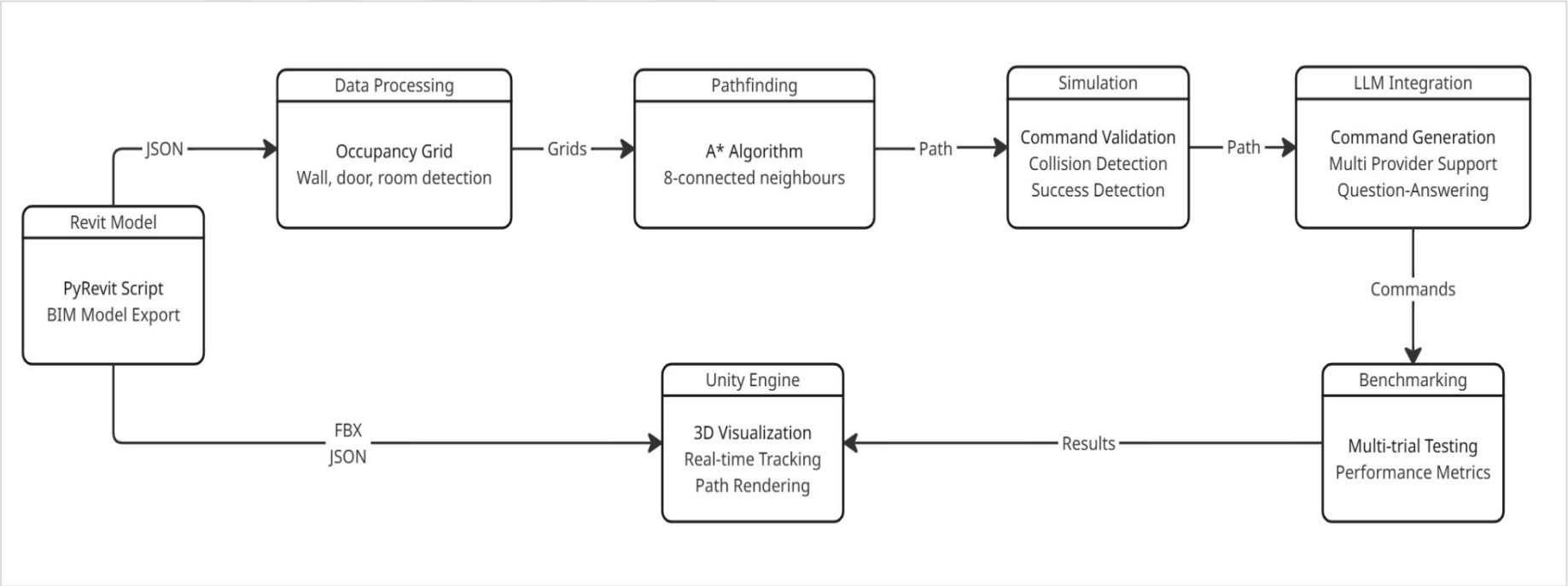


Figure 1: Overall System Architecture Diagram.

3.2. Data Model and Building Representation

The system processes building data exported from Autodesk Revit in JSON and FBX formats. This dual-export approach based on Revit model is intentionally chosen. Revit is a widely used software in the AEC (Architecture, Engineering, and Construction) sector. Therefore, many projects already exist as Revit models. This makes it highly likely that building data is already available in this format for the potential future usage.

For this framework, Autodesk 3ds Max software is also chosen as an intermediary tool. Before the FBX export is loaded into Unity, some necessary changes are made within this environment. Since FBX is a widely utilised 3D data interchange standard in both the gaming and construction industries, it is utilised in this process. There are many modelling, visualisation, and real-time rendering programs that support this file format. This guarantees dependable data transfer between Unity, 3ds Max, and Revit.

3D model view of the building that used in this study is shown in Figure 2. FBX export format is transferred to 3Ds Max for final adjustments. Figure 3 shows the same building in 3ds Max software using the FBX format.

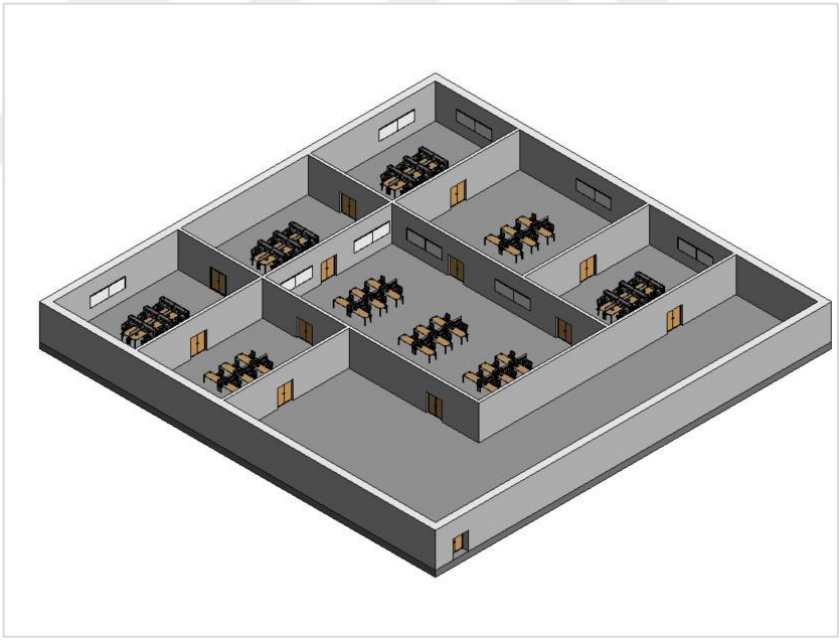


Figure 2: 3D view of the Revit model of building.

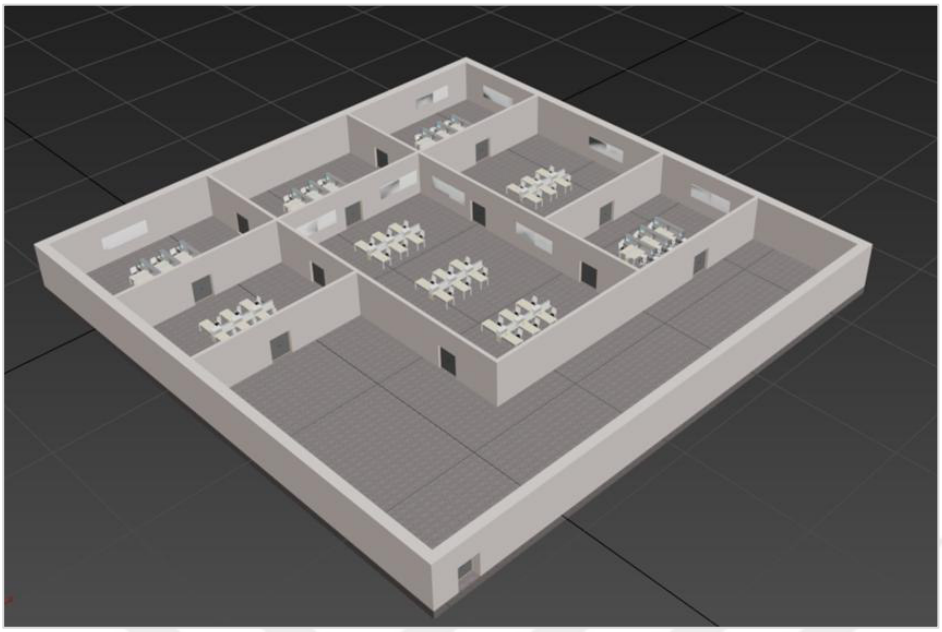


Figure 3: FBX export of the building in 3Ds Max Interface.

JSON facilitates data parsing and validation, as well as debugging. The data model uses Pydantic for validation, which helps keep data types correct and ensures data accuracy throughout the system. JSON is simple to read, making it useful for debugging and understanding data. At the same time, JSON is machine-readable, making it ideal for automated processing. This format strikes a good balance between being readable by both humans and computers.

The export script, which handles data transfer from Revit, uses PyRevit, an open-source tool developed for Autodesk Revit. PyRevit allows users to extend Revit's features and automate repetitive tasks through Python scripts. This enables users to develop custom tools and extract detailed building data from Revit models more efficiently. The script extracts various building components from the model, such as levels, rooms, doors, walls, stairs, and obstacles. To maintain data accuracy, coordinates are rounded to three decimal places, and Revit's internal unit, feet, is converted to meters using a coefficient of 0.3048. Furthermore, the coordinate system used in the exported JSON file is modified to provide compatibility between the Revit model. This ensures that the resulting data accurately represents the building geometry. All parameters used in the export script are defined as shared parameters in the Revit. This shared parameter approach allows the same parameter definitions to be used in different families (Revit model objects) and projects, making the workflow more consistent and sustainable. This parameter set allows users to reuse existing parameters in new projects without having to redefine them. It also ensures that critical information such as obstacle markers, exit doors, room connections, and door widths are always in the correct format. This structure not only increases data consistency but also makes data transfer between Revit and other software more organized.

The main shared parameters used in this system are the *"isObstacle"* definition, used to indicate furniture and equipment that might block movement; the *"isExit"* definition used to indicate main exit door; the *"from_room"* and *"to_room"* fields, which assigned doors with their respective room numbers; and the *"width_m"* parameter, which indicates the door width that will be used for some calculations.

Only elements with the *"isObstacle"* parameter set to "1" are exported as obstacles by the PyRevit script. By this way, only furniture and equipment that block movement are included to the pathfinding calculations.

A room definition refers to each separate space within a building. Each room has a unique identifier (ID). It also contains floor information showing the room's level, labelled as "Level 0". The room's border information is maintained as nested lists of 3D coordinates. This helps to accurately depict the room's real geometric shape. There is also a central point that can be utilised to rapidly locate the room, which is represented by 3D coordinates. A unique room ID enables that rooms to be linked with other model elements, such as doors and neighbour rooms. The room number is designed for easy user understanding and can be customized as needed. Room boundaries are stored as nested lists of 3D coordinates. This structure allows for the representation of more complex geometries and enables the modeling of voids inside the buildings created by obstacles such as columns. A more realistic representation of the building plan is generated by this methodology. Furthermore, the centroid provides practical use in quick check operations however, for precise spatial operations, all boundary information must be used. In the nested list structure of boundary data, the outer list consists of loops, and each loop contains a 3D point list. The first loop stores the boundary of the room, and subsequent loops stores the voids within the room if applicable. This method allows for the modeling of complex room shapes that cannot be defined with simple polygons. Accurate representation of these voids is important because many rooms may contain structural elements or obstacles within the walkable area. For example, in a large open office space, columns can block the walkable area. Thus, accurate modeling of these gaps is critical to obtaining accurate pathfinding results.

The door model connects rooms to each other. Each door contains an ID as well, a type or family name indicating whether it is locked or other characteristics (such as single or double door), source and destination room IDs specifying the rooms it connects to, the level information, 3D coordinates indicating the center point, door width in meters, and a marker indicating whether the door is the main exit. This structure has a critical role in the pathfinding process because it clearly defines the connections between rooms. The source and destination room identifiers inform the system about which rooms are linked and which way the doors open, which is essential for modelling door leaf blocking. The exit flag indicates if the door opens to the outside of the building, which also influences door leaf blocking. In the subsequent section, additional information regarding the orientation of door opening and its impact on pathfinding will be provided.

The Obstacle model identifies physical objects that block the movement. Each obstacle has a unique identifier. It also has a type or family name used for classification. In addition, there is a descriptive name, level information, and a clearance parameter that allows for additional clearance around the object. The model uses a 2D bounding box to represent the size of the obstacle. This box is defined by minimum and maximum coordinate points as a rectangle. The inflation parameter enlarges this bounding box, leaving enough space for people to safely pass around the object. This parameter can be increased or decreased based on the safety requirements. It also provides a more computationally efficient process. In this study, furniture in the office building was considered as obstacles.

The Wall model is used to represent building walls. Each wall has a unique ID. It also includes a wall type text indicating the wall's thickness and sometimes other characteristics if needed. This text is usually written in a format such as "Wall-30 cm". The wall's location is shown by a curve composed of 3D coordinate points. This curve represents the wall's centerline, thus determining its orientation and placement. The system analyzes the wall type text to determine the wall's thickness. This information is important for occupancy grid generation. The cells occupied by the walls are correctly marked through this process. This prevents routes to passing through the walls. This feature is important for indoor wayfinding.

The Pydantic library is used to validate all data models in this system. This library helps ensure data consistency by checking whether the information is in the correct. It checks whether the defined data types are compatible with the actual data and whether the values are within the specified limits. For example, distance values must be positive, coordinates must be valid, and they must be in the defined boundary of the building. In addition, the system examines reference relationships within the data. For example, it checks whether the room IDs to which doors are linked are defined in the system. If any problem is detected during this process, Pydantic generates error messages. This makes it easier to quickly find and correct errors before the procedure starts. The data validation process runs automatically using type hint definitions as soon as the data is loaded into the system. This allows potential errors to be detected early, simplifies the debugging process, and increases the overall reliability of the system.

The system can export the building model in JSON format as well as in FBX format for use in Unity. This allows both data analysis and 3D visualization to be performed on the same model. However, Revit places door pivot points in the center of the door by default rather than door hinges; this is not suitable for realistic door animations in Unity. Therefore, the FBX file is first exported from Revit, then imported into 3Ds Max, the door pivots are moved to the hinge points, and the updated model is exported again and loaded into Unity. By moving the pivots to the hinge position, doors can be opened by rotating like real doors, which is particularly important for realistic evacuation simulations where door leaf movement might be obstructed. The Revit model is used as the single source for all building data, and data consistency is maintained because both JSON and FBX files are generated from this model. For example, room boundaries, door positions, obstacles, and wall geometries completely

match the JSON data and the FBX visualization. Similarly, room numbers, door type and connection information are valid in both formats. This consistency is important in this system that JSON data is used for pathfinding and the FBX model is used for visualization. Ensuring that the generated routing commands accurately match the environment seen in the Unity scene is important. The consistency between the Revit model and the occupancy grid can be seen in the building plans shown in Figures 4 and 5. In this representation, walls are marked as blocked areas along with their thicknesses, door opening directions are calculated by considering their opening directions and hinge positions, and obstacles are located at the same position and dimensions in both models. Furthermore, the algorithm considers locked and unlocked doors differently: no opening is created in the wall for locked doors, and these doors are shown as fully closed wall segments within the occupancy grid. Thus, the pathfinding algorithm cannot pass through locked doors as it would in real life and operates based on an accurate representation of the building geometry.

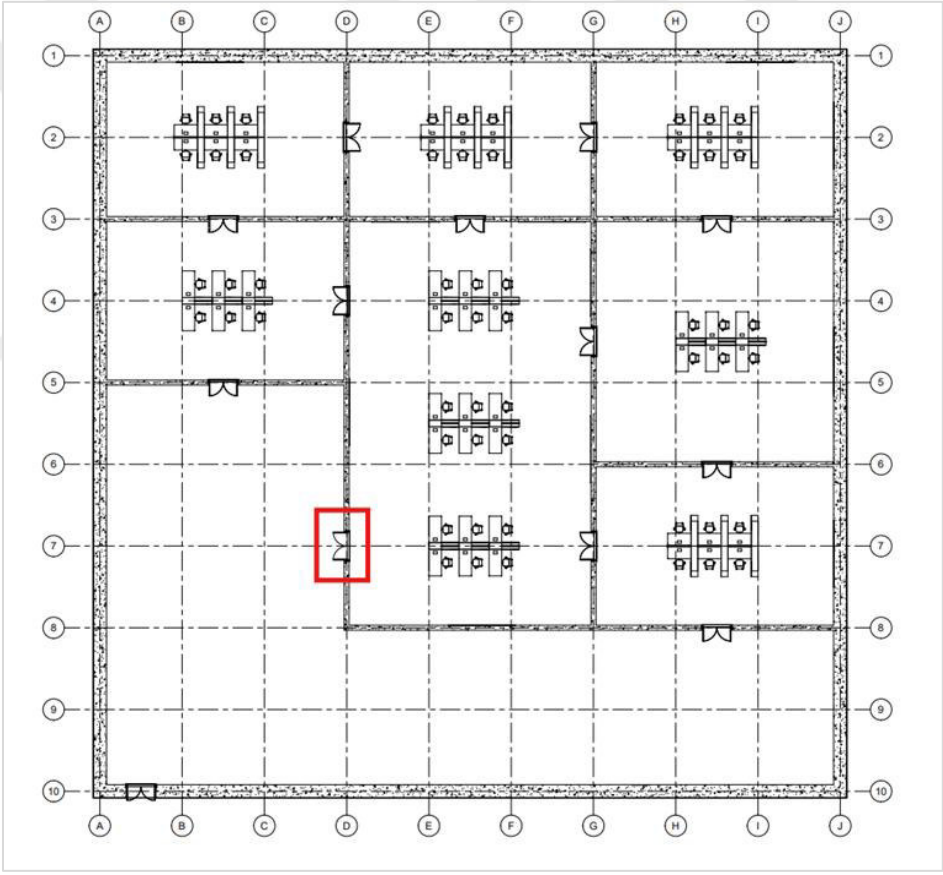


Figure 4: Revit plan view of the building showing the locked door.

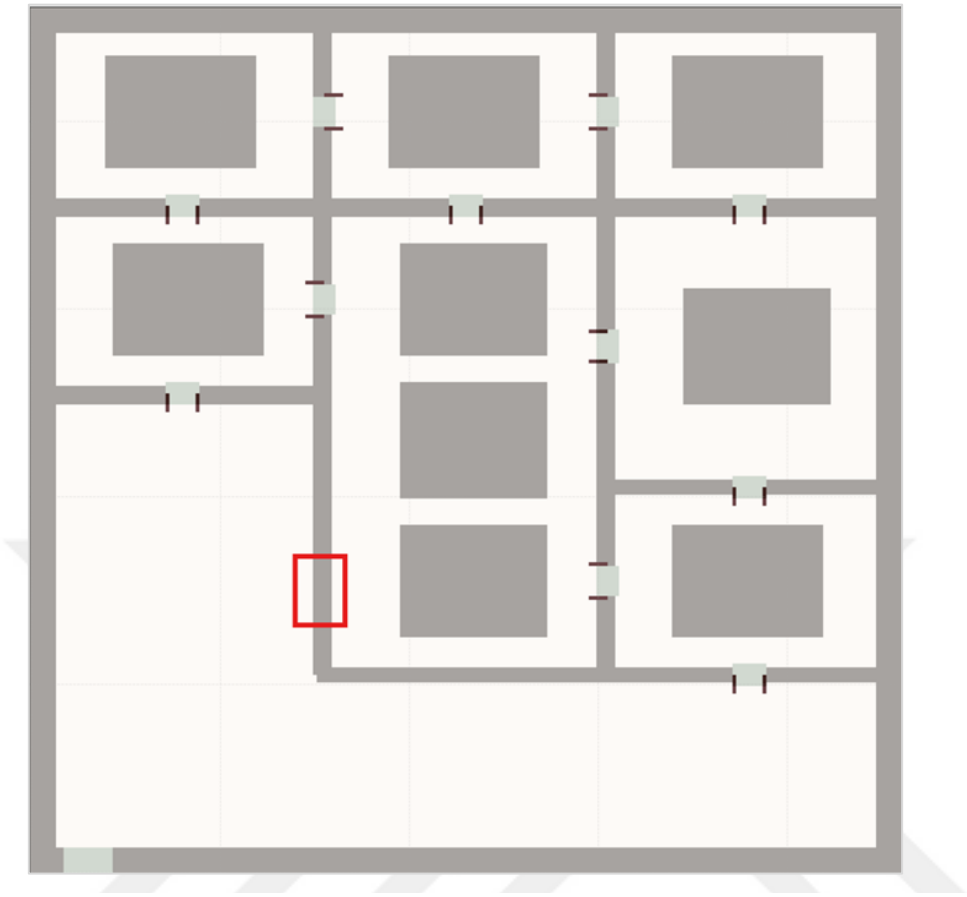


Figure 5: Occupancy grid representation of the building, highlighting the location of the locked door.

3.3. Occupancy Grid Generation Process

Occupancy grids are a simple but effective method used in pathfinding. In this approach, the building space is divided into a grid, and each cell is marked as either walkable (free) or blocked. However, since continuously dividing space into cells can lead to some quantization errors, a balance must be provided between accuracy and memory and computational cost when determining grid resolution. While higher resolution provides more accurate results, it requires more computational resources. The first step in creating the grid is to define the bounding box area of room polygons. The system examines all rooms and converts them to the walls into Shapely polygons and calculates the building's boundaries using minimum and maximum x and y coordinates. A 1-meter padding is typically added to prevent data loss at the edges.

The grid is initially represented as 2D Boolean array created using NumPy. "True" indicates that a cell is blocked, while "False" indicates that it is walkable. The grid size is determined by the bounding box and the selected resolution, and for safety reasons, all cells are initially set as blocked. The system then converts the center point of each cell to world coordinates. Then, Shapely's contains method is used to check if this

point is within a room polygon. Cells within the room are marked as walkable, which defines the basic movement area. While this process can be computationally costly for large grids, it is be performed efficiently with Shapely's optimized methods in this study.

Walls are marked as blocked on this grid to prevent passage through them. The system determines the centerline of each wall from its start and end points, then converts the wall thickness to meters by checking the type information (e.g., "Wall-30 cm"). A rectangle with this thickness is created, and all grid cells intersecting this area are marked as blocked to be on the safe side. Similarly, the safe distance around the obstacle is extended by adding an custom inflation parameter and a human radius. These values can be modified based on the specific needs. First, a bounding box parallel to the axes is calculated for the obstacle, then this box is expanded in all directions, and the cells within this area are marked as blocked. The default human radius is selected as 0.3 meters in this study. This allows for realistic considerations of narrow spaces.

One of the most complex parts of the system is the modeling of the doors. Locked doors are marked as blocked at the beginning, and the pathfinding algorithm does not use them for calculations. For unlocked doors, an opening is created along the door's axis, by considering the door width and passage clearance. For interior doors, the door leaves are also modeled as the rectangular obstacles; this more represents that how open doors create narrow passages rather than large gaps. During this process, the opening direction is determined using the room IDs to which the door is connected, and the position of the door leaves is calculated as well. These leaves are then marked as blocked areas on the grid as well since paths cannot pass through them.

To accurately represent thin geometries, the system also uses a method called "rasterization padding". This method ensures that thin objects, such as door leaves correctly fill the grid cells. Without padding, a thin obstacle might not align with the cell center, and the pathfinding algorithm might pass the edge of the obstacle. Padding prevents such situations, ensuring that obstacles are correctly represented on the grid.

Grid resolution selection is critical for accuracy and performance. Smaller cells (in other words, higher grid resolution) allow for more accurate route calculations and better representation of narrow passages. However, they require more memory and computing power. On the other hand, larger cells may accelerate the system but might reduce the accuracy. In this study, the default resolution is set at 0.2 meters per cell. Since human radius is accepted as 0.3 meters, a person would occupy approximately two to three cells. This provides a good balance between accuracy and computational cost. Users can change this value according to their needs and system resources.

The system is also designed to support multi-storey buildings. Each floor can be processed separately using level names such as "Level 0". Independent occupancy grid is created for each floor if they have any difference. This allows to separate pathfinding operations on each floor. However, more advanced modeling is required for vertical movement (such as stairs or elevators) which is not applied currently. However, this

system architecture can be expanded in the future to support 3D navigation. Although multi-storey building support is not fully investigated in this study, the system is designed to accommodate such expansions.

3.4. Door Leaf Blocking Implementation

Accurate modeling of door geometry is important for realistic pathfinding because doors are mostly the only and narrowest passageways inside the buildings. If the algorithm assumes the entire width of the door is open, by ignoring its leaves, the calculated route may not be applicable in reality. Since most doors opens as 90 degree, the leave might block the passage. Additionally, the door leaves' thicknesses can reduce the passage area. In double-leaf doors, both leaves occupy space when open, leaving a walkable gap in the middle; however, this gap is often smaller than the total width of the door.

The door leaf blocking algorithm uses geometric calculations to determine the relationship of the door between walls. First, the orientation of the door is determined by finding the wall segment closest to the center of the door. The door axis is determined perpendicular to this reference line, thus assigning the door to the belonging wall. Accurate calculation of the door axis is important because the position of the door leaves is determined relative to this axis; if the axis is incorrect, obstacles will be placed incorrectly, and the pathfinding results will not be correct. The door opening direction is determined using the source and destination room identities to which the door is connected. In most cases, since doors open towards the room people wish to enter, the system positions the door leaves towards the destination room. The position of the door leaves is calculated by using the door width. For double doors, the leaves are considered as symmetrical, leaving a passage gap in the middle of the door frame. Each door leaf is represented by a rectangle with exact dimensions. The length of the rectangle represents the door opening distance, and the width represents the door thickness, and it is positioned to the door axis.

These rectangles are then converted into grid cells and marked as blocked areas. The system transforms the corners of the rectangles into grid coordinates and blocks all cells that intersect with this area. Additionally, rasterization padding is applied to ensure accurate representation of thin geometries as mentioned earlier. Some door types need to be handled with special rules. Emergency exit doors generally open to the outside, so their leaves do not block passage. Therefore, the system does not apply this calculation for emergency doors to reduce calculation efforts. Locked doors are also not included to calculations by default. Additionally, for single-leaf doors, only one door leaf is modelled different that the double doors.

Several verification methods are used to confirm this door leaf blocking implementation. The first method is visual verification. By examining the grid images generated by the system, it is checked whether the obstacles created by the door leaves are correctly assigned to the cells. This visual check is a process that only needs to be performed once before the calculations begin, as the geometric representation of the building does not change throughout the process. At this stage, the generated model is

compared with Revit architectural drawings to verify whether the door positions and geometry are consistent with the architectural design. The second method is the path analysis. In this analysis, the generated paths are examined to verify that the routes do not pass through areas blocked by door leaves. Finally, user feedback is also used as an important evaluation method. Feedback obtained from users helps identify potential problems in this block modeling approach.

3.5. Pathfinding Algorithm Details

The A* algorithm's implementation is used with some optimizations in this study. The algorithm kept in some data structures. A heap is used to implement the open set as a priority queue, with the $f(n)$ values in order. This makes it possible to insert and extract the least amount of data in $O(\log\ n)$ time. The closed set is used an array-based storage system that enables check the items in $O(1)$ time. There are two separate arrays are stored for the g-scores and f-scores so that they can be accessed quickly, and there are other separate arrays for the parent pointers so that paths can be quickly reconstructed [46]. Node representation uses integer grid coordinates (x, y) for the efficiency. These coordinates are converted to world coordinates only when they are needed for the output. This integer-based representation prevents floating-point precision errors and improves lookup efficiency. [1]

3.5.1. Heuristic Function

The heuristic function uses Euclidean distance, which is a basic and applicable formula for 2D pathfinding. This heuristic equation can be is defined as:

$$h(n) = \sqrt{(g_x - n_x)^2 + (g_y - n_y)^2} \times r \quad (1)$$

where g_x and g_y are the destination coordinates, n_x and n_y are the current coordinates, and r is the selected grid resolution [46].

Since the straight-line distance is the shortest possible path, this heuristic calculation never overestimates the true cost. It also satisfies the triangle inequality, by ensuring the consistency and efficiency which reduces the number of node expansions. [1]

3.5.2. Neighbour Generation

Neighbour generation uses an 8-connected neighbourhood for comprehensive pathfinding. The system generates four cardinal neighbours with coordinates:

$$(x \pm 1, y), (x, y \pm 1) \quad (2)$$

with a movement cost of 1.0, representing north, south, east, and west movements.

Additionally, four diagonal neighbours are generated with coordinates:

$$(x \pm 1, y \pm 1) \tag{3}$$

with a movement cost of:

$$\sqrt{2} \approx 1.414 \tag{4}$$

representing north-east, north-west, south-east, and south-west movements. The diagonal cost stands for the longer distance travelled and it ensures that the algorithm finds optimal paths correctly. To prevent corner cutting, diagonal movements are not allowed when either of the adjacent cardinal cells is blocked. This ensures that the generated paths do not pass through the obstacle corners.

3.5.3. Cost and Evaluation Function

The A* algorithm function combines the actual cost from the start with the heuristic estimate:

$$f(n) = g(n) + h(n) \tag{5}$$

The actual path cost is calculated as:

$$g(n) = g(\text{parent}) + c \times r \tag{6}$$

where c is the movement cost multiplier (1.0 for cardinal moves and $\sqrt{2}$ for diagonal moves), and r is the grid resolution. This formulation ensures that diagonal movements are correctly penalized [46].

3.5.4. Path Reconstruction

The path is reconstructed from the target node to the starting node by following the parent node markers. Then, the list is reversed to get the path in the correct order, from start to the end. The duration of this reconstruction depends on how long the path is and makes a list of waypoints that can be used for commands or visualizations.

3.5.5. Performance Optimizations

A few optimizations are used to increase the algorithm's performance. Early termination ends the when the exit is reached, so it does not check extra nodes. Grid bounds checking ignores neighbors outside the grid, which prevents errors. Blocked cell checking skips over areas that cannot be crossed. These steps make pathfinding efficient, even for larger grids.

3.5.6. Path Simplification

Path simplification aims to simplify the existing route while maintaining the safety. This process uses the greedy line-of-sight algorithm. The algorithm starts from the first

waypoint and attempts to find the furthest waypoint that can be reached in a straight line without passing through any obstacles. If a direct route between two points is possible, the intermediate waypoints are removed. This process is repeated until the route cannot be simplified any further. The super cover traversal method also is used to check for the clear line-of-sight. This method identifies all grid cells through which the line between two points passes. This is used for quick and accurate collision detection. Waypoints are only removed when there is a clear line of sight exists between them. This ensures the simplified route remains safe and never exceeds the original route.

To sum up, Algorithm 1 shows the overall A* pathfinding method with 8-connected neighbours. It shows how the method starts, looks at neighbours, figures out costs, and reconstruct the path. Algorithm 2 explains how to simplify a path by removing waypoints that can be directly connected without passing through obstacles. This keeps the path safe and efficient.

Algorithm 1:
A* Pathfinding with 8-Connected Neighbours

Input: OccupancyGrid grid, start coordinates (s_x, s_y), goal coordinates (g_x, g_y)
Output: Path as list of grid coordinates, or None if no path exists

1. Initialize:
2. g_score[height, width] ← ∞ for all cells
3. came_from_x[height, width] ← -1
4. came_from_y[height, width] ← -1
5. open_heap ← empty priority queue
6. in_open ← empty set
7. g_score[s_y, s_x] ← 0.0
8. f_start ← h(s_x, s_y) where h(n_x, n_y) = $\sqrt{((g_x - n_x)^2 + (g_y - n_y)^2)}$ × grid.resolution
9. push (f_start, s_x, s_y) to open_heap
10. add (s_x, s_y) to in_open
11. While open_heap is not empty do
12. Pop (f_current, c_x, c_y) from open_heap
13. If (c_x, c_y) not in in_open then
14. continue
15. Remove (c_x, c_y) from in_open
16. If (c_x, c_y) == (g_x, g_y) then
17. Reconstruct path by following came_from pointers backward
18. Reverse path and return
19. base_cost ← g_score[c_y, c_x]
20. For each neighbor (n_x, n_y) in 8-connected neighborhood do
21. If (n_x, n_y) not in grid bounds or grid.is_blocked(n_x, n_y) then
22. skip
23. If diagonal move (n_x ≠ c_x and n_y ≠ c_y) then
24. If grid.is_blocked(n_x, c_y) or grid.is_blocked(c_x, n_y) then
25. skip (prevent corner cutting)
26. end if
27. cost_multiplier ← 1.0 for cardinal, $\sqrt{2}$ for diagonal
28. tentative_g ← base_cost + cost_multiplier × grid.resolution
29. If tentative_g < g_score[n_y, n_x] then
30. came_from_x[n_y, n_x] ← c_x
31. came_from_y[n_y, n_x] ← c_y
32. g_score[n_y, n_x] ← tentative_g
33. f_new ← tentative_g + h(n_x, n_y)
34. Push (f_new, n_x, n_y) to open_heap
35. Add (n_x, n_y) to in_open
36. end if
37. end for

38. Return None (no path found)

Algorithm 2:
Line-of-Sight Path Simplification

Input: Occupancy Grid grid, path as list of waypoints
Output: Simplified path with fewer waypoints
1. If path length ⩽ 2 then
2. return path copy
3. simplified ← [path[0]]
4. i ← 0
5. While i < path.length - 1 do
6. j ← path.length - 1
7. While j > i + 1 do
8. If line-of-sight check passes between path[i] and path[j] then
9. Break (found farthest visible point)
10. j ← j - 1
11. Append path[j] to simplified
12. i ← j
13. end while
14. Return simplified

3.6. Baseline Command Generation

The baseline command generation is a guide for turning waypoints into navigation commands. This standard is used for some purposes: to set a standard for performance by comparing LLMs to it, to make sure the evaluation framework works correctly, and to set the format for commands that LLMs should follow.

The baseline algorithm goes through the waypoints one at a time and makes *"turn"* and *"walk"* commands for each part of the route. The heading is set to 0 degrees at the start, which means that the coordinate system is facing east. This starting point was chosen to keep things the same and make it possible to get the same results again.

The algorithm uses the *atan2* function to find the target heading for each segment between two waypoints. This function finds the angle in degrees from the current waypoint to the next. The answer is then changed so that it is between -180 and 180 degrees. The formula is as follows:

$$\theta = \text{atan2}\,(y_{i+1} - y_i,\, x_{i+1} - x_i) \tag{7}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) denote two consecutive waypoints, and θ represents the target heading angle relative to the global coordinate frame.

The *atan2* function works in all four quadrants, so it does not have the problems that simpler arctangent functions can have. The algorithm finds the heading change by

taking the difference between the current heading and the target heading and then bringing the result down to a range of -180 to 180 degrees. This way, it always chooses the shortest rotation and cuts down on extra turning.

The algorithm makes a turn command if the heading change is greater than a certain amount, usually 1 degree. This stops small, useless turns. The amount of turn is rounded to the nearest 15 degrees to match how people usually turn in steps, which is not smoothly. If the rounded change is positive, it tells the system to *"turn left"* with that value. If it is negative, it tells player to *"turn right".* Current heading is updated by adding the rounded change and bringing it down to a range of -180 to 180 degrees. The turns feel more real and navigation is easy and accurate when people use 15-degree steps.

After the turn, the algorithm measures the distance between waypoints using the Euclidean distance formula. It uses the world coordinates of the waypoints to get accurate results. The Euclidean distance formula is:

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{8}$$

where $(x_i, y_i)$and (x_{i+1}, y_{i+1})denote two consecutive waypoints, and $d_i$represents the distance between them.

When the distance is more than the threshold, usually 0.40 meters, a *"walk"* command is raised with the distance rounded to two decimal places. This limit keeps tiny movement commands from being given that are hard to follow and not worth the trouble. 0.40 meters is almost half of a normal step size, so it can be considered as minimum for useful movement commands.

Segment merging is another feature that used to combine sequential path segments in the same direction into a single command. If the directional difference between two segments is less than 20 degrees, these segments can be combined under a single *"walk"* command. This eliminates separate turn commands for small directional changes, reducing the total number of commands. This approach considers the natural directional movements by people to make it more human followable and applicable, especially in emergency situations. Commands are written in a simple and standard format. Turn commands are *"turn left X.XX deg"* or *"turn right X.XX deg"*, and walk commands are "*walk Y.YY m"*; angle and distance values with two decimal places. Each command is written separately with standard units such as degrees and meters. This easy structure makes commands easily understandable by both humans and systems. [1]

3.7. LLM Integration Architecture

LLM integration is designed to allow the use of different providers. By that design, using different LLM services and comparison of their performance is possible. All LLM clients use the same structure and include a method called "generate". This method takes the model's name and prompt text and returns an "LLMResult" object

This result object includes the generated text, model and provider information, latency, and token usage. The OpenAI client uses the official OpenAI Python SDK to connect to the OpenAI API. API authentication is done via environment variables; thus, API keys are not stored in the code, ensuring security which is a common practice working with LLMs. The client also has a retry mechanism that uses exponential backoff for transient errors. This method ensures that the request is retried at specific intervals, especially in case of rate limitations or temporary service interruptions. The system also processes API responses, extracting text content, token usage, and other metadata, and generates error messages in cases such as rate limitations, timeouts, or API errors. On the other hand, the Gemini client connects to Google's Gemini API via the Google Generative AI SDK and its working principle is similar to the OpenAI client. Authentication is done using environment variables as well. The client also standardizes model names and automatically adds the "models/" prefix when necessary, providing the format expected by the API. In addition, the system can extract text and metadata from API responses and includes error handling mechanisms for situations such as quota errors, network issues, or API errors. [1]

The goal of prompt strategies is to guide the model to generate accurate and reliable navigation commands. In waypoint mode, the model is given a complete list of waypoints and instructed to generate commands for each segment according to specific rules. The algorithm, coordinate system, and output format to be used are explicitly specified within the prompt; it is also stated that *"atan2"* should be used for angle calculations and Euclidean distance for distance calculations. This approach helps the model perform geometric calculations correctly and produce more consistent results. In contrast, in segment mode, the model is not expected to perform calculations; pre-calculated turn and traverse segments are provided, and the model is instructed to write them in the correct format. This method makes it easier to distinguish formatting errors from calculation errors. [1]

The primary goal in prompt design is to provide the model with clear and understandable instructions. Using clear rules and a standard output format reduces the likelihood of the model misinterpreting the process and ensures more consistent results. Additional instructions to prevent common errors are also included in the prompt. In all LLM calls, the temperature value is set to 0.0; this makes the output more predictable and ensures the repeatability of experiments. Higher temperature values produce more random results, while a value of 0.0 allows the model to select the most likely word. However, because small variations may occur due to model architecture or implementation differences, experiments need to be repeated multiple times and the results statistically analyzed. When determining prompt length, the aim is to strike a balance between the amount of information and efficiency. Waypoint mode prompts are typically around 500–1000 tokens, segment mode prompts 300–600 tokens, and question-and-answer (QA) prompts 200–400 tokens. While longer prompts can improve accuracy, they can also increase cost and response time. [1] Therefore, the suggested lengths are balanced values determined as a result of experimental tests.

3.8. Command Simulation System

The command simulation system checks the commands that the LLM makes. It can be checked by running the commands on an occupancy grid. During this time, the system checks to see if there has been a crash and if the target has been reached. The simulator checks the agent's status all the time. It keeps track of a number of important pieces of information, such as the agent's position in world coordinates, its direction in degrees, the total distance travelled, the total turn angle, and whether or not it has collided with something. The system keeps track of the agent's position and direction as commands are carried out. This makes sure that the agent's location and direction of movement are tracked correctly. Distance and angle counters measure the agent's total movement. This information is used to see how well the system works. The collision detection checks tells whether the agent collides with any obstacle. This information is very important for checks on safety.

In the simulation, commands are executed sequentially. When a turn command is given, the player's heading is updated according to the that turning angle. The angle increased when turning left and decreased when turning right. After each turn, the heading value is normalized to maintain a range of -180 to 180 degrees. Total turning angle is also calculated to evaluate the complexity of the given commands.

The default step length for walk commands is 0.2 meters. This small step size allows for more precise collision detection. The simulator determines the agent's new position using trigonometric calculations, then converts this position into grid coordinates to check if the relevant cell is blocked or not. If the current cell is defined as blocked, it means the collision is occured and the movement is stopped. If it is empty, the agent's position is updated and the total distance travelled value is increased. Since this collision checking is applied directly on the occupancy grid, this process is quite fast.

The simulator also checks whether the target, the main exit door, is reached after each command. The distance between the current position and the target is calculated by using Euclidean distance. If this distance is less than 0.6 meters, this is considered as the exit is reached and the simulation is successfully ended. This 0.6 meter tolerance is defined since reaching the exact point of the exit is not always possible since the approach is grid based. Additionally, early termination reduces the compute power as well.

At the end of each simulation, the system produces a result object. This object includes the information about whether the simulation is successful, whether any collision occurred, the final position and heading direction, total travel distance, and total turning angle. This information then used to calculate evaluation metrics selected for the evaluation process such as success rate, collision rate, path ratio, and SPL (Success weighted by Path Length.

3.9. Evaluation Metrics and Statistical Analysis

The system uses several metrics to calculate LLM performance. The main metrics are success rate, collision rate, path ratio, and SPL (Success weighted by Path Length). It also calculates average command count and average latency for comparison purposes. These results are calculated for each model separately for each trial.

3.9.1. Success Rate

Success rate calculates the percentage of trials in which the agent reaches the exit without collisions over total trials. It is defined as:

$$\text{Success Rate} = \frac{\text{Number of successful trials}}{\text{Total number of trials}} \times 100\% \tag{9}$$

Success rate is the most fundamental metric, directly indicating whether the system can accomplish its primary task [47]. While a high success rate suggests general capability, this metric does not indicate efficiency or safety itself.

3.9.2. Collision Rate

Collision rate is calculated as the percentage of trials that whether any single collision occurred over all trials:

$$\text{Collision Rate} = \frac{\text{Number of trials with collisions}}{\text{Total number of trials}} \times 100\% \tag{10}$$

This metric is important for safety evaluation, since any single collision may be critical in real evacuation scenarios [30]. Collision rate directly indicated the models' safety performance.

3.9.3. Path Ratio

Path ratio indicates the navigation efficiency by comparing executed path lengths over optimal path length:

$$\text{Path Ratio} = \frac{\text{Executed path length}}{\text{Optimal A* path length}} \tag{11}$$

A ratio close to 1.0 indicates near-optimal navigation, while larger values indicate inefficiency, which means higher evacuation time that is not preferred in emergency situations.

3.9.4. Success weighted by Path Length (SPL)

SPL combines success and efficiency into a single metric:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \times \frac{l_i}{\max(p_i, l_i)} \tag{12}$$

where S_i is a binary success indicator (1 for success, 0 for failure),
l_i is the optimal path length,
p_i is the executed path length, and
N is the total number of trials.

SPL penalizes both failures and inefficient paths, providing a balanced measure of overall performance [29]. However, it should be interpreted carefully, as it aggregates multiple factors into a single value.

3.10. Question-Answering Evaluation Methodology

The QA benchmark tests how well LLMs can answer questions about decisions in an evacuation. There are four kinds of questions: spatial questions about where the agent is right now, distance questions about how far it is to the exit, direction questions about where the exit is, and irrelevant questions that should be ignored. This setup makes it easier to test different skills. Spatial questions check to see if the model can understand and explain where things are; distance questions check to see if it can figure out and share distances; direction questions check to see if it understands how things are related in space; and irrelevant questions check to see if it knows when it does not have enough information to answer.

The system makes context for each test position by finding the current room, using A* pathfinding to figure out how far away the exit is, gathering information about the room, and putting all of this into a context string. The system uses polygon containment tests to figure out which room the test position is in. Then, it uses A* pathfinding to find the real distance to the exit. If that does not work, it uses the straight-line (Euclidean) distance instead. The system gathers information about the room, such as its name, number, and any other relevant details. This context gives the model all the information it needs to give the right answers. The process is automated, so the context stays the same for each trial.

Prompts consist of context, a clear question, and clear instructions on how to respond. The QA design also includes direct instructions and the restricted output format, similar with the navigation command prompts. The instructions enforce the models to generate responses based on the given context. Responses are evaluated by considering these three criteria: relevance, information content, and rejection quality. The system checks whether the LLM responses contain accurate information as well as whether irrelevant questions are rejected. Since this process is also automated, it allows for fast

evaluation over large number of questions. Relevance refers to the answer being accurate and relevant to the question; information content refers to the inclusion of necessary information such as room number or distance; and rejection quality refers to irrelevant questions being rejected with an appropriate message.

3.10.1. Appropriate Answer Rate

The appropriate answer rate measures the percentage of questions that are answered appropriately by the system:

$$\text{Appropriate Answer Rate} = \frac{\text{Number of appropriately answered questions}}{\text{Total number of questions}} \times 100\% \quad (13)$$

This metric corresponds to the "*Accuracy"* metric widely used in QA evaluations [48]. It evaluates the overall correctness of the system across all questions, regardless of their relevance or answerability

3.10.2. Contains Expected Information Rate

The contains expected information rate evaluates performance on relevant questions by measuring the percentage of responses that contain the expected information:

$$\text{Contains Expected Info Rate} = \frac{\text{Number of relevant questions with expected information}}{\text{Total number of relevant questions}} \times 100\% \quad (14)$$

This metric captures how well the system provides complete and informative answers when the question is relevant and should be answered. It is similar with "*Answer Recall"* in QA literature, both metrics measure how effectively a system includes the required information in its responses [49].

3.10.3. Rejects Irrelevant Rate

The rejects irrelevant rate measures the system's ability to correctly identify and reject irrelevant questions over the total number of irrelevant questions asked to the LLM:

$$\text{Rejects Irrelevant Rate} = \frac{\text{Number of correctly rejected irrelevant questions}}{\text{Total number of irrelevant questions}} \times 100\% \quad (15)$$

This metric indicates the performance of the system in understanding questions that should not be answered, which is a very critical aspect for reliability and safety. It can be directly corresponded to "*NIL Recall"* in TREC QA evaluations and "*No-Answer Accuracy"* in benchmarks such as SQuAD 2.0, which these metrics evaluates whether the model correctly avoids from answering unanswerable or irrelevant queries [48,51].

3.10.4. Overall QA Performance Assessment

These metrics are investigated together to represent the system's question-answering performance. They measure if the system answers relevant questions accurately and completely, and if it can understand and reject questions that irrelevant.

3.11. HTTP Server Architecture

The game engine client enables communication between the Unity environment's pathfinding and LLM services using the HTTP server's REST API. The server is built with a modern web framework offering features such as automated API documentation, request validation, and asynchronous processing support. These features simplify the development process and increase system's reliability. Automated API documentation allows developers to easily inspect API endpoints, reducing the need for manual documentation. Request validation helps to detect faulty requests early, which resulting in a more stable system and a better user experience. The server also uses a global state structure to store critical components such as building data, occupancy grids, and LLM clients in the memory. This data is loaded only once and can be used in all requests afterwards, which significantly reduces the processing time. Global state contains the building geometry, occupancy grids, clients and model selection settings. This structure reduces startup time by eliminating the need to restart the system with each request. However, attention to thread safety and proper cleanup of unused resources are important and should be considered while using global state.

API endpoints are very important parts of the game engine client. These endpoints let the Unity environment talk to the system. The building loading endpoint is used to load building data and set up the system. Checking the input data is part of this process. After that, the building's shape is broken down, and the occupancy grid is created. At last, the LLM client is up and running. This endpoint should be called first before calling any other API endpoints. It sets up the system for the other endpoints to work. The path calculation endpoint figures out the best way for the player to get from where they are now to the exit point. During this process, the coordinates are changed from world space to grid space. Then, pathfinding algorithms are run, and a list of waypoints is sent back. As the player moves, this endpoint is called a lot. So, for the system to work well, it needs to work very well.

The question-answering endpoint answers questions about the building and the evacuation plan. It sends the question and the player's current location and building data to LLM, which then sends back the answer. With this feature, players can ask questions and get answers directly in the game.

The server has an audio processing endpoint which supports voice input. The speech-to-text endpoint takes audio data, sends it to a transcription service, and returns the text result. Users can ask their questions by speaking during the evacuation simulation with this feature.

CORS configuration uses middleware to allow the game client to send requests from different sources (cross-origin). In the current settings, requests from any source are allowed along with credentials, and all HTTP methods and headers are supported. While this open configuration is suitable for development and testing processes, it should be configured more restrictively in a production environment for some security reasons.

The server effectively apply error management. When validation errors occur, clients are sent relevant HTTP status codes and descriptive information indicating the areas that should be checked. This allows users to easily correct their input. If a requested resource cannot be found, the server responds with a clear message specifying which information is missing. When the server is operating in development mode, error messages contain more detailed information for developers to facilitate debugging. In production mode, more general error messages are used for security reasons. Additionally, if an operation takes longer, the server sends a timeout status code, indicating that the request has timed out.

3.12. Game Engine Client Architecture

The Unity client has different components, allowing real time user interaction. Developers can customize some settings in the development environment without modifying the overall code. The central control component tracks the player's position and manages the pathfinding process by updating the path whenever required. The player's position within the building is continuously monitored using the game engine's transform system. When the player moves over a certain distance, the system checks if a new route is needed and requests new route information from the server if necessary. Communication between the client and server occurs via asynchronous HTTP. This ensures that network operations do not interrupt the game loop and the system continues to operate smoothly.

The calculated routes are displayed to the user as line renderings on the ground. This visualization component can be customized by developers such as path color, thickness, height, material, and optional waypoint markers. The system also provides additional visual guidance with directional arrows placed along the player. This component calculates the position and direction of the arrows based on waypoint information and automatically updates the arrows when the route changes.

The system also includes a separate service component for the QA feature. This component receives questions from the user interface with the voice recognition system, sends them to the server with the player's current location as well. Then it displays the responses from the LLM. The voice input module controls the microphone and records the user's speech; the recording is sent to the server, converted into text, and the resulting text initiates this QA process. This structure allows users to control the system without using their hands while they can keep moving in the environment rather than writing long question texts.

3.13. Voice Interaction System

The speech-to-text process begins with the user permission for microphone access. Once permission is provided, the system typically selects the default microphone and collects the audio as a 3 to 5 seconds. The game engine's audio processing component uses Unity's microphone recording capabilities to create an audio clip with a specific duration and sampling rate; 44100 Hz is used for high quality recording. The recorded audio is then converted to the appropriate format. This data is sent to the speech-to-text service on the server via an HTTP POST request. The server converts the audio to text by using OpenAI's Whisper, and the resulting text is used as the initial input for the QA process. The QA process begins with the transcribed text. The user's transcribed question text and the player's current location are sent to the LLM. The server collects contextual data such as current room information, distance to the exit, and the current location coordinates. The language model generates a response within the context of the building plan and evacuation scenario; this response is then sent to the client and displayed on the user interface. The voice interaction system also uses an error handling mechanism. If voice recording fails, the recording process is retried. If transcription fails, a clear error message is displayed to the user, offering them the to try again. This approach ensures that users can continue interacting even if a problem occurs in the system. This structure also offers significant advantages in terms of accessibility. Thanks to this hands-free operation, users can keep moving in the game. This feature is particularly useful in emergency situations, such as when people need to carry items, help others, or move physically. Although the system currently only supports English, it can be developed in the future by adding different language supports.

3.14. Performance Optimization

The system has some optimizations across its backend, server, and client components to ensure real-time and stable performance during evacuation simulations. These optimizations aim to accelerate computational processes, reduce network traffic, and improve visualization performance. Backend optimizations focus particularly on making data processing and spatial operations more efficient, thereby reducing overall processing time. Large occupancy grid data is processed quickly using vectorized array operations on the backend. Performing batch operations instead of processing each cell individually significantly reduces computation time. Hash-based data structures are used for spatial queries; this allows for quick verification of whether a point is within a specific room or has collided with an obstacle. Furthermore, lazy loading ensures that LLM clients are only started when needed, reducing system startup time and memory usage. Occupancy grids created for each building floor are cached, allowing the same grid to be reused instead of being recalculated, resulting in improved performance, especially during transitions between floors. Server-side optimizations focus on improving network performance and handling many requests simultaneously. Asynchronous request processing allows the server to handle multiple requests at the same time which reduces the waiting time. Additionally, connection pooling reuses the same HTTP connections in LLM API calls, eliminating the need to

generate a new connection for each request and shortening processing time as well. Furthermore, response compression reduces the data size, especially when sending large waypoint lists allowing clients to receive responses faster. Client-side optimizations focus particularly on rendering performance and memory management. Object pooling is used for waypoint markers; this allows existing objects to be reused instead of creating new ones each time, reducing both memory usage and garbage collection overhead. The coroutine mechanism enables asynchronous processes to run in the background in a single-threaded game engine architecture, preventing interruptions to the user interface. Visualization operations are performed with a single drawing operation using line rendering; this approach reduces the number of draw calls, increasing frame rate. Additionally, unnecessary memory allocations are reduced in frequently running code sections, and memory leaks during long game sessions are prevented by cleaning up temporary files and audio clips. Furthermore, additional methods could be considered to further improve performance in the future. For example, request batching, which combines multiple operations into a single request, level-of-detail systems that reduce the level of detail of remote objects, and stream-based transfer of large datasets instead of loading them completely at the outset, could be implemented. These optimizations could further improve system performance; however, they have not yet been implemented in this study.

3.15. Unity Integration and Automation of Digital Twin Generation

Unity game engine integration includes automated tools that allow building data to be easily imported into the scene. These tools simplify ensures compatibility between JSON data and 3D visualization. The system can automatically generate indoor lighting, colliders and interactive elements in the scene using the exported JSON data. Building data is loaded into Unity by importing a JSON file exported from Revit. This JSON file includes all building geometry, such as rooms, walls, doors, obstacles, and level information. This data enables the automatic creation of some settings at the Unity scene, ensuring that the Unity environment accurately reflects the building structure in the Revit model. The system also generates automatic interior lighting. Point light sources are created for each room using the boundaries and area information in the JSON data. The room area is calculated, and the lights are placed in a grid pattern to provide homogeneous illumination. Developers can change some settings such as brightness, range, hue, and ceiling height. Incorrect wall illumination is prevented by ensuring that the lights are only within the relevant room. Additionally, the height of the lights can be adjusted to for different floor heights. This automatic approach eliminates the need for manual light placement. The system also creates collider components for walls and obstacles using JSON data. Wall segments in the JSON file are processed, and box colliders are assigned. At door locations, openings are created in the wall colliders to allow player passage. However, if the door is locked, the system does not remove the collider, preventing passage. For obstacles such as furniture, colliders are created using the axis-aligned bounding box information in the JSON. The system also allows for origin offset definition, which handles the differences

between JSON and Unity coordinate systems. Thus, the data is correctly aligned with the building model transferred to the Unity scene.

The third-person controller controls player movement and animation states. The system uses a character controller to control movement based on basic physics. The orientation of the camera determines the movement input. The animation system uses an animator controller with state machines to control different movement states. It switches between them based on speed and whether the character is grounded. Ground detection makes sure that animations and movements work correctly on different surfaces. It has adjustable offset and radius settings for precise ground checks.

Door animations enable players to open doors and pass through them. When the player opens a door, the two door leaves move around the hinges. When the player comes closer to the door the system enables door activation UI. When the player turns it on, the door leaves smoothly and turns 90 degrees around their hinges. This animation is made possible by quaternion interpolation, which is used over a set amount of time. The door moves in a natural way thanks to smooth step interpolation. After being opened, the doors stay open for a while before closing on their own. The closing mechanism then smoothly moves the leaves back to where they were before. The system looks for the word “locked” in the family names of the doors that are locked. These doors will not open no matter how hard the player tries. The system knows where each door starts, so it can always close it correctly after it has been opened.

Automated generation tools simplify the setup process and ensure that JSON building data remains consistent with the Unity scene. Generating pathfinding, lighting, collision geometry, and interactive components from a single data source provide compatibility between visual representation and navigation data.

3.16. User Interface and Interaction

The Unity client has the UI that enables user navigation in the evacuation simulation, provides visual feedback, and allows interaction with building elements for players. The UI has some components such as path visualization, directional arrow, door interaction texts, audio input feedback, a game timer, and some information screens. The path visualization shows the calculated evacuation path as a line on the ground. This route is shown as a green line connecting waypoints and is positioned slightly above the ground for better visibility. The route can be displayed as a series of green circles or a continuous line, clearly indicating the exit direction to the player. The color and thickness of the line can be adjusted. This line is automatically updated as the player moves and a new path is calculated. A directional arrow is positioned above the character to guide the player. This arrow always shows the next waypoint or final exit point, indicating the turning direction. The arrow also rotates smoothly when there is a significant direction change is needed. This arrow has light-independent materials to be visible in all lighting conditions. Additionally, the arrow disappears when the player approaches the target at approximately 1.5 meters, providing a clearer view of the target area.

The interface also has a game timer which measures the time passed since the player started movement and successfully reached to the exit. The timer is displayed in bright green, large font in the upper right corner of the screen, and the time is shown in minutes, seconds, and milliseconds in the format “MM:SS.MS”. The timer starts automatically when the player presses any of the movement keys (W, A, S, D, or arrow keys. When the player exits the building, the timer stops, and the total time is displayed on the exit success screen.

Figure 6 shows the complete UI of the gameplay, including the route line, directional arrow, timer, and audio interaction panel.

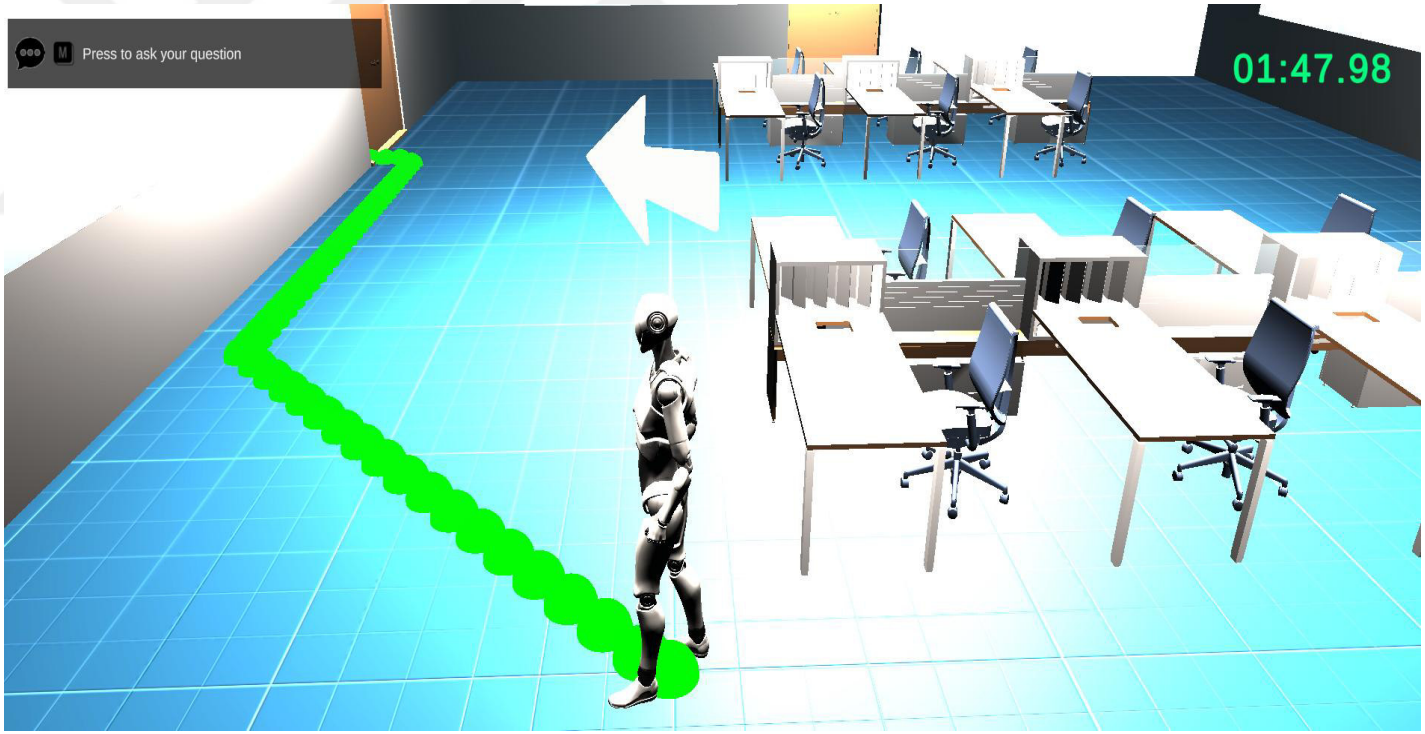


Figure 6: User Interface of the Digital Twin via Unity Game Engine.

When the player approaches a door within the interaction distance, the door interaction interface appears. In this case, a black panel on the screen displays the “O” symbol and the text “Push O to open the door,” indicating that the door can be opened (Figure 7). This warning is only displayed for unlocked doors. The locked doors remain closed. When the player presses the “O” key, the door opens with an animation, and the warning on the interface disappears. The door usually remains open for approximately 5 seconds before automatically closing again; this time can be changed in the system settings.



Figure 7: Door interaction UI elements.

Input controls use standard keyboard and mouse inputs for player movement and interaction. Movement is controlled using the W, A, S, D keys or arrow keys, with W and up arrow moving forward, S and down arrow moving backward, and A/D or left/right arrows moving left and right. Mouse movement controls camera rotation, allowing the player to look around and orient themselves. The M key is used for voice interaction: holding M starts microphone recording and releasing M stops recording and processes the speech. These controls are displayed in the initial information screen to familiarize users with the interface before gameplay begins.

Voice interaction UI provides visual feedback throughout the speech recognition and question-answering process. As shown in Figure 8, the UI panel is located in the upper-left corner of the screen and displays different states with corresponding icons and text. In the idle state, a chat icon is shown with the text “*Press M to ask your question*” and an M key icon.

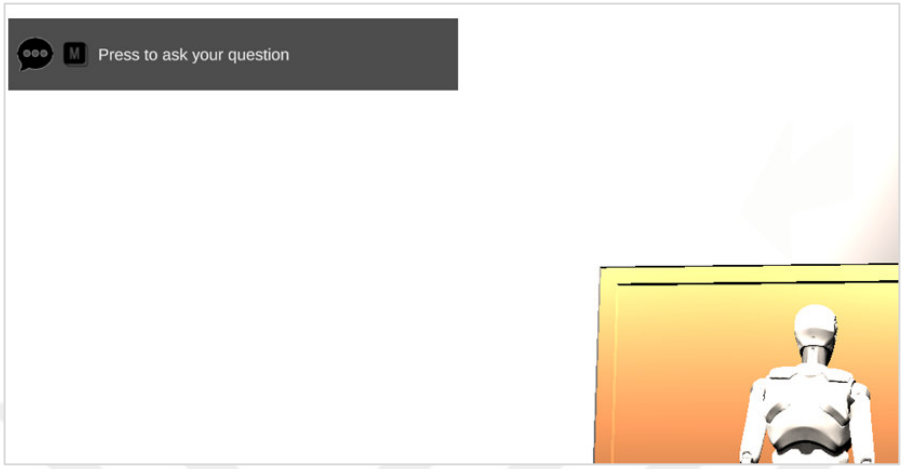


Figure 8: Voice interaction UI elements.

When the M key is held down, the icon changes to a red microphone icon with "*Recording...*" text, indicating active audio capture. When recording stops, the icon changes to a yellow microphone icon with "*Processing...*" text, indicating that the audio is being transcribed.

After transcription, the recorded question is displayed with the microphone icon. When the LLM is generating a response, the icon changes to indicate processing, and the question text remains visible. The user question text, "*Which room I am in*?" can be seen in the top-left corner of Figure 9.

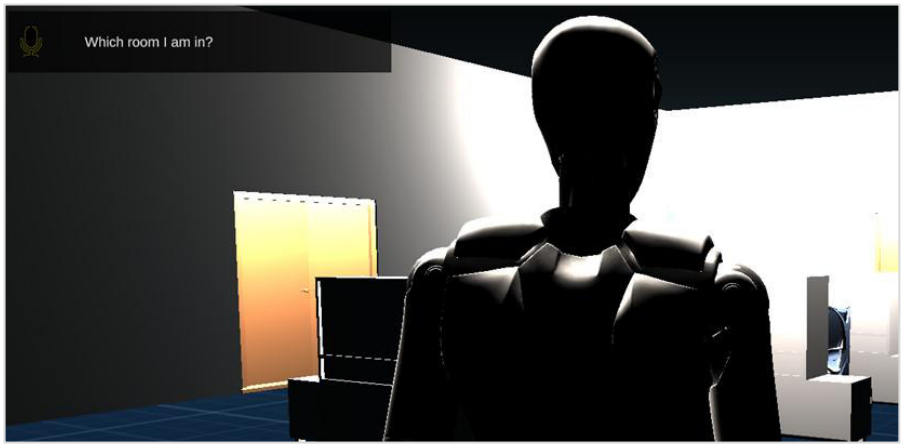


Figure 9: Recorded user question located at the top left corner.

Once the answer is received, the icon changes to a speech or chat icon with a green color, and the answer text is displayed. The answer remains visible for a configurable duration (typically 10 seconds) before returning to the idle state. The UI panel dynamically resizes based on text content, accommodating both short and long responses. The LLM response as "*You are in Room Number 3. Keep mowing towards the exit, you're doing great!*" can be seen the top left corner of Figure 10, as well.

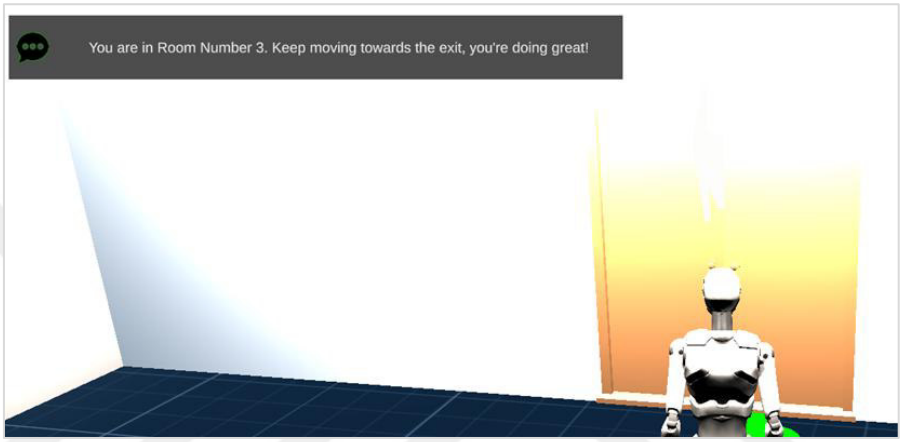


Figure 10: LLM response located at the top left corner.

The initial information screen appears before gameplay begins, displaying instructions and controls while the system connects to the server and calculates the initial path, is shown in Figure 11. The screen shows the game title, control instructions including movement keys (W, A, S, D) and mouse controls, the game objective explaining that the player must find the evacuation exit, information about following the green path, explanation of the arrow indicator for directional guidance, and instructions for using voice interaction by holding the M button. A loading message at the bottom of the screen cycles through status messages such as "*Connecting to server...*", "*Loading building data...*", "*Calculating path...*", and "*Almost ready...*". When the first path is ready this screen disappears and the gameplay starts.

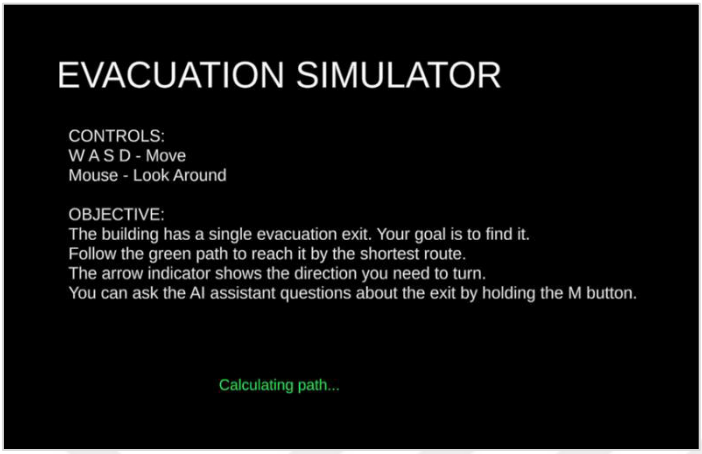


Figure 11: Loading screen of the simulation.

The exit success screen appears when the player exits from the building. This is detected by checking whether the player's position is outside the building boundary. This screen can be seen in Figure 12. The screen has a black layout with a message written as *"Congratulations! You have successfully exited the building"* and the task completion time. The timer stops when the player leaves the building, which shows precise time calculation that will be used for evaluation which is described in later chapters.

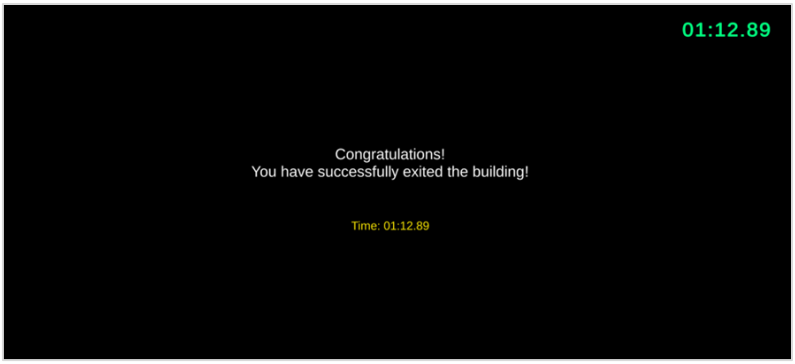


Figure 12: Exit succeed screen.

The UI system aims to give users the information and feedback. Visual elements are designed transparent, so they do not block overall components in the gamescreen such as path. Colour coding makes system more understandable; green shows paths, the timer, and success, red means active recording, yellow is for processing, and white is used for informational text. UI panels of the QA resize automatically depends on the voice output (user query length) and LLM response, to keep information clear and readable.

CHAPTER 4

USER EVALUATION

4.1. Ethics

The Middle East Technical University Human Subjects Ethics Committee approved this research in December 2025. The committee reviewed the documentation, including the study's objectives, methods, and instruments will be used during the tests. All participants were given an informed consent form before the data collection procedure. This form explained the research objectives of the study and how the data would be collected. At the beginning of each session, it was emphasized that participants could leave the study if they felt uncomfortable or stressed. Participation in the study was entirely voluntary. Participants did not receive any financial compensation for this study.

4.2. Participants

This user evaluation study had 30 participants, which is a common number for studies on human-computer interaction and spatial navigation. Usability testing and HCI studies typically involve 12 to 30 participants. Within-subjects designs frequently require fewer participants than between-subjects designs due to their enhanced statistical power [51, 52]. Nielsen found that five users can find about 85% of usability problems. This set a standard for how to do user testing quickly and effectively [53]. But for quantitative comparisons and statistical analysis, bigger groups are still needed. Virzi discovered that 8 to 12 users can identify 80-90% of usability issues; however, larger samples facilitate the detection of more significant differences [54]. In studies on spatial navigation and wayfinding, the number of participants is usually between 15 and 30 [55, 56]. Studies on virtual reality navigation usually have 20 to 40 people [57, 58], and studies on emergency evacuation simulations usually have 20 to 50 people [59, 60]. The 30 participants in this study conform to the established ranges, yielding sufficient statistical power for within-subject comparisons.

Sample size is chosen based on statistical power, practical limits, and the needs of the within-subjects design. Based on the literature, the study could find significant differences if there were 30 people in each experimental group. Power analysis shows that a within-subjects design with 30 participants can find medium effect sizes ($d=0.5$) with about 80% power at $\alpha=0.05$ [61, 62].

Participants are needed to have basic computer using skills such as keyboard and mouse usage. This requirement made sure that everyone could use the navigation system without any problems that could affect the results.

4.3. Procedure

The experiment used a within-subjects design with counterbalancing to compare two navigation scenarios. In Scenario A, participants navigated without AI help, while in Scenario B, they used AI assistance with LLM-based voice interaction. All participants completed both scenarios in different orders to reduce learning and carry-over effects.

Participants were randomly assigned to two equal-sized groups. Group A (n=15) completed Scenario A first, which is evacuation without any AI help, then moved on to Scenario B, which includes AI-assisted navigation. Group B (n=15) did the scenarios in the opposite order, starting with Scenario B then moved to Scenario A. This counterbalancing method helps to distribute any possible learning, practice, or order effects evenly between both groups, which keeps the comparison fair [63].

To reduce the learning effects, two isomorphic building models were also used. These models are structurally identical except for their spatial arrangement, keeping the same dimensions, room layouts, and navigational complexity [64, 65]. This method is commonly used in spatial navigation research to control for environmental factors and allow repeated testing [56, 66]. Both building models had the same room sizes and layouts, corridor structures and widths, door placements, exit locations relative to rooms, and overall building size. The main differences were in their spatial orientation and room numbering, which stopped participants from memorizing specific routes but kept the navigation difficulty the same. Additionally, a door and wall pair is added one of the models which do not influence on decision points, to only provide more complexity in memorising the path, which can be observed the Grids 3 to G-J in Figure 12.

A wash-out period was used between the two experimental scenarios to reduce carryover effects and stop participants from memorizing routes [67, 68]. During this time, participants took a break and did tasks unrelated to spatial navigation or building environments. The wash-out period helped separate the scenarios which gave participants time to reset. It also reduced the possibility that the first task would affect the second task [69, 70]. This is a common practice in experimental design, especially in crossover trials and repeated measures studies [67, 68].

4.3.1. Scenario A: Baseline Navigation (Without AI Assistance)

In Scenario A, participants walked around one of the isomorphic building models using only keyboard controls without any AI assistance. They used standard first-person control buttons of the keyboard only. There was no AI help and voice interaction in this scenario.

Plan view of the Scenario A building can be seen in the Figure 13. Player starter position is marked with circle, and the exit building is highlighted with arrow.

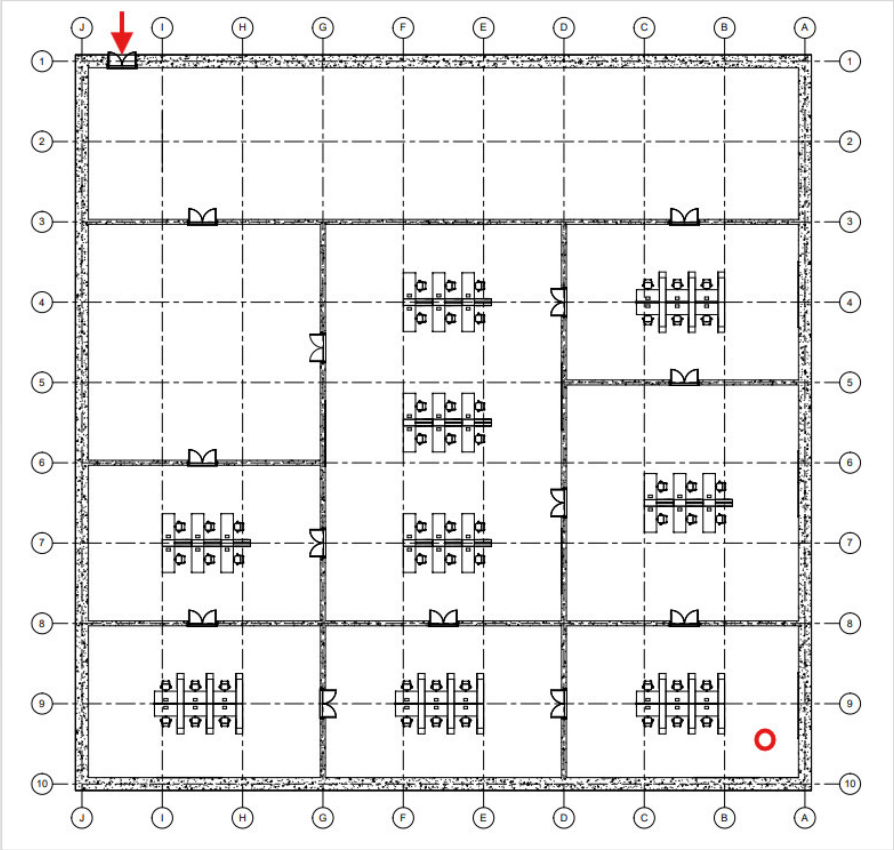


Figure 13: Revit plan view of the building Scenario A.

The scenario simulates a traditional navigation task. Without any AI-assistance, users can only use their spatial awareness and. This baseline is used as a reference for comparing the AI-assisted condition, to identify the effects of AI assistance. The timer for Scenario A started as soon as participants used the movement keys. Participants were told to reach the exit as quickly as possible and avoid hitting walls, doors, or obstacles.

4.3.2. Scenario B: AI-Assisted Navigation (With LLM)

In Scenario B, participants walked around in the second isomorphic building model with AI assistance. Before starting the navigation timer, they asked some questions to the LLM The game time is not measured in this stage to ensure a fair comparison between scenarios.

Figure 14 shows the plan view of the Scenario B building. The player's starting position is marked with a circle, and the exit is highlighted with an arrow.

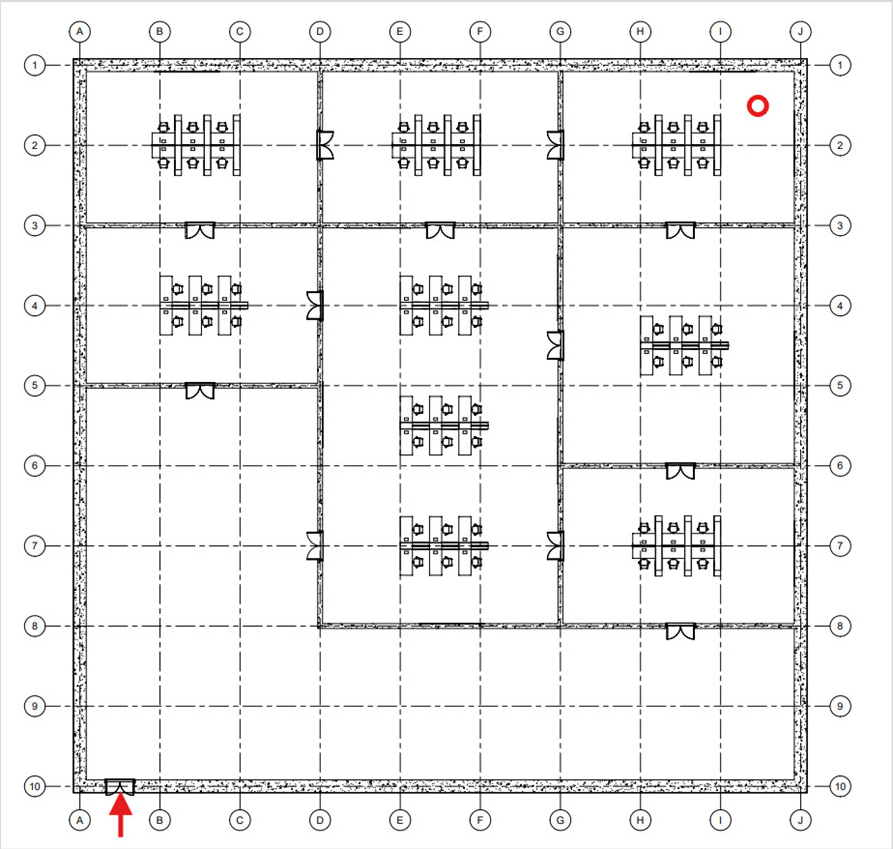


Figure 14: Revit plan view of the building Scenario B.

Participants were told to press the "M" key to start the voice interaction system. By doing that, they could ask questions about where they were, how far they were from the exit, the room details that they are currently in, and some irrelevant questions. This setup let participants get spatial information from the AI assistant without it affecting their ability to navigate. Then, they started moving by pressing the W, A, S, and D keys. The time measurement is started with the first movement. The AI assistant can be actively used if users have further questions. However, this feature is not used in the evacuation process since this might affect the evacuation time.

4.3.3. Time Measurement Protocol

The time measurement protocol makes sure that the two scenarios can be compared fairly and accurately. The measurement does not include time spent during the game starting or QA procedure. In both scenarios, the timer starts when the first movement key (W, A, S, or D) is pressed. This method makes sure that the measurement starts exactly when navigation starts. This protocol makes sure that time measurements only show how well AI assistance in navigation is working. [1]

4.3.4. Data Collection Methods

The data collection method used both objective performance metrics and subjective usability assessments to comprehensively evaluate the system. This approach aims to provide quantitative data on system performance. It also allows for the collection of feedback on user experience. This method is consistent with standard practices commonly used in Human-Computer Interaction (HCI) research. In HCI studies, both quantitative results and user feedback are considered important for understanding how well a system works [70, 71]. [1]

4.3.5. Time Measurement Comparison

Navigation time was used as the key performance metric in this study. Shorter times indicate more efficient navigation. Therefore, navigation time was carefully measured as mentioned earlier. The aim is to accurately capture the true performance differences between different conditions. Time-based metrics play a significant role in navigation research and user studies because they provide objective data showing how efficiently tasks are completed.[1]

4.3.6. Usability Questionnaires

After completing both experimental scenarios, participants evaluated the AI-assisted navigation system (Scenario B) using standard usability questionnaires. Scenario A was not included in the evaluation; it is used only for time comparison. The main focus of the study is the AI-powered system that can be used in practice. Therefore, understanding users' opinions about the system's usability and satisfaction was considered the most important evaluation objective. [1]

4.3.6.1 System Usability Scale (SUS)

System Usability Scale (SUS) is a commonly used questionnaire to measure users' perceptions of usability. [74, 75]. John Brooke proposed the SUS in 1996, and it is still a standard tool since it is easy to use, reliable, and valid [77]. There are 10 questions on the SUS, and each one is rated on a 5-point Likert scale from "Strongly Disagree" to "Strongly

[1] Kindly check the DISCLAIMER (p. 91)

Agree". The overall results give a single score between 0 and 100, with higher scores indicates higher usability. The SUS is a quick and effective way to measure how usable a system is. This is helpful for comparing systems and setting standards. A lot of research has shown that the scale is accurate, and it works well with other measures of usability [78, 79, 80]. The SUS can be used with many different systems and interfaces since it does not focus on a single area or technology. [1]

The 10 questions in the SUS check different aspects of usability, like how easy it is to learn, how well it works, how easy it is to remember, how mistakes are handled, and how satisfied users are. There is both positive and negative questions to avoid bias in the answers. The SUS gives a full picture of how usable a system is, which helps to researchers to understand how users feel about system and possible changes required. [1]

4.3.6.2 Usefulness, Satisfaction, and Ease of Use (USE) Questionnaire

The USE questionnaire is developed by Arnold Lund. This questionnaire evaluates three fundamental aspects of user experience: Usefulness, Satisfaction, and Ease of Use [81]. Each section consists of multiple questions measured using a 7-point Likert scale. This structure helps to evaluate different aspects of user experience in detail. It is used to evaluate user experience in a multidimensional way, focusing on three core dimensions: usefulness, satisfaction, and ease of use. Usefulness measures how users perceive whether the system helps them perform their tasks more quickly and efficiently. This dimension shows the value and benefit users see in the system. In this context, usefulness refers to how AI helps people find their way more easily and quickly. The satisfaction dimension evaluates how satisfied users are while using the system and shows the emotional aspect of the user experience. This element is one of the important factors determining whether users will continue to use the system; a high level of satisfaction indicates that users find the system both useful and enjoyable. Ease of use measures how simple and understandable users find the system. This feature is especially important for emergency systems, since complex processes can slow down people's actions in emergencies. Easy-to-use systems require less training and can be easily used by more people. Overall, the USE survey is considered as a reliable and valid measurement tool. By evaluating multiple dimensions together, it provides more detailed results compared to tools that focus on a single metric, thus helping to investigate which aspects of the system are working well and which areas need improvement. [1]

4.3.7. Questionnaire Administration

Participants completed the SUS and USE questionnaires immediately after finishing the two experimental scenarios. This method ensured that their experiences were still fresh and that they had observed all the features of the system. All participants were given the same instructions. Participants were asked to focus only on Scenario B, the AI-assisted navigation system. This is important because participants also tried Scenario A for the

[1] Kindly check the DISCLAIMER (p. 91)

comparison. The aim of the study was to ensure that the feedback focused on the AI-assisted navigation system being evaluated.

In addition to standard questionnaires, participants also provided their textual feedback. This feedback includes the system's strengths, weaknesses, and suggestions for improvement. This qualitative data helped to better understand user opinions addition to the SUS and USE scores.

CHAPTER 5

RESULTS

5.1. LLM Benchmark Results

This chapter shows the results of benchmark tests measuring the navigation command generation and QA performance of LLMs. The evaluation focused on two main areas: navigation command generation and QA performance.

In navigation command generation tests, each model was run 500 times. Different LLMs are used in this test. The evaluation compared a baseline method, two OpenAI models (GPT-4o and GPT-4o-mini), and two Google Gemini models (Gemini-2.0-flash-exp and Gemini-2.5-flash). The baseline method is used as a reference considered as the best possible geometric pathfinding performance. In each trial, the model generated navigation commands from a random starting point to the exit. Success rate, collision rate, path efficiency, and SPL metrics were used for evaluation.

Figure 15 shows three example trial outputs over the occupancy grids. Dark gray blocks represent walls and obstacles. The red circle indicates the starting point; the green circle indicates the exit. The blue line shows the route created. These examples demonstrate how agents find the optimal route from the starting point to the exit.



Figure 15: Command Simulation Across Three Evacuation Scenarios.

5.1.1. Success Rate

The baseline geometric method had a perfect success rate of 100.0% in all 500 trials, which was the best possible performance. Gemini-2.5-flash was also 100% successful in 500 trials, which is the same as the other LLM models. GPT-4o had the second highest rate, 99.8% (499 out of 500 successful trials), which was very close to the baseline. The success rates for GPT-4o-mini and Gemini-2.0-flash-exp were 96.2% (481 successful trials) and 94.2% (471 successful trials), respectively. Figure 16 shows how these success rates compare.

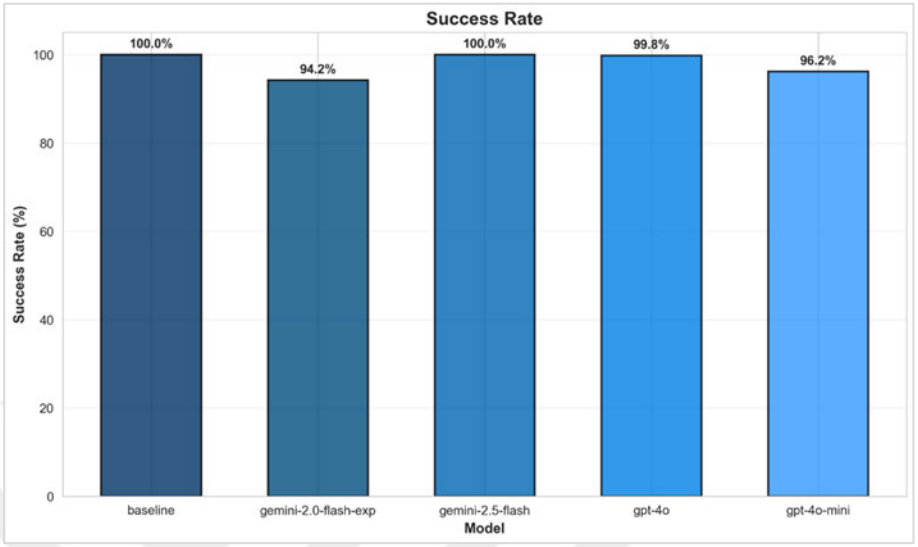


Figure 16: Success rate results of the baseline model and four LLMs across 500 trials.

5.1.2. Collision Rate

Gemini-2.5-flash and Gemini-2.0-flash-exp both avoided collisions completely, with a collision rate of 0.0%. GPT-4o and GPT-4o-mini had very low collision rates of 0.2%, which means they were very aware of their surroundings and good at avoiding obstacles. Figure 17 shows how the collision rates compare.

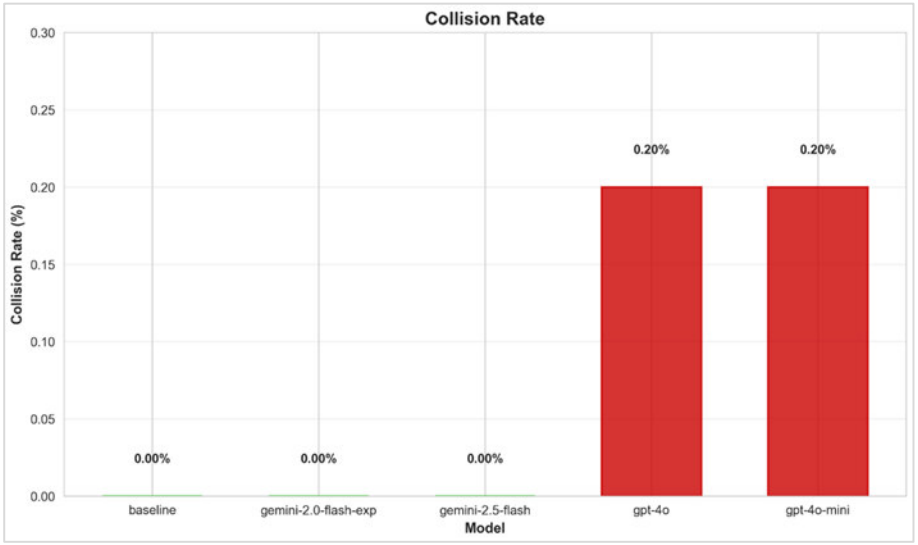


Figure 17: Collision rate results of the baseline model and four LLMs across 500 trials.

5.1.3. Path Ratio

The baseline method had a path ratio of 0.951, which is the standard for how well a path works. GPT-4o's path ratio was 0.949, which is very close to the baseline. This shows that the route optimisation was very good. The path ratio for GPT-4o-mini was 0.915, and the path ratio for Gemini-2.0-flash-exp was 0.896. Gemini-2.5-flash had a path ratio of 0.951 over 500 trials, which was the same as the baseline. These path ratio comparisons can be seen in Figure 18.

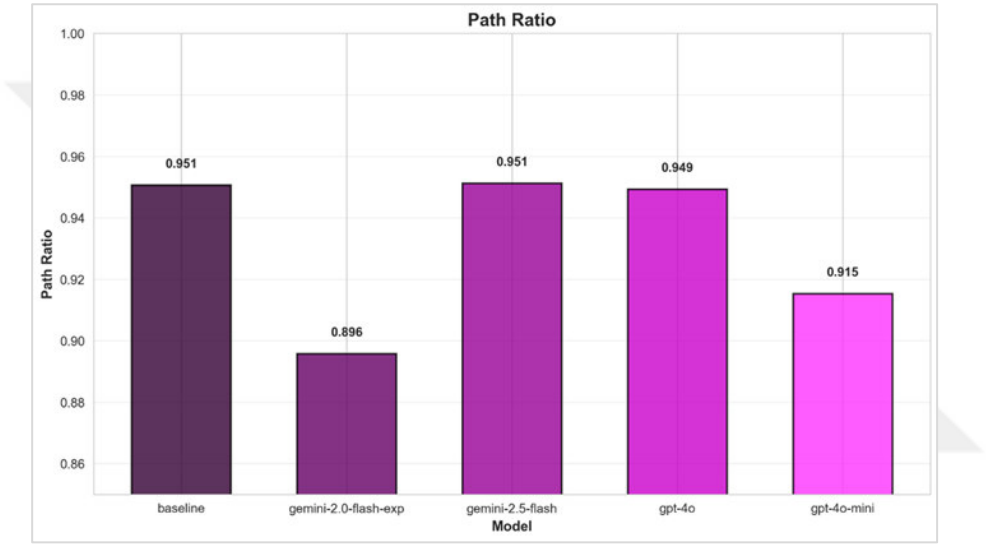


Figure 18: Path ratio results of the baseline model and four LLMs across 500 trials.

5.1.4. SPL (Success weighted by Path Length)

The baseline method reached a perfect SPL of 1.000, showing complete success and nearly optimal path efficiency. Gemini-2.5-flash also scored a perfect SPL of 1.000 in 500 trials. GPT-4o had an SPL of 0.998, which is very close to the baseline and shows a high success rate and efficient path planning. GPT-4o-mini scored 0.962, and Gemini-2.0-flash-exp scored 0.942. Figure 19 shows the SPL comparisons.

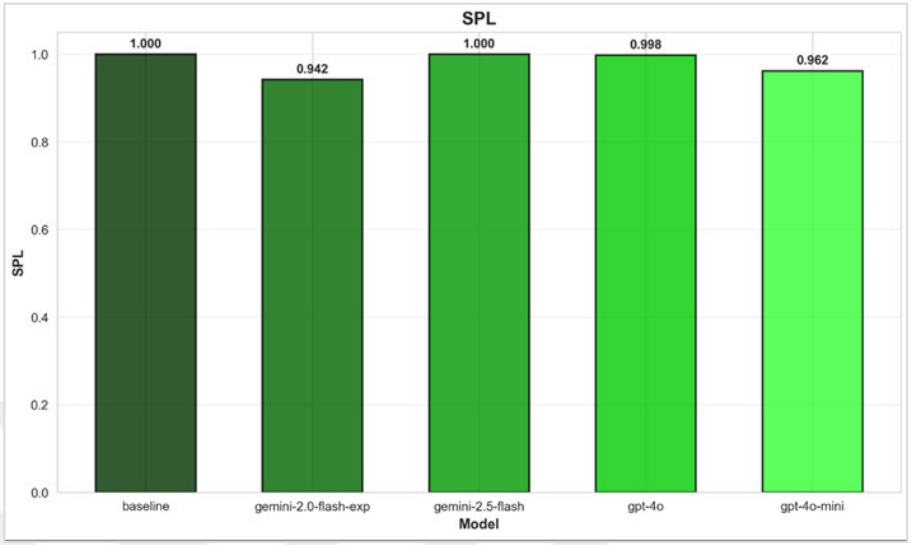


Figure 19: SPL results of baseline model and four LLM models across 500 trials.

5.1.5. Average Commands

The baseline method used an average of 13.06 commands per trial. GPT-4o averaged 13.07 commands, which is almost the same as the baseline. This suggests that the LLM creates command sequences with similar complexity to geometric pathfinding. GPT-4o-mini averaged 12.77 commands, slightly fewer than the baseline, and Gemini-2.0-flash-exp averaged 12.46 commands. Gemini-2.5-flash averaged 14.27 commands across 500 trials. Average commands comparisons can be seen in the Figure 20.

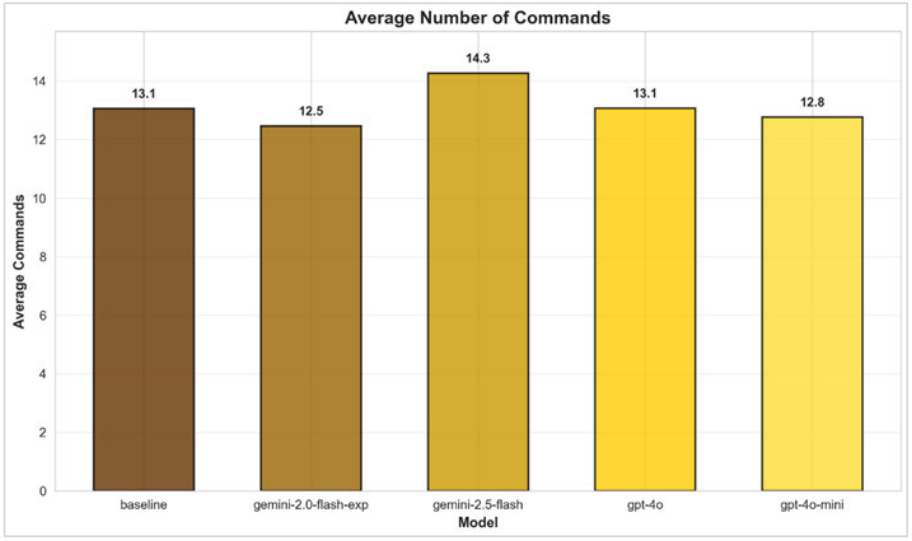


Figure 20: Average number of commands results of baseline model and four LLM models across 500 trials.

5.1.6. Average Latency

Gemini-2.0-flash-exp had the lowest average latency at 1.11 seconds, making it the fastest for navigation command generation. GPT-4o had 1.52 seconds, indicating a good balance between performance and speed. GPT-4o-mini averaged 2.70 seconds, and Gemini-2.5-flash averaged 2.25 seconds over 500 trials. Figure 21 shows the average latency comparisons.

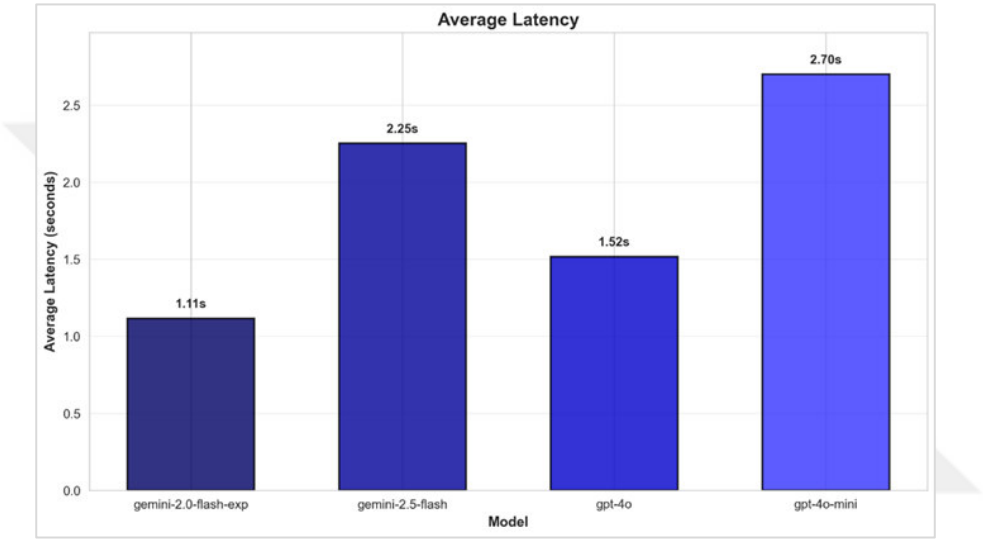


Figure 21: Average latency results of four LLMs.

5.2. Question-Answering Results

The QA evaluation showed how well the models could provide information about the user's questions and reject irrelevant questions. The evaluation is performed at three spots in the building, with 14 questions at each, 8 about spatial context and 6 irrelevant, for a total of 42 questions per model. Four models were tested: GPT-4o, GPT-4o-mini, Gemini-2.0-flash-exp, and Gemini-2.5-flash. The evaluation checked if the models gave the right answers, included the expected information for relevant questions, and rejected irrelevant ones.

5.2.1. Appropriate Answer Rate

GPT-4o, GPT-4o-mini, and Gemini-2.5-flash each had a perfect appropriate answer rate of 100%, showing they understood the questions and gave suitable responses. Gemini-2.0-flash-exp had a 71.4% rate, which suggests it sometimes struggled to provide appropriate answers. The appropriate answer rate comparisons can be seen in the Figure 22.

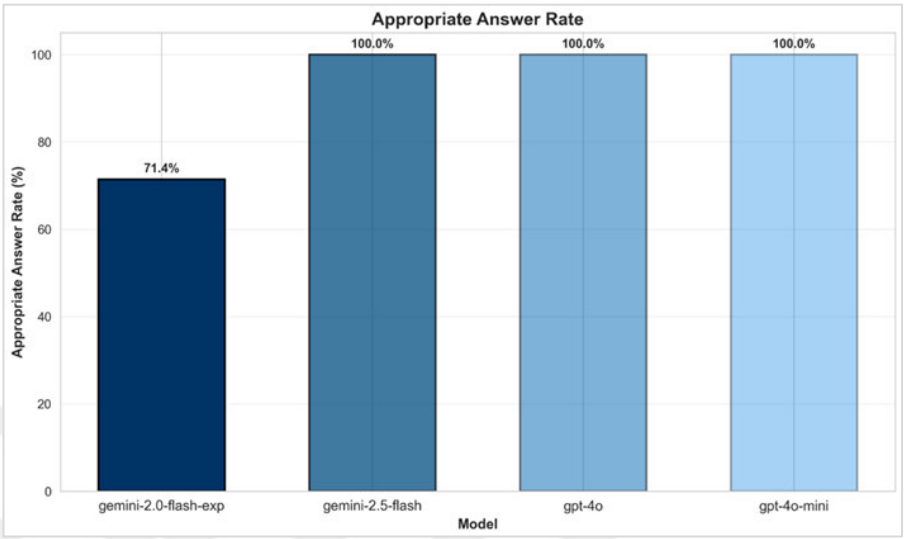


Figure 22: Appropriate answer rate results of four LLMs.

5.2.2. Contains Info Rate

GPT-4o, GPT-4o-mini, and Gemini-2.5-flash each had a perfect contains info rate of 100.0%, meaning they always provided the necessary spatial information for relevant navigation questions. Gemini-2.0-flash-exp had a contains info rate of 87.5%, so it performed well, but sometimes missed expected details. Figure 23 shows a comparison of these rates.

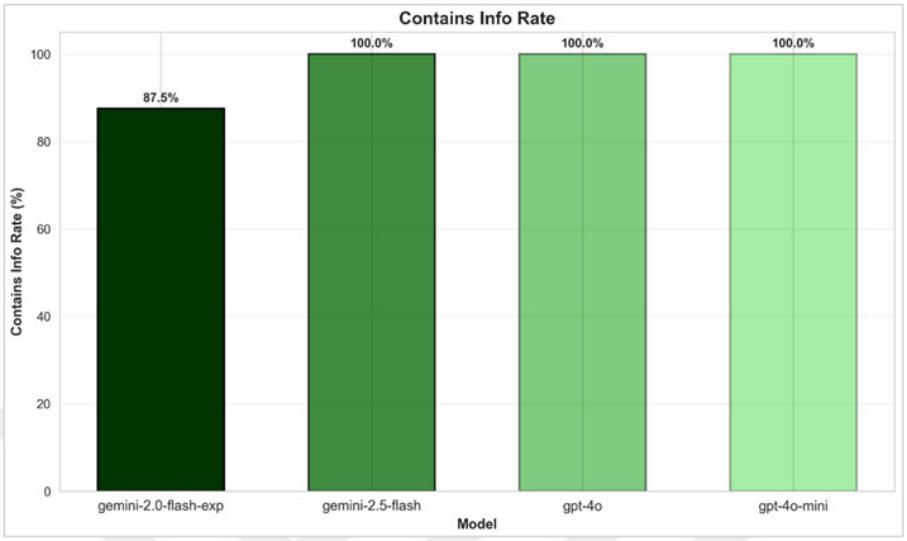


Figure 23: Contains info rate results of four LLMs.

5.2.3. Rejects Irrelevant Rate

GPT-4o, GPT-4o-mini, and Gemini-2.5-flash each had a perfect rejection rate of 100%, showing they could reliably spot and reject questions that did not fit the navigation context. In contrast, Gemini-2.0-flash-exp had a 50% rejection rate, suggesting it struggled to consistently identify irrelevant questions. Figure 24 shows the comparison of rejection rates of the models.

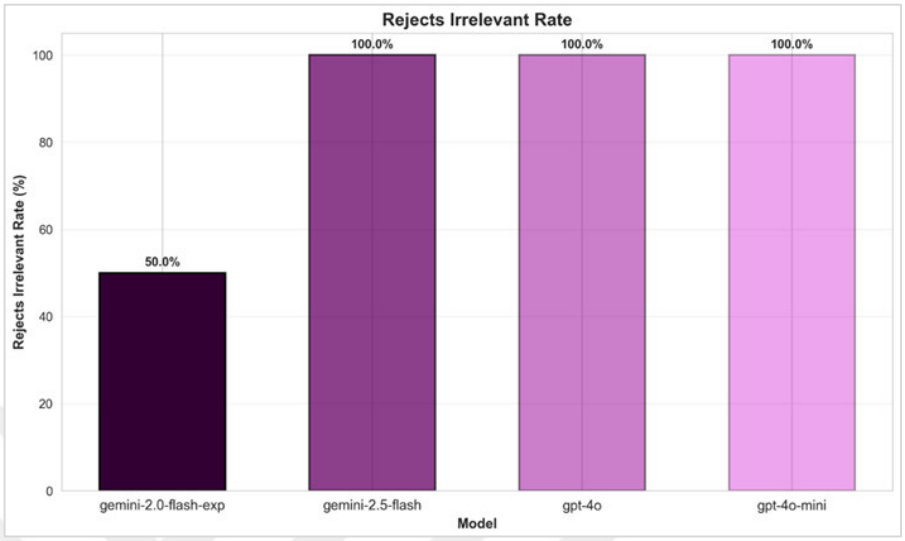


Figure 24: Rejects irrelevant rate results of four LLMs.

5.2.4. QA Latency

The QA evaluation includes how quickly each model responded to user questions. Gemini-2.0-flash-exp was the fastest, with a latency of 0.56 seconds. GPT-4o followed at 0.64 seconds, then GPT-4o-mini at 0.70 seconds, and Gemini-2.5-flash at 0.77 seconds. Figure 25 shows the average latency comparisons.

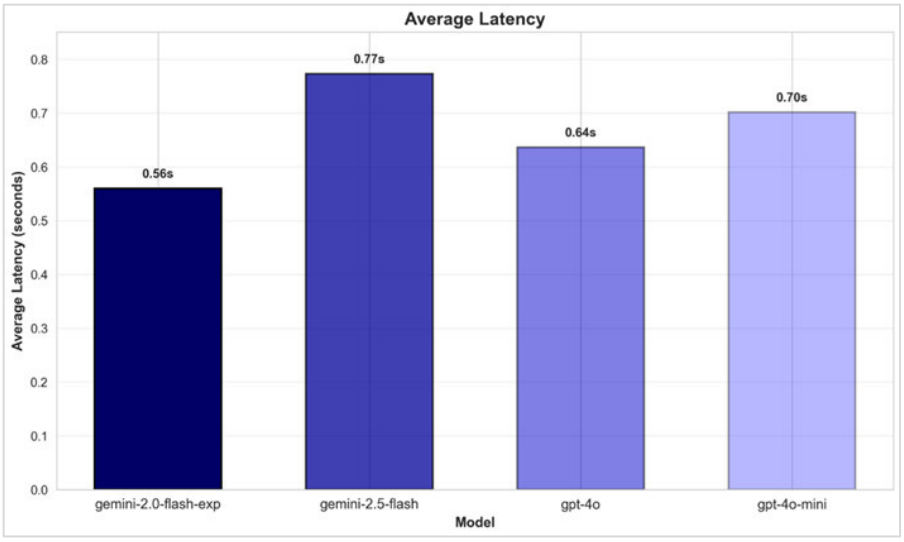


Figure 25: Average latency results of four LLMs.

5.3. User Evaluation Results

5.3.1. System Usability Scale

30 participants filled out the System Usability Scale (SUS) after using the AI-assisted navigation system. The SUS results show that users had a positive view of the system's usability. The average SUS score is 78.33, and the median is 80.00. Scores ranged from 52.50 to 100.00. 60% of the participants (18 users) rated it as "Excellent" (SUS score 80 or above), 23.3% of the participants (7 users) rated it as "Good" (SUS score 68-79), and 16.7% of the participants (5 users) rated it as "OK" (SUS score 51-67). No participant rated the system as "Poor" (SUS score below 51). Table 1 shows the descriptive statistics, and Table 2 shows the score distribution.

Table 1: Descriptive Statistics of SUS Scores

Descriptive Statistics						
Metric	**Mean**	**Median**	**Standard Deviation**	**Minimum**	**Maximum**	**Range**
SUS Score	78.33	80.0	10.99	52.50	100.0	57.5

Table 2: Category Distribution of the SUS Scores

Category	**Score Range**	**n**	**Percentage (%)**
Excellent	≥ 80	18	60.0
Good	68–79	7	23.3
OK	51–67	5	16.7
Poor	< 51	0	0
Total	—	**30**	**100.0**

5.3.1.1 SUS Score Box Plot

Figure 26 shows the box plot of SUS scores. The main is 78.3, shown with cross mark. The median is 80.0, shown with a thick line. The interquartile range ,s from 70.0 (Q1) to 87.5 (Q3), so half of the participants scored between these values. The upper whisker is at 100.0, which is the highest score within 1.5 times the interquartile range above Q3. The lower whisker goes down to 52.5, the lowest score within 1.5 times the interquartile range below Q1. There are no outliers below the lower whisker since the minimum score of 52.5 is within the expected range. [1]

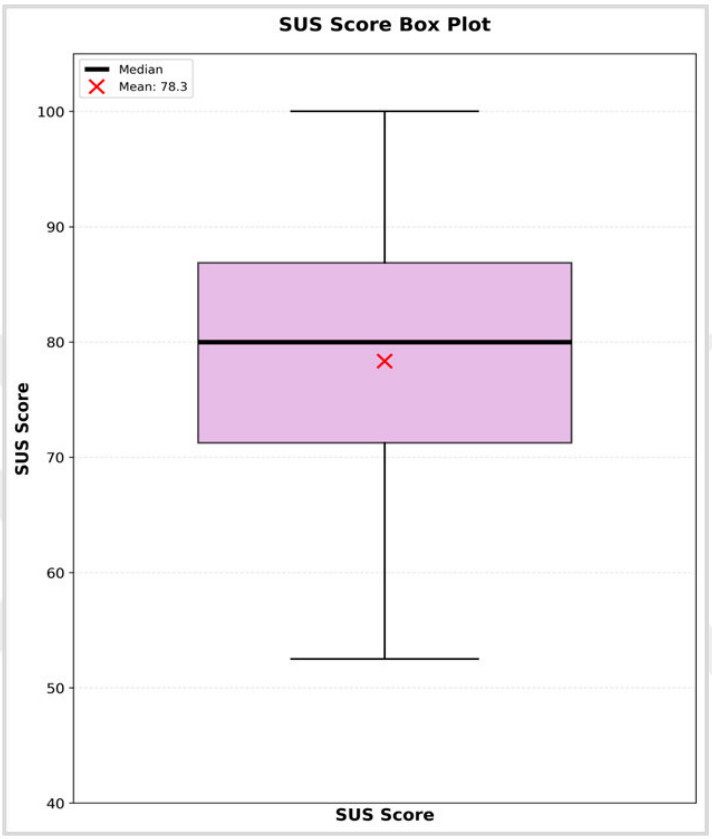


Figure 26: SUS score box plot.

5.3.1.2 SUS Score Interpretation Distribution

Figure 27 shows how SUS scores are distributed among different categories. Sixty percent of participants (18 users) rated the system as "Excellent," 23.3% (7 users) rated it as "Good," 16.7% (5 users) chose "OK," and none rated it as "Poor." Most participants (60%) gave the system an "Excellent" rating, which suggests they had a very positive view of its usability. [1]

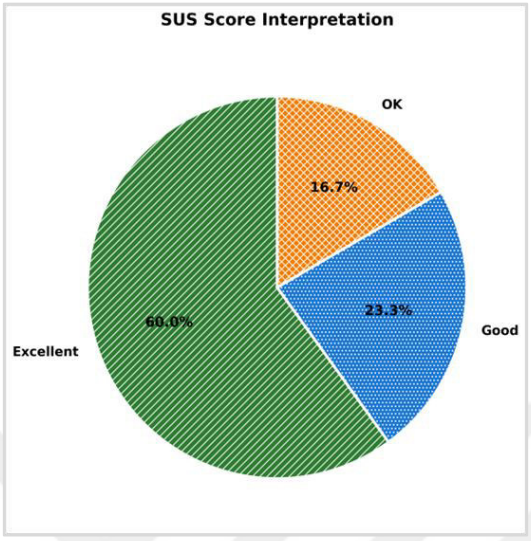


Figure 27: Pie chart of the SUS score interpretation.

5.3.2. Usefulness, Satisfaction, and Ease of Use (USE) questionnaire

Thirty participants completed the Usefulness, Satisfaction, and Ease of Use (USE) questionnaire after trying the AI-assisted navigation system. The USE questionnaire measures four areas: usefulness, ease of use, ease of learning, and satisfaction, on a 7-point scale, where higher scores mean more positive feedback. Usefulness included 5 questions, Ease of Use had 9, Ease of Learning had 4, and Satisfaction had 6, with each item rated from 1 (strongly disagree) to 7 (strongly agree). Results showed positive feedback in all areas. The average Usefulness score was 6.48, and the median was 6.54. For Ease of Use, the average was 6.56, and the median was 6.68. Ease of Learning had an average of 6.64 and a median of 6.72. Satisfaction averaged 6.53, with a median of 6.58. The overall USE score, which is the average of 4 dimensions, had a mean of 6.55 and a median of 6.62. Scores ranged from 5.50 to 7.00, shows that participants rated the system positively. Table 3 summarizes the USE questionnaire results.

Table 3: USE Questionnaire Dimensions

USE Questionnaire Dimensions						
Metric	**Mean**	**Median**	**Standard Deviation**	**Minimum**	**Maximum**	**Score Range**
Usefulness	6.48	6.54	0.39	5.50	7.00	1.50
Ease of Use	6.56	6.68	0.40	5.56	7.00	1.44
Ease of Learning	6.64	6.72	0.39	5.58	7.00	1.42
Satisfaction	6.53	6.58	0.41	5.50	7.00	1.50
Overall USE	6.55	6.62	0.39	5.53	7.00	1.47

5.3.2.1 USE Score Box Plot

Figure 28 shows a box plot of Usefulness scores, with a median of 6.54 and a mean of 6.48. The interquartile range is about 6.25 (Q1) to 6.75 (Q3), covering the middle half of participant scores. The whiskers go from about 5.50 to 7.00, which includes scores within 1.5 times the IQR from the quartiles. Outliers appear as separate points.

The box plot for Ease of Use scores (Figure 29) has a median of 6.68 and a mean of 6.56. The interquartile range goes from about 6.25 (Q1) to 6.95 (Q3), covering the middle half of participant scores. The whiskers stretch from roughly 5.55 to 7.00, showing the range of scores within 1.5 times the IQR from the quartiles. Any outliers appear as separate points.

The box plot for Ease of Learning scores (Figure 30) has a median of about 6.70 and a mean of 6.64. The interquartile range goes from roughly 6.40 (Q1) to 7.00 (Q3), covering the middle half of the scores. The whiskers ranges from about 5.55 to 7.00, showing the range within 1.5 times the IQR, and any outliers appear as separate points.

The box plot of Satisfaction scores (Figure 31) shows a median of 6.58 and a mean of 6.53. The interquartile range (IQR) goes from around 6.30 (Q1) to 6.85 (Q3), covering the middle half of participant scores. The whiskers stretch from about 5.50 to 7.00, showing the range of scores within 1.5 times the IQR from the quartiles. Any outliers appear as separate points.

Finally, Figure 32 shows a box plot of Overall USE scores, with a median of 6.62 and a mean of 6.55. The interquartile range goes from about 6.30 (Q1) to 6.90 (Q3), covering the middle half of participant scores. The whiskers stretch from roughly 5.50 to 7.00, which includes scores within 1.5 times the IQR from the quartiles. Any outliers appear as separate points.

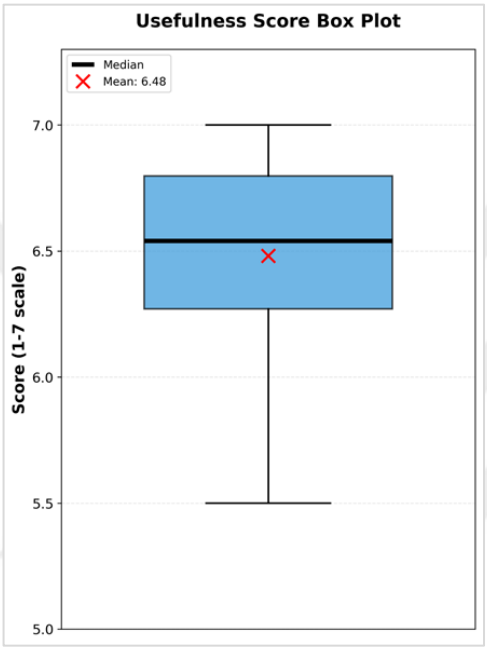


Figure 28: Usefulness score box plot.

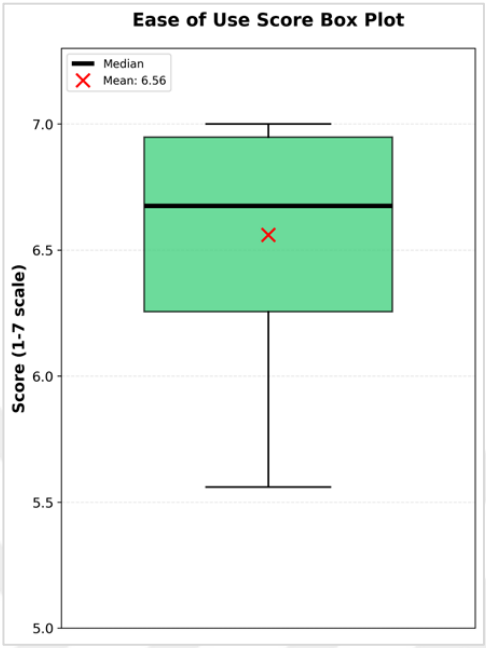


Figure 29: Ease of Use score box plot.

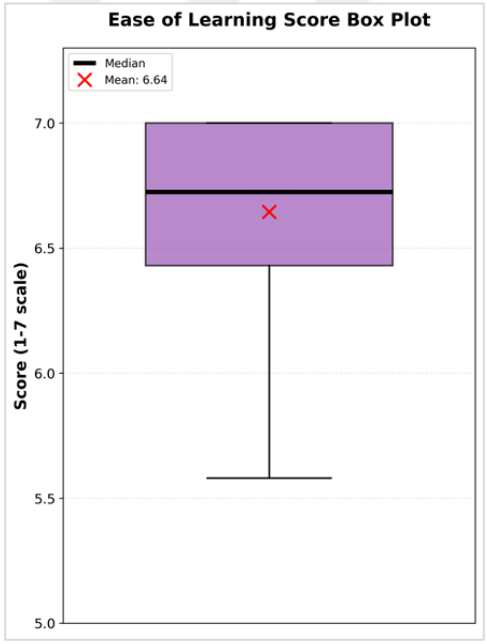


Figure 30: Ease of Learning score box plot.

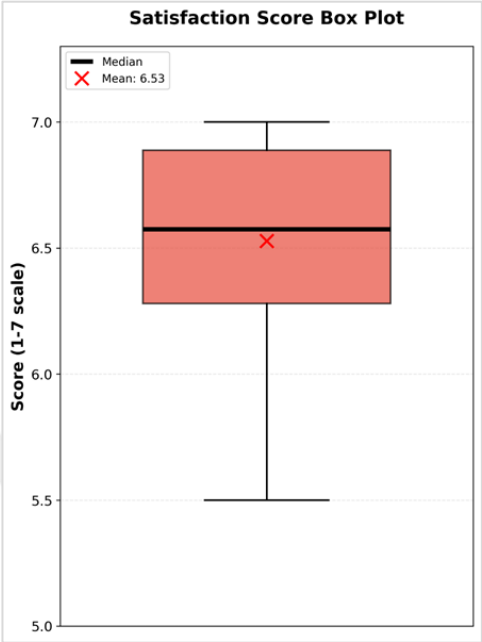


Figure 31: Satisfaction score box plot.

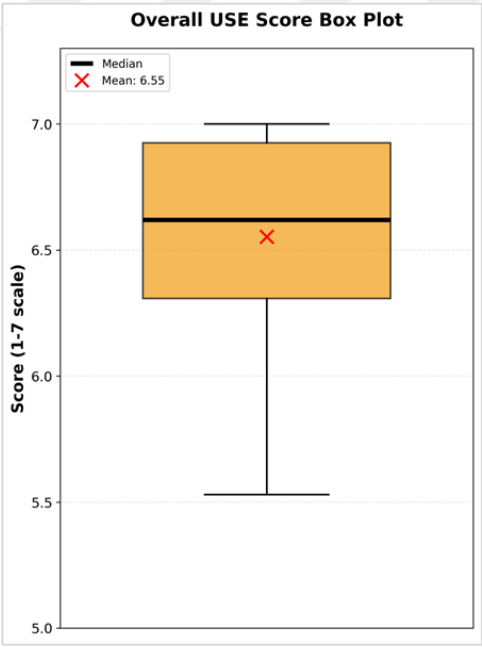


Figure 32: Overall USE score box plot.

5.3.3. Correlation Analysis Between SUS and USE Scores

Pearson correlation analysis examined the relationship between SUS scores and USE dimensions. Scatter plots with regression lines show that there is a positive linear relationship.

Figure 33 shows how SUS scores and Usefulness scores are related. On the x-axis, it can be seen SUS scores from 55 to 100, and on the y-axis, it can be observed Usefulness scores from 5.4 to 7.2. The regression line shows a straight line that goes up. The Pearson correlation coefficient is $r = 0.717$ ($p < 0.001$), which means that the two things are strongly related.

Figure 34 shows how SUS scores and Ease of Use scores are related. The x-axis shows SUS scores from 55 to 100, and the y-axis shows Ease of Use scores from 5.4 to 7.2. The regression line shows that there is a positive linear relationship. The Pearson correlation coefficient is $r = 0.701$ ($p < 0.001$), which means that there is a strong positive relationship.

Figure 35 shows how SUS scores and Ease of Learning scores are related. The x-axis shows scores for SUS (55 to 100) and the y-axis shows scores for Ease of Learning (5.4 to 7.2). The regression line shows a straight line that goes up. The Pearson correlation coefficient is $r = 0.709$ ($p < 0.001$), which means that there is a strong positive relationship.

Figure 36's scatter plot shows how SUS scores and Satisfaction scores are related. The x-axis shows SUS scores from 55 to 100, and the y-axis shows Satisfaction scores from 5.4 to 7.2. The regression line shows a straight line that goes up. The Pearson correlation coefficient is $r = 0.685$ ($p < 0.001$), which means there is a strong positive relationship.

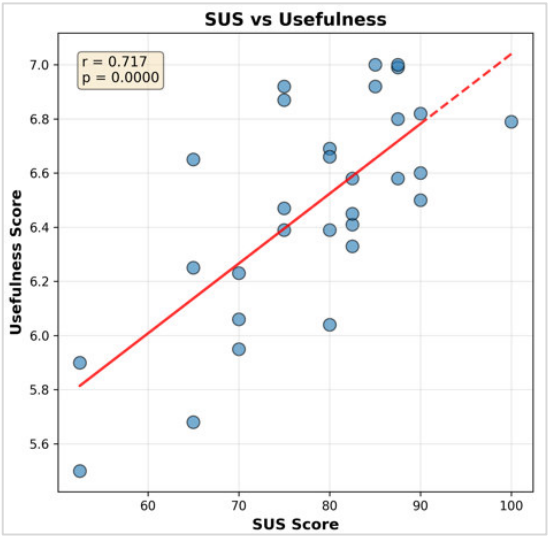


Figure 33: Correlation between SUS scores and Usefulness scores.

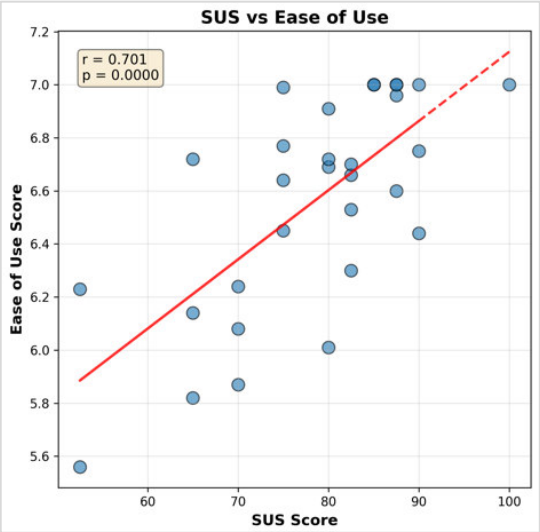


Figure 34: Correlation between SUS scores and Ease of Use scores.

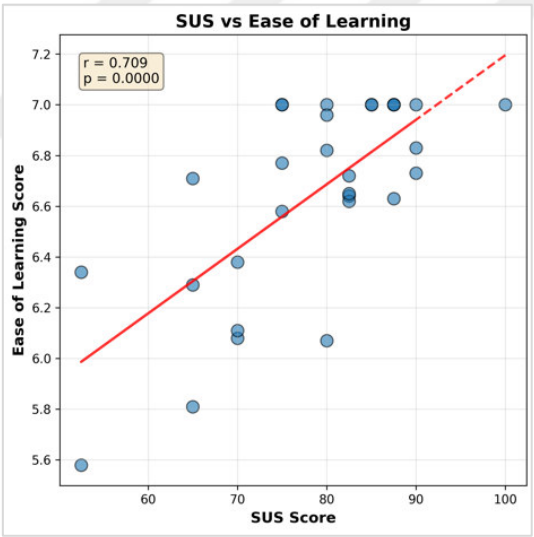


Figure 35: Correlation between SUS scores and Ease of Learning scores.

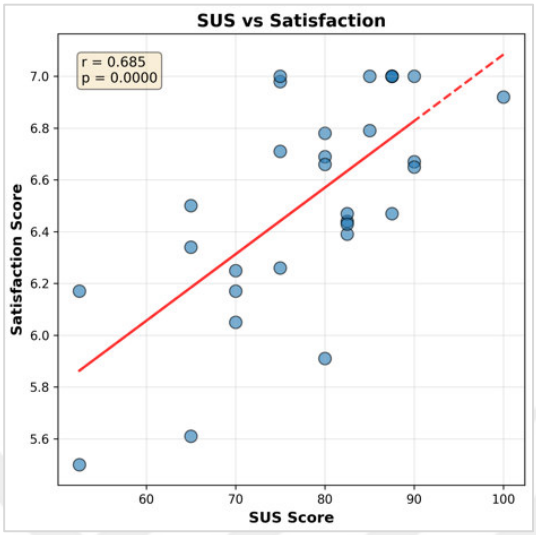


Figure 36: Correlation between SUS scores and Satisfaction scores.

5.3.4. User Feedback Results

User feedback from the testing process highlights the system's strengths, weaknesses, and suggestions for improvement. The word frequency maps below show the most important words and ideas from their responses. Figure 37 shows the most common words for system strengths, Figure 38 for weaknesses, and Figure 39 for user suggestions.

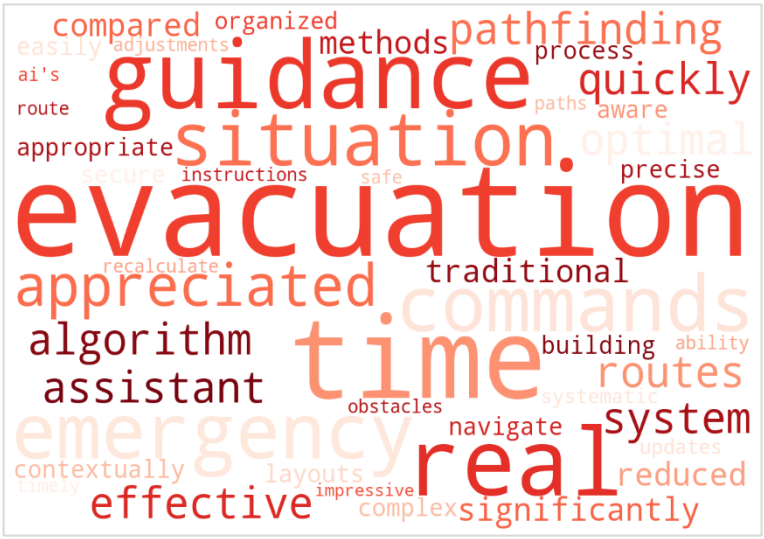


Figure 37: Frequency Mapping of most used words about system's strengths.



Figure 38: Frequency Mapping of most used words about system's weaknesses.

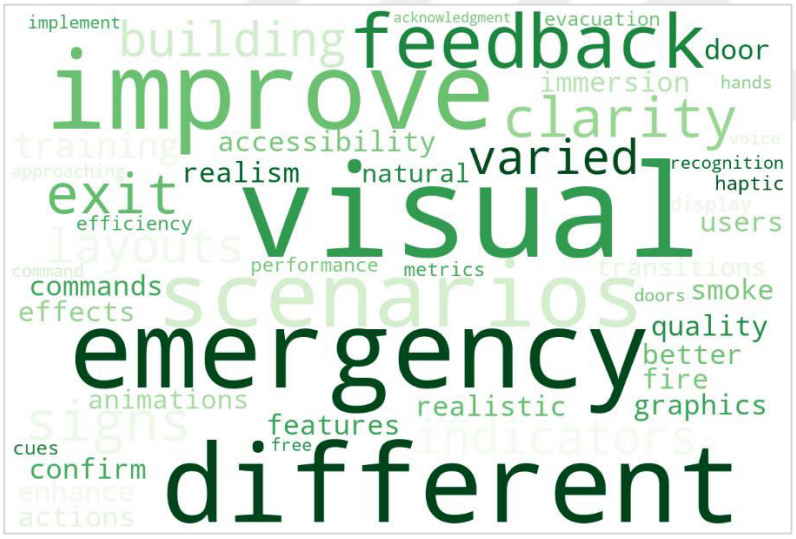


Figure 39: Frequency Mapping of most used words about users' suggestions for improvement.

5.3.5. Time Measurement Results

The task completion time analysis shows that participants completed Scenario A in approximately 2 minutes and 7.80 seconds (127.80 seconds). Scenario B was completed in approximately 58.25 seconds. The difference between the two scenarios is approximately 69.55 seconds.

CHAPTER 6

DISCUSSION

This chapter discusses the findings obtained from both LLM benchmark and user evaluations. Results from navigation and question-answer benchmark tests have been compiled, and feedback regarding user experience has also been evaluated. This approach aims to examine the performance of the AI-assisted emergency evacuation system more comprehensively.

6.1 Discussion of LLM Benchmark Results

Benchmark evaluations conducted over 500 trials for navigation and question-and-answer tasks provide valuable insights into the models' performance in emergency evacuation scenarios. Four models were examined in the evaluation: GPT-4o, GPT-4o-mini, Gemini-2.0-flash-exp, and Gemini-2.5-flash. Each model was evaluated across multiple performance metrics. These metrics aim to analyze how the models perform in different aspects.

6.1.1. Navigation Performance Analysis

Navigation benchmark results show the significant performance differences between the language models. The baseline model's performance is 100% by default since it is used as the reference. The GPT-4o model has shown a strong performance. It achieved a result very close to the baseline method with 99.8% success rate. This shows that GPT-4o can accurately understand the spatial information and generate reliable navigation commands. It failed only once over the 500 trials shows that the model is quite reliable for AI-assisted navigation systems. The Gemini-2.5-flash model is reached to the baseline performance with 100% success rate. This result shows that the model can successfully perform the navigation tasks. The GPT-4o-mini model achieved a 96.2% success rate. This result shows that the model generally performs well. However, the exit was not reached in 19 scenarios over 500 trials, indicating that the model's performance may decrease in some challenging scenarios. However, since it has lower cost, it can be a suitable alternative for projects where cost is important, and it continues to offer an acceptable level of reliability. The Gemini-2.0-flash-exp model achieved a 94.2% success rate and failed in 29 out of 500 trials. This result represents the lowest success rate among the tested models. However, this model has the fast response time. The average response time was measured at 1.11 seconds. For comparison, the average response time of the GPT-4o model was 1.52 seconds, the Gemini-2.5-flash model 2.25 seconds, and the GPT-4o-mini model 2.70 seconds. This high speed can provide a significant advantage, especially in emergency

scenarios requiring rapid response. Therefore, in system design, not only the success rate but also balancing response speed and accuracy should be considered.

Collision rate results show that all models performed high level of safety. Gemini-2.0-flash-exp and Gemini-2.5-flash models produced no collisions in 500 trials. The GPT-4o and GPT-4o-mini models had a collision rate of 0.2%, with only one collision over 500 trials. These low rates show that all tested models can generate navigation commands by protecting user safety. This is especially important in emergency scenarios.

Efficiency analysis is implemented by considering the path ratio metric. The results show that the models generally produce quite efficient routes. The reference model had a path ratio of 0.951. The GPT-4o model reached a value of 0.949, indicating that the routes it produced were approximately 94.9% more efficient than the best route found by the A* algorithm. The Gemini-2.5-flash model resulted the same performance as the baseline with a value of 0.951. The efficiency of the other models was slightly lower. The path ratio of the GPT-4o-mini model was measured at 0.915, and that of the Gemini-2.0-flash-exp model at 0.896, which was the lowest result among the tested models. Nevertheless, the findings show that while the models did not always produce the shortest route, they were generally able to create efficient navigation paths by reducing the unnecessary movements. This provides a significant advantage in terms of increasing efficiency during evacuation.

The SPL (Success Weighted by Path Length) metric is a measure that evaluates both success rate and route efficiency. The results show that the SPL values are quite close to the success rates for all models. This indicates that when models are successful, they can generally produce efficient and short routes. According to the results, the Gemini-2.5-flash model showed the best performance in terms of both success and route efficiency with an SPL value of 1.000. The GPT-4o model achieved a very close result with a value of 0.998. The GPT-4o-mini model reached 0.962, and the Gemini-2.0-flash-exp model reached 0.942 SPL. The fact that SPL values are generally quite close to success rates shows that effective and efficient routes are produced in most successful navigation attempts.

The number of commands required to reach the exit varies between approximately 12.5 and 14.3 depending on the model. The Gemini-2.0-flash-exp model uses the fewest commands, averaging 12.5, while the GPT-4o-mini model achieves a similar result with 12.8 commands. The Baseline and GPT-4o models use an average of 13.1 commands. The highest number of commands, 14.3, is seen in the Gemini-2.5-flash model; this suggests that the model may have generated more detailed or cautious commands during navigation. Nevertheless, the command sequences generated by all models are quite similar, indicating that they generally exhibit similar performance in generating navigation commands.

6.1.2. Latency Analysis

Latency tests show that the response times of the models differ. The Gemini-2.0-flash-exp model was the fastest, with an average response time of 1.11 seconds, making it the best option for real-time usage. The GPT-4o model showed a balanced result between speed and performance with an average response time of 1.52 seconds. The Gemini-2.5-flash model had an average response time of 2.25 seconds, showing a moderate latency. The slowest model was the GPT-4o-mini with an average response time of 2.70 seconds. This high latency could negatively impact user experience, especially in scenarios requiring fast responses. The Gemini-2.0-flash-exp is approximately 55% faster than the GPT-4o-mini, providing a time advantage of about 1.4 seconds per command. Such speed differences become an important factor in model selection, especially in real-time navigation and emergency applications.

These latency differences are important when designing systems. In emergency evacuations, for example, faster response times can help people get to safety more quickly. However, it is important to consider that quicker responses might sometimes reduce the accuracy which is also important for evacuation. Thus, both speed and success rate should be considered together.

6.1.3. Question-Answering Performance Analysis

Question-answering benchmark results show that although there are some differences between the model performances, overall high performance is achieved. The GPT-4o, GPT-4o-mini, and Gemini-2.5-flash models achieved 100% success in all QA metrics, including appropriate answer rate, contains information rate, and rejects irrelevant questions rate. It can refer that modern LLMs can correctly understand the context, provide the necessary information, and appropriately reject questions that are irrelevant to the system. However, the Gemini-2.0-flash-exp model is showed the lower performance. The model measured a 71.4% correct response rate, an 87.5% rate of informative responses, and only a 50% rate of rejecting irrelevant questions. The particularly low rejection rate indicates that the model struggles to reject irrelevant questions. This can create a risk of users encountering unnecessary information and becoming distracted in critical situations such as emergency evacuations.

QA latency results show that all models have very fast response times. The Gemini-2.0-flash-exp model had the fastest response time at 0.56 seconds, followed by GPT-4o at 0.64 seconds, GPT-4o-mini at 0.70 seconds, and Gemini-2.5-flash at 0.77 seconds. The fact that all models have response times under one second ensures that interaction remains natural and fluid, especially in emergency situations where users need quick information. These results demonstrate that QA features can be integrated into real-time evacuation systems without creating significant latency.

6.1.4. Model Comparison and Selection Considerations

Benchmark results reveal that no single model demonstrates the best performance across all criteria. Each model has strengths that stand out in different areas. The Baseline model and the Gemini-2.5-flash model achieved 100% success rates and the highest route efficiency values. While the Baseline model offers a deterministic reference point, the Gemini-2.5-flash model shows that LLM-based approaches can perform similarly to deterministic methods. The GPT-4o model demonstrated very strong performance in navigation tasks. With a 99.8% success rate and an almost perfect SPL value (0.998), it can be considered a suitable option for scenarios requiring high reliability. The GPT-4o-mini model offers a balanced alternative in terms of cost and performance. With a 96.2% success rate, this model also produced successful results in question-and-answer tasks. Thanks to these features, it stands out as a suitable option, especially for conversational interfaces. In contrast, the Gemini-2.0-flash-exp model is the fastest responding model and can offer an advantage for real-time applications with an average response time of 1.11 seconds per instruction. However, its lowest success rate (94.2%) and lowest route efficiency (0.896) among the tested models indicate some limitations in terms of accuracy. Therefore, the balance between speed, accuracy, and cost should be considered when selecting a model in system design.

The choice of model depends on the application's requirements. Reliability is a crucial factor, especially in systems that assist in the safe evacuation of people. In such cases, the GPT-4o stands out as the most suitable option among LLM models. It offers a balanced performance with a nearly perfect success rate and low latency. Gemini-2.5-flash can also be considered as an alternative; this model achieves the same success rate as the baseline but has a higher response time compared to GPT-4o model (average 2.25 seconds). For applications requiring more cost-effective solutions, GPT-4o-mini may be a good option. The model offers a cost-performance balance with a 96.2% success rate and moderate latency. In contrast, Gemini-2.0-flash-exp can provide an advantage for real-time scenarios requiring fast responses. With an average response time of 1.11 seconds, it is the fastest model. However, its 94.2% success rate and 0.896 path efficiency value indicate lower accuracy compared to other models. Therefore, researchers need to carefully consider the balance between speed, accuracy, and cost when selecting a model. [1]

Models used in user evaluation studies were selected considering performance metrics and real-world conditions. Accordingly, GPT-4o was chosen as the primary model for user testing. The main reason for this selection is the model's very high route efficiency (SPL: 0.998) and near-perfect success rate (99.8%). Furthermore, its average latency of 1.52 seconds is acceptable for real-time use. Therefore, using GPT-4o allows user evaluations to more accurately and reliably reflect the system's actual performance. The GPT-4o model also achieved highly successful results in QA tasks. The model demonstrated full success in all QA metrics with a 100% appropriate response rate, a 100% rate of

[1] Kindly check the DISCLAIMER (p. 91)

containing necessary information, and a 100% rate of rejecting irrelevant questions. Additionally, the average question-and-answer latency of 0.64 seconds indicates the system's ability to respond quickly during user interaction. These features reveal that the model offers strong performance in terms of user experience. [1]

The Gemini-2.5-flash model can be considered as an alternative model in user studies. Its 100% success rate in navigation tasks, matching baseline performance, makes it a suitable option for comparative analysis. However, its average latency of 2.25 seconds creates a disadvantage compared to GPT-40 in terms of real-time interactions. Nevertheless, the Gemini-2.5-flash model also demonstrated high performance in question-and-answer tasks. It achieved 100% success in all QA metrics and exhibited fast performance with an average response time of 0.77 seconds. The use of such high-performance models allows user evaluations to reveal not only the model's performance but also the overall performance of the system under optimal conditions.

Benchmark results show that the models' performance in navigation tasks is quite similar. Success rates range from 94.2% to 100.0%. Furthermore, all models were found to provide safe navigation with very low collision rates. The system's modular architecture allows for easy integration of different LLM models. This means that when changing models, no changes need to be made to the system's core operation. Thus, the system can work compatibly with a wide variety of LLM models. Users or developers can select models based on criteria such as cost, speed, or performance according to the application's requirements, and continue to benefit from the system's basic navigation and interaction features.

6.2 Discussion of User Evaluation Results

A user evaluation study with 30 participants used the System Usability Scale (SUS) and the Usefulness, Satisfaction, and Ease of Use (USE) questionnaire to examine the AI-assisted navigation system. The findings show that users had mostly positive views on usability and overall experience.

6.2.1. System Usability Scale (SUS) Results Discussion

An analysis of SUS (System Usability Scale) scores by category provides more detailed information about user experience. 60% of participants (18 people) rated the system in the "Excellent" category. This category belongs to SUS scores 80 and above, indicating that most users found the system highly usable. This strong result shows that the AI-assisted navigation system meets the basic usability requirements necessary for emergency evacuation scenarios. 23.3% of participants (7 people) rated the system as the "Good" category. This category belongs to SUS scores between 68 and 79, indicating that users generally found the system usable. However, this result also suggests that there are some areas for improvement. 16.7% of the participants (5 people) rated the system in the

[1] Kindly check the DISCLAIMER (p. 91)

"Average" category (SUS scores between 51 and 67). This indicates that most users had a positive experience, however some users may have encountered certain difficulties during the process. No participant rated the system in the "Bad" category (SUS < 51). This result indicates that all users found the system as acceptably usable. The standard deviation of the SUS scores was calculated as 10.99. This value indicates a moderate difference among user evaluations. Furthermore, the range of scores from 52.50 to 100.0 shows that user experiences are quite diverse and that considering individual differences in system design is important. Analysing the questions individually in the SUS survey also provides important clues about user perceptions. The survey uses both positive and negative statements, and participants' responses is consistent with this structure; this indicates that participants carefully considered the questions. The highest average score was obtained for question 7 with 4.67. The statement in this question, "I think most people can learn this game very quickly," shows that users considered the system as easy to learn. The lowest average scores were obtained for questions 2 (1.67) and 8 (2.00). These questions contain negative statements regarding the complexity and difficulty of use of the system. The low scores indicate that users do not find the system complex or difficult. Furthermore, since these questions are reverse scored in the SUS calculation, the low scores actually represent a positive outcome. This contributed to an increase in the system's overall usability score. [1]

6.2.2. USE Scale Results Discussion

The USE survey evaluated four dimensions of user experience: Usefulness, Ease of Use, Ease of Learning, and Satisfaction. Users consistently gave positive evaluations across all dimensions. Average scores ranged from 6.48 to 6.64 out of 7. These results indicate that the system is perceived very positively by users.

The Usefulness dimension was measured with 5 questions. The average score for this dimension was 6.48, and the median was 6.54. The standard deviation was 0.39. Scores ranged from 5.50 to 7.00. These high scores indicate that users find the AI-powered navigation system very useful for emergency evacuation scenarios. Users' consistent evaluation of the system as useful shows that it meets the need for effective guidance in emergency situations.

The Ease-of-Use dimension was evaluated with 9 questions. The average score was calculated as 6.56, and the median value as 6.68. The standard deviation was measured as 0.40. Scores ranged from 5.56 to 7.00. These results indicate that users found the system easy to use. This is particularly important for emergency evacuation systems that need to operate under stress. High scores suggest that features such as voice commands and AI-assisted navigation made navigation less mentally demanding.

The Ease of Learning dimension was measured with 4 questions. This dimension received the highest average score among all categories. The average score was calculated as 6.64, and the median value as 6.72. The standard deviation was measured as 0.39. Scores ranged

[1] Kindly check the DISCLAIMER (p. 91)

from 5.58 to 7.00. This result shows that users found the system easy to learn. This feature is particularly important for emergency evacuation systems, as people may have to use this system for the first time in a real emergency. High scores indicate that the system design helps users learn the system quickly.

The satisfaction dimension was evaluated with 6 questions. The average score was 6.53, and the median value was 6.58. The standard deviation was calculated as 0.41. Scores ranged from 5.50 to 7.00. This result shows that users are generally highly satisfied with the system. This satisfaction stems from both the system's efficient operation and the positive user experience.

When the overall USE score, representing the average of all four dimensions, is examined, the average value is 6.55, and the median value is 6.62. The standard deviation was measured as 0.39. Scores ranged from 5.53 to 7.00. These results show that the AI-powered navigation system successfully meets different aspects of user needs. The system offers a balanced performance between usefulness, ease of use, learnability, and satisfaction.

The standard deviation values for all USE dimensions ranged from 0.39 to 0.41. These values indicate low variability among user evaluations. This consistency shows that the system offers a similar experience for different users. This feature is particularly important for emergency evacuation systems that will be used by different user groups.

6.2.3. User Feedback Analysis Results

Participants highlighted several strengths of the AI-powered evacuation system. Many users found the AI guidance effective. For example, one participant stated, "The AI assistant was very effective at providing real time guidance during the emergency". Another participant said, "The AI commands were sufficient, and the AI assistant is aware of the current situation and position of mine" indicating that the AI provided appropriate guidance. The pathfinding algorithm has also received positive feedback from users. One user said, "The pathfinding algorithm works really well and finds the shortest routes quickly," indicating that the algorithm works quickly and efficiently. Another user commented, "It was quite impressive that the AI could recalculate the route when obstacles appeared" expressing their appreciation for the system's ability to dynamically update the route when obstacles arose.

Participants also frequently emphasized safety and efficiency. One participant stated, "I felt safe during the evacuation because the AI provided instructions very fast" indicating that the system gave people to a sense of security. Another participant said that "The system significantly reduced my evacuation time compared to the first scenario," indicating that the system significantly reduced evacuation time while comparing two evacuation scenarios of the experiment. Users valued the system's ability to adapt in real time. One participant positively evaluated the continuous updates to the routes, saying, "I

[1] Kindly check the DISCLAIMER (p. 91)

appreciated the in-time updates and adjustments to the evacuation route." Overall, participants considered the system a well-organized and helpful tool. [1]

On the other hand, the participants found things that could be improved. Some participants said they were worried about the quality of the images. One person said, “Game graphics could be better to make them look more real” and another said, “Graphics are good but could be more polished and detailed to be more attractive”. A third person said, “The visual quality of the building model needs to be improved”. However, they also said that they compared the system to other computer games instead of digital twins. Another problem was door interactions. One participant said, “Passing through doors is awkward sometimes and could be smoother” and another said, “Door passages felt a little jarring”. There were also comments about how the interface and visualisation looked. One person said, “The user interface could be more user-friendly” and another said, “A map could be added to make the building layout visualisation clearer”. Some people thought the route updates are too fast. “Sometimes the AI commands come too quickly and can be overwhelming”, one participant said. “The system sometimes gives redundant commands just for a second before updating,” another person said. Lastly, environmental clarity was brought up, with one person saying, “Some parts of the building do not have enough light or visual clarity”. [1]

Finally, participants provided suggestions for enhancing the system. Participants often recommended visual improvements. For example, one suggested enhancing graphics quality for better immersion and realism, while another wanted clearer exit signs and emergency indicators. Door mechanics were also mentioned. One participant asked for better door animations to make transitions feel more natural, and another suggested adding haptic feedback or visual cues when approaching doors instead of just opening a UI. Some focused-on user experience, with suggestions like adding more visual feedback when following AI commands to confirm actions or implementing voice recognition for hands-free command acknowledgment. Training and scenario variety came up as well. One participant wanted different building layouts for varied training, and another recommended adding more realistic emergency scenarios, such as smoke or fire effects. Accessibility and performance metrics were also noted. One participant suggested including accessibility features for users with different needs, and another wanted a performance metrics display to show evacuation efficiency. [1]

6.2.4. Correlation Analysis Between SUS and USE Scores Discussion

Correlation analysis showed strong and positive relationships between SUS scores and the dimensions of the USE scale. Pearson correlation coefficients ranged from 0.685 to 0.717, and all results were found as statistically significant ($p < 0.001$). The highest correlation is between the Usefulness dimension ($r = 0.717$). This indicates that SUS scores strongly reflect how useful users find the system. There is also a high correlation between the Ease of Learning dimension and SUS ($r = 0.709$), showing that SUS is effective in evaluating

[1] Kindly check the DISCLAIMER (p. 91)

how easy the system is to learn. Similarly, there is a strong correlation between Ease of Use and SUS (r = 0.701), showing that SUS scores significantly represent the aspect of ease of use. The correlation value for the Satisfaction dimension, while slightly lower than the others, also shows a strong relationship (r = 0.685). This result suggests that SUS also reflects user satisfaction but represents it in a more limited way compared to other dimensions. Overall, the results show that the SUS and USE scales measure different aspects of usability. The increase in USE scores as SUS scores increase in each USE dimension indicates that the two measurement tools reveal similar perceptions of usability. This suggests that using both methods together in the evaluation of the AI-powered navigation system could provide a more comprehensive analysis. Furthermore, small differences in correlation values indicate that SUS offers a broad assessment of overall usability, while the USE scale provides more detailed information about specific dimensions of user experience. When checking the individual dimensions, SUS is most closely related to Usefulness and Ease of Learning, which have the strongest correlations (r = 0.717 and r = 0.709). This means that SUS shows how useful and how easy it is to learn the system. The slightly lower correlation with Satisfaction (r = 0.685) suggests that SUS may not measure satisfaction as directly as it could. This might be because SUS is more concerned with how well and how quickly things work, while the Satisfaction dimension in USE includes more emotional factors. [1]

6.2.5. Time Measurement Results Discussion

There is a significant difference in task completion times between Scenario A (2 minutes 7.80 seconds) and Scenario B (58.25 seconds). The total difference between the two scenarios was calculated as 69.55 seconds. One reason this finding is significant is that both scenarios were designed at the same difficulty level. Furthermore, a counterbalanced design was applied in the experiment. This method helps control for sequence effects, making it possible to lower the learning effects. The shorter completion times observed in the AI-assisted scenario shows that users can use the system faster with guidance. This is very important, especially in real-world emergency evacuation scenarios.

6.3 Practical Implications

The outcomes highlight some important implications related with related with AI-assisted evacuation. The evaluation results show that, Gemini-2.5-flash and GPT-4o models have performed well which makes them proper to be used in real world emergency evacuation situations. The higher benchmark results related assigned with “safety”, positive user evaluation scores and feedback results show that, AI integration should be implemented to such scenarios.

User feedback show that, the system can be considered as easy to learn. This is an important outcome, since people can be used such systems when any emergency occurs even though they are using the framework for the first time. Higher easy to learn scores

[1] Kindly check the DISCLAIMER (p. 91)

indicates that, including AI assistance and voice interactions makes the process easier, which is an important advantage.

While designing such systems, including only technical evaluation might not be the only consideration. Including user evaluation should be the case since users can observe unpredictable or underestimated situations while using the system. For example, although developers think that the door opening animations are coded successfully, people might have some struggles while using that interactions which shows the design should be updated. For this study, although most users rate the system good to excellent, some users have encountered with some challenges which show the improvement opportunities of the proposed framework. Additionally, some users with disabilities also should be considered in the design phase to improve accessibility features.

During the design phase of such LLM integration to digital twins, there should be a balance between different factors. Especially, there should be a trade-off between safety and response time since both aspects are important for the evacuation cases. While selecting the LLM, all safety, speed, cost and user experience dimensions should be evaluated.

[1] Kindly check the DISCLAIMER (p. 91)

CHAPTER 7

CONCLUSION

This thesis includes implementation and evaluation of an computational approach which combines AI-assisted BIM analytics and digital twin development. This approach is investigated with an evacuation simulation case study. The proposed system includes BIM data and game engine integration. By this integration, the main purpose is creating a digital twin of the BIM model which also includes AI features that can be used in evacuation scenarios. In this chapter, key contributions and the result of the study are summarized as well as practical considerations and possible outcomes. Additionally, limitations of the study and the future work that might be followed by researchers are explained.

7.1 Conclusion

The results show that integration of all BIM, digital twin and LLM technologies are possible and applicable. Especially, this kind of integration can be used for evacuation scenarios. By this integration, an AI-assisted navigation system is developed. The system contains the data retrieved from the BIM models within the game engine environment. Revit is the selected BIM tool as well as Unity is the selected game engine since both are most used software in the industry. Revit exports are not only included to Unity environment but also used in the analysis layer. The architectural data is transformed into occupancy grid in this layer to be able to use A* algorithm for pathfinding applications. The study uses LLMs together with pathfinding algorithm. This is hybrid approach. While A* algorithm enables safe and reliable pathfinding results, LLMs take the responsibility of decision making and spatial reasoning. The LLMs also give answers to user queries by using speech-to-text technology. Players ask their questions while keeping moving within the environment which that provided hands-free feature. This feature makes the digital twins more accessible and user friendly which are important aspects of the evacuation scenarios.

The Benchmark tests applied on LLMs performance on pathfinding and navigation have shown some important results of those models on evacuation in emergency situations. Results show that, LLMs are quite successful o this matter. Test results show that, all the selected LLMs for this study are quite successful while giving higher success rates with lower collision rates as well as the outputs routes are quite close to optimal routes. Note that, success rates and collisions rates are related with “safety” concerns as well as optimal routes are related with “cost” concerns. As a results, selected models provide safe, efficient and cost-efficient outcomes. Additionally, LLMs also performed successfully in

answering location and navigation related questions during the evacuation scenarios. Those models provide necessary and valuable information to users which helps to reduce evacuation time during the process. A user evaluation study was conducted. In this study, 30 participants completed two different scenarios. After completion of the process, participants filled out the System Usability Scale (SUS) and the Usefulness, Satisfaction, and Ease of Use (USE) questionnaires. Both scores can be considered sufficient. The SUS results show that most users found the system highly usable. The USE questionnaire results also show that, peoples find the system usable, easy to learn, easy to use and satisfied from the system based on their evacuation navigation expectations.

The study evaluates technical performance and user expectations together to be able to analyse the relationship between those aspects. Higher user evaluation results indicate that high performance on navigation affects users positively. Higher success and lower collision rates make people to feel safe and secure during the process. Additionally, sufficient question-answering performance has a positive effect on user feedback. This combined evaluation methodology offers a successful framework for deployment as well. Especially, high "easy-to-learn" feedback scores highlights this system's potential for the usage of evacuation scenarios while people might need to use this application first time.

7.2 Limitations and Future Work

While this research has made significant contributions, it has some limitations. User evaluations were conducted in a controlled laboratory environment. This condition may not fully reflect the stress, time pressure, and challenging conditions encountered in real-world emergencies. Benchmark tests were conducted using only a single building plan. Therefore, the results may not be directly applied to other building types. Dynamic factors such as fire, smoke, blocked roads, or changing building conditions were not considered in the evaluation process. Therefore, the system's performance in some situations, such as when evacuation routes change or new hazards emerge, is not yet known. The study focused on individual pathfinding. However, real evacuation scenarios may involve coordination challenges because of crowd dynamics. The research excludes certain factors, such as long-term use, multilingual support, and accessibility for users with special needs. Building data processing was designed for specific Revit export formats. LLM integration was also tested using specific settings.

Future research might involve testing the system under more real-world conditions and considering various building types. Additionally, the system should be evaluated by multiple users. Long term usability and learning curve might be investigated. Researchers should evaluate the system architecture's robustness under challenging conditions, whether the system supports different languages and cultures. The accessibility should also be examined. In addition, advanced AI features can also be included, such as personalized AI support and adaptive learning. It can be investigated how LLMs can make decisions autonomously as well as how they can be adopted for complicated evacuation procedures. It can also be examined that this system can be integrated with building

management systems and IoT devices. Furthermore, ethical and safety concerns should be expanded. How the system works in accordance with the emergency response tools can be another future research area. In addition, the practicality and cost-effectiveness of large-scale use of this system should also be investigated.

DISCLAIMER

Artificial intelligence tools are used in this thesis in a limited capacity for grammar correction, increasing the fluency of sentences and overall readability. The parts that have been corrected or rewritten using AI tools are indicated with a superscript footnote reference ([1]). AI usage is limited to some textual contexts of the following chapters: Introduction, Literature Review, System Architecture and Components, User Evaluation, Results and Discussion. However, it should be highlighted that all AI tool outputs are revised and confirmed by the author so that there is no discrepancy between the thesis context or research outputs and the text. Please note that AI usage is limited only to language editing; the context of the research ideas, literature survey, analyses and experimental results, all algorithms, all comments and the corresponding conclusions belong to the author. All system design, source codes, experimental setup and results, figures and visualizations are separate from AI usage. The used tools can be listed as: Grammarly AI for grammar edits and ChatGPT (provided by OpenAI) for increasing fluency and overall readability of the text.

REFERENCES

[1] Lei, Z., Dong, Y., Li, W., Ding, R., Wang, Q. R., & Li, J. (2025). Harnessing Large Language Models for Disaster Management: A Survey. Findings of the Association for Computational Linguistics: ACL 2025.

[2] Chen, R., Wang, C., Sun, Y., Zhao, X., & Xu, S. (2025). From Perceptions to Decisions: Wildfire Evacuation Decision Prediction with Behavioral Theory-informed LLMs. arXiv preprint arXiv:2502.17701.

[3] El Fekih Zguir, A., Ofli, F., & Imran, M. (2025). RoadMind: Towards a Geospatial AI Expert for Disaster Response. arXiv preprint arXiv:2509.19354.

[4] Zaman, Ahmad Akib Uz & Abdelaty, Ahmed & Sobuz, Md. Habibur. (2024). Integration of BIM data and real-time game engine applications: Case studies in construction safety management. Journal of Information Technology in Construction. 29. 117-140. 10.36680/j.itcon.2024.007.

[5] Saddiq Ur Rehman, Inhan Kim, Kyung-Eun Hwang, Advancing BIM and game engine integration in the AEC industry: Innovations, challenges, and future directions, Journal of Computational Design and Engineering, Volume 12, Issue 4, April 2025, Pages 26–54.

[6] Alexander, Kris & McArthur, Jenn & Lachapelle, Geoffrey & El Mokhtari, Karim & Damm, Mark. (2023). Applying video game design to building digital twin creation. 10.35490/EC3.2023.209.

[7] Clausen, Christian Skafte Beck & Ma, Zheng & Jørgensen, Bo. (2022). Can we benefit from game engines to develop digital twins for planning the deployment of photovoltaics?. Energy Informatics. 5. 42. 10.1186/s42162-022-00222-7.

[8] Rundel, Samuel & Amicis, Raffaele. (2023). Leveraging digital twin and game-engine for traffic simulations and visualizations. Frontiers in Virtual Reality. 4. 10.3389/frvir.2023.1048753.

[9] Crampen, David & Yadav, Yogender & Ishar, Shofia & Zlatanova, Sisi & Tian, Wenwen. (2024). Development of 3D UDTs for Traffic Monitoring in Unreal 5 Game Engine. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLVIII-4-2024. 131-138. 10.5194/isprs-archives-XLVIII-4-2024-131-2024.

[10] Aksu, Koray & Demirel, Hande. (2024). Integration of Building Information Model into a Game Engine Platform for Indoor Accessibility Analyses. WSEAS TRANSACTIONS ON ENVIRONMENT AND DEVELOPMENT. 20. 391-396. 10.37394/232015.2024.20.36.

[11] Lin, Jia-Rui & Chen, Ke-Yin & Song, Sheng-Yu & Cai, Yun-Hong & Pan, Peng & Deng 邓逸川, Yichuan. (2025). Digital Twin of Buildings and Occupants for Emergency Evacuation: Framework, Technologies, Applications and Trends. Advanced Engineering Informatics. 66. 103419. 10.1016/j.aei.2025.103419.

[12] Han, T., Zhao, J., & Li, W. (2020). Smart-Guided Pedestrian Emergency Evacuation in Slender-Shape Infrastructure with Digital Twin Simulations. *Sustainability*, *12*(22), 9701. https://doi.org/10.3390/su12229701.

[13] von Danwitz, Max & Bonari, Jacopo & Franz, Philip & Kühn, Lisa & Mattuschka, Marco & Popp, Alexander. (2024). Contaminant Dispersion Simulation in a Digital Twin Framework for Critical Infrastructure Protection. 10.48550/arXiv.2409.01253.

[14] Cantrell, Walter & PETTY, Mikel & KNIGHT, Samanatha & SCHUELER, Whitney. (2018). Physics-based modeling of crowd evacuation in the Unity game engine. International Journal of Modeling, Simulation, and Scientific Computing. 09. 10.1142/S1793962318500290.

[15] Wahlqvist, Jonathan & Ronchi, Enrico & Gwynne, Steve & Kinateder, Max & Rein, Guillermo & Mitchell, Harry & Benichou, Noureddine & Ma, Chunyun & Kimball, Amanda & Kuligowski, Erica. (2021). The simulation of wildland-urban interface fire evacuation: The WUI-NITY platform. Safety Science. 136. 105145. 10.1016/j.ssci.2020.105145.

[16] Wang, Bin & Li, Haijiang & Rezgui, Yacine. (2013). Intelligent building emergency management using building information modelling and game engine. ICIC Express Letters. 7. 1017-1023.

[17] Yu, Dazhou & Bao, Riyang & Mai, Gengchen & Zhao, Liang. (2025). Spatial-RAG: Spatial Retrieval Augmented Generation for Real-World Spatial Reasoning Questions. 10.48550/arXiv.2502.18470.

[18] Zhang, Yifan & Li, Jingxuan & Wang, Zhiyun & He, Zhengting & Guan, Qingfeng & Lin, Jianfeng & Yu, Wenhao. (2024). Geospatial large language model trained with a simulated environment for generating tool-use chains autonomously. International Journal of Applied Earth Observation and Geoinformation. 136. 10.1016/j.jag.2024.104312.

[19] Cui, Lei & Dong, Li & Mao, Shaoguang & Wei, Furu & Wu, Wenshan & Xia, Yan & Zhang, Yadong. (2024). Mind's Eye of LLMs: Visualization-of-Thought Elicits Spatial Reasoning in Large Language Models. 90277-90317. 10.52202/079017-2866.

[20] Milin, V., Stanivuk, T., Skoko, I., & Bulić, T. (2025). Dijkstra and A* Algorithms for Algorithmic Optimization of Maritime Routes and Logistics of Offshore Wind Farms. *Journal of Marine Science and Engineering*, *13*(10), 1863. https://doi.org/10.3390/jmse13101863

[21] Lu, Runjie. (2025). Heuristic Pathfinding Algorithms in Video Games Indepth Analysis and Performance. Science and Technology of Engineering, Chemistry and Environmental Protection. 1. 10.61173/3rvm6624.

[22] Martinez-Gil, Francisco & Lozano, Miguel & Fernández, Fernando. (2015). Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. Autonomous Agents and Multi-Agent Systems. 29. 98-130. 10.1007/s10458-014-9252-6.

[23] Kwak, Jaeyoung & Lees, Michael & Cai, Wentong & Ong, Marcus. (2020). Modeling Helping Behavior in Emergency Evacuations Using Volunteer's Dilemma Game. 513-523. 10.1007/978-3-030-50371-0_38.

[24] Templeton, Anne & Xie, Hui & Gwynne, Steve & Hunt, Aoife & Thompson, Peter & Köster, Gerta. (2023). Agent-based models of social behaviour and communication in evacuations: A systematic review Author names and affiliations.

[25] Cimellaro, G. & Mahin, Stephen & Domaneschi, Marco. (2018). Integrating a Human Behavior Model within an Agent-Based Approach for Blasting Evacuation. Computer-Aided Civil and Infrastructure Engineering. 34. 3-20. 10.1111/mice.12364.

[26] Alhazmi, Turki & Azzedin, Farag & Hammoudeh, Mohammad. (2025). MQTT Based Data Distribution Framework for Digital Twin Networks. 1008-1013. 10.1145/3726122.3726269.

[27] Boje, Calin & Kubicki, Sylvain & Rezgui, Yacine & Zarli, Alain. (2021). A Digital Twin factory for construction. 10.1201/9781003191476-41.

[28] Zhang, Y. (2024). Edge Computing for Digital Twin. In: Digital Twin. Simula SpringerBriefs on Computing, vol 16. Springer, Cham. https://doi.org/10.1007/978-3-031-51819-5_4

[29] Kuhlmann, Richard & Wolfram, Jakob & Sun, Boyang & Xing, Jiaxu & Scaramuzza, Davide & Pollefeys, Marc & Cadena, Cesar. (2025). Sight Over Site: Perception-Aware Reinforcement Learning for Efficient Robotic Inspection. 10.48550/arXiv.2509.17877.

[30] Seong, Haebin & Kim, Sungmin & Kim, Minchan & Cho, Yongjun & Joe, Myunchul & Choi, Suhwan & Jung, Jaeyoon & Youn, Jiyong & Kim, Yoonshik & Seong, Samwoo & Yubeen, Park & Yu, Youngjae & Lee, Yunsung. (2025). CostNav: A Navigation Benchmark for Cost-Aware Evaluation of Embodied Agents. 10.48550/arXiv.2511.20216.

[31] Žagar, Dejan & Svetina, Matija & Kosir, Andrej & Dimc, Franc. (2020). Human Factor in Navigation: Overview of Cognitive Load Measurement during Simulated Navigational Tasks. Journal of Marine Science and Engineering. 8. 10.3390/jmse8100775.

[32] Radford, Alec & Kim, Jong & Xu, Tao & Brockman, Greg & McLeavey, Christine & Sutskever, Ilya. (2022). Robust Speech Recognition via Large-Scale Weak Supervision. 10.48550/arXiv.2212.04356.

[33] Baldwin, C. Y., & Clark, K. B. (2000). Design rules: The power of modularity. MIT press.

[34] Meszaros, G. (2007). xUnit test patterns: Refactoring test code. Addison-Wesley Professional.

[35] Newman, S. (2021). Building microservices: Designing fine-grained systems (2nd ed.). O'Reilly Media.

[36] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley Professional.

[37] T. Collins, J. J. Collins and D. Ryan, "Occupancy grid mapping: An empirical evaluation," 2007 Mediterranean Conference on Control & Automation, Athens, Greece, 2007, pp. 1-6, doi: 10.1109/MED.2007.4433772.

[38] Wang, X.; Li, G.; Bian, Z. Research on the A* Algorithm Based on Adaptive Weights and Heuristic Reward Values. World Electr. Veh. J. 2025, 16, 144. https://doi.or

[39] Pydantic. (2024). Pydantic: Data validation using Python type annotations. Pydantic Documentation. https://docs.pydantic.dev/

[40] Gillies, S. (2024). Shapely: Manipulation and analysis of geometric objects. Shapely Documentation. https://shapely.readthedocs.io/

[41] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(3), 90-95.

[42] W. Zhou, L. Li, M. Luo and W. Chou, "REST API Design Patterns for SDN Northbound API," 2014 28th International Conference on Advanced Information Networking and Applications Workshops, Victoria, BC, Canada, 2014, pp. 358-365, doi: 10.1109/WAINA.2014.153.

[43] Ramírez, S. (2024). FastAPI: Modern, fast, web framework for building APIs with Python. FastAPI Documentation. https://fastapi.tiangolo.com/

[44] Mozilla Developer Network. (2024). Cross-Origin Resource Sharing (CORS). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[45] Unity Technologies. (2024). Unity: Real-time 3D development platform. Unity Documentation. https://docs.unity3d.com/

[46] Wang, X., Li, G., & Bian, Z. (2025). Research on the A* Algorithm Based on Adaptive Weights and Heuristic Reward Values. World Electric Vehicle Journal, 16(3), 144. https://doi.org/10.3390/wevj16030144

[47] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., ... & Batra, D. (2019). Habitat: A platform for embodied AI research. Proceedings of the IEEE/CVF international conference on computer vision.

[48] Voorhees, E.M. (2004). Overview of the TREC 2003 Question Answering Track. Text Retrieval Conference.

[49] Kwiatkowski, Tom & Palomaki, Jennimaria & Redfield, Olivia & Collins, Michael & Parikh, Ankur & Alberti, Christopher & Epstein, Danielle & Polosukhin, Illia & Devlin, Jacob & Lee, Kenton & Toutanova, Kristina & Jones, Llion & Kelcey, Matthew & Chang, Ming-Wei & Dai, Andrew & Uszkoreit, Jakob & Le, Quoc & Petrov, Slav. (2019). Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics. 7. 453-466. 10.1162/tacl_a_00276.

[50] Rajpurkar, Pranav & Jia, Robin & Liang, Percy. (2018). Know What You Don't Know: Unanswerable Questions for SQuAD. 784-789. 10.18653/v1/P18-2124.

[51] Charness, G., Gneezy, U., & Kuhn, M. A. (2012). Experimental methods: Between-subject and within-subject design. Journal of Economic Behavior & Organization, 81(1), 1-8.

[52] Gravetter, F. J., & Forzano, L. A. B. (2018). Research methods for the behavioral sciences (6th ed.). Cengage Learning.

[53] Nielsen, J. (1993). Usability engineering. Academic Press.

[54] Virzi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? Human Factors, 34(4), 457-468.

[55] Thorndyke, P. W., & Hayes-Roth, B. (1982). Differences in spatial knowledge acquired from maps and navigation. Cognitive Psychology, 14(4), 560-589.

[56] Ishikawa, T., & Montello, D. R. (2006). Spatial knowledge acquisition from direct experience in the environment: Individual differences in the development of metric

knowledge and the integration of separately learned places. Cognitive Psychology, 52(2), 93-129.

[57] Ruddle, R. A., & Lessels, S. (2009). The benefits of using a walking interface to navigate virtual environments. ACM Transactions on Computer-Human Interaction, 16(1), 1-18.

[58] Riecke, B. E., Cunningham, D., M., & Bühlhoff, H. H. (2005). Spatial updating in virtual reality: The sufficiency of visual information. Psychological Research, 69(5-6), 298-313.

[59] Gwynne, S., Galea, E. R., Owen, M., Lawrence, P. J., & Filippidis, L. (1999). A review of the methodologies used in the computer simulation of evacuation from the built environment. Building and Environment, 34(6), 741-749.

[60] Ronchi, E., & Nilsson, D. (2013). Modelling total evacuation strategies for high-rise buildings. Building Simulation, 7(1), 73-87.

[61] Keppel, G., & Wickens, T. D. (2004). Design and analysis: A researcher's handbook (4th ed.). Pearson.

[62] Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). Experimental and quasi-experimental designs for generalized causal inference. Houghton Mifflin.

[63] Wagenmakers, E. J., Krypotos, A. M., Criss, A. H., & Iverson, G. (2012). On the interpretation of removable interactions: A survey of the field 33 years after Loftus. Memory & Cognition, 40(2), 145-160.

[64] Thorndyke, P. W., & Hayes-Roth, B. (1982). Differences in spatial knowledge acquired from maps and navigation. Cognitive Psychology, 14(4), 560-589.

[65] Montello, D. R. (1991). Spatial orientation and the angularity of urban routes: A field study. Environment and Behavior, 23(1), 47-69.

[66] Meilinger, T., Knauff, M., & Bülthoff, H. H. (2008). Working memory in wayfinding—A dual task experiment in a virtual city. Cognitive Science, 32(4), 755-770.

[67] Senn, S. (2002). Cross-over trials in clinical research (2nd ed.). John Wiley & Sons.

[68] Jones, B., & Kenward, M. G. (2014). Design and analysis of cross-over trials (3rd ed.). CRC Press.

[69] Fleiss, J. L. (1986). The design and analysis of clinical experiments. John Wiley & Sons.

[70] Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. International Journal of Human-Computer Studies, 64(2), 79-102.

[71] Tullis, T., & Albert, B. (2013). Measuring the user experience: Collecting, analyzing, and presenting usability metrics (2nd ed.). Morgan Kaufmann.

[72] Golledge, R. G. (1999). Human wayfinding and cognitive maps. In R. G. Golledge (Ed.), Wayfinding behavior: Cognitive mapping and other spatial processes (pp. 5-45). Johns Hopkins University Press.

[73] Montello, D. R. (2005). Navigation. In P. Shah & A. Miyake (Eds.), The Cambridge handbook of visuospatial thinking (pp. 257-294). Cambridge University Press.

[74] Wiener, J. M., Büchner, S. J., & Hölscher, C. (2009). Taxonomy of human wayfinding tasks: A knowledge-based approach. Spatial Cognition & Computation, 9(2), 152-165.

[75] Dalton, R. C. (2003). The secret is to follow your nose: Route path selection and angularity. Environment and Behavior, 35(1), 107-131.

[76] Hegarty, M., Richardson, A. E., Montello, D. R., Lovelace, K., & Subbiah, I. (2002). Development of a self-report measure of environmental spatial ability. Intelligence, 30(5), 425-447.

[77] Brooke, J. (1996). SUS: A quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), Usability evaluation in industry (pp. 189-194). Taylor & Francis.

[78] Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the System Usability Scale. International Journal of Human-Computer Interaction, 24(6), 574-594.

[79] Sauro, J. (2011). A practical guide to the System Usability Scale: Background, benchmarks & best practices. Measuring Usability LLC.

[80] Lewis, J. R., & Sauro, J. (2009). The factor structure of the System Usability Scale. In Proceedings of the Human Computer Interaction International Conference (pp. 94-103). Springer.

[81] Lund, A. M. (2001). Measuring usability with the USE questionnaire. Usability Interface, 8(2), 3-6.

APPENDICES

APPENDIX A

SYSTEM USABILITY SCALE

System Usability Scale (SUS) [77]

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the website today.

		Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1.	I think I would like to use this tool frequently.	□	□	□	□	□
2.	I found the tool unnecessarily complex.	□	□	□	□	□
3.	I thought the tool was easy to use.	□	□	□	□	□
4.	I think that I would need the support of a technical person to be able to use this system.	□	□	□	□	□
5.	I found the various functions in this tool were well integrated.	□	□	□	□	□
6.	I thought there was too much inconsistency in this tool.	□	□	□	□	□
7.	I would imagine that most people would learn to use this tool very quickly.	□	□	□	□	□
8.	I found the tool very cumbersome to use.	□	□	□	□	□
9.	I felt very confident using the tool.	□	□	□	□	□
10.	I needed to learn a lot of things before I could get going with this tool.	□	□	□	□	□

APPENDIX B

USE QUESTIONNARIE

USE Questionnaire [81]

- Try to respond to all the items.
- For items that are not applicable, use: **NA**

Usefulness

		Strongly disagree			**Neither**		**Strongly agree**	
	Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	It helps me be more effective.							
2	It helps me be more productive.							
3	It is useful.							
4	It gives me more control over the activities in my life.							
5	It makes the things I want to accomplish easier to get done							

Ease of Use

		Strongly disagree			**Neither**		**Strongly agree**	
	Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**

1	It is easy to use.							
2	It is simple to use.							
3	It is user friendly.							
4	It requires the fewest steps possible to accomplish what I want to do with it.							
5	I can use it without written instructions.							
6	I don't notice any inconsistencies as I use it.							
7	Both occasional and regular users would like it.							
8	I can recover from mistakes quickly and easily.							
9	I can use it successfully every time.							

Ease of Learning

		Strongly disagree			**Neither**		**Strongly agree**	
	Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	I learned to use it quickly.							

2	I easily remember how to use it.							
3	It is easy to learn to use it.							
4	I quickly became skillful with it.							

Satisfaction

		Strongly disagree			**Neither**		**Strongly agree**	
	Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	I am satisfied with it.							
2	I would recommend it to a friend.							
3	It is fun to use.							
4	It works the way I want it to work.							
5	I feel I need to have it.							
6	It is pleasant to use.							

APPENDIX C

PROMPTS

A. Navigation Prompts

1. Segment Prompt

```
You are controlling a person walking in a 2D floor plan.
Goal: reach the EXIT without walking through walls/obstacles.
Start (m): ({start.x:.3f}, {start.y:.3f})
Exit  (m): ({goal.x:.3f}, {goal.y:.3f})
I will provide you with the exact, precomputed movement segments you must execute.
Segments (in order):
{seg_lines}
Output ONLY commands, one per line, using this grammar:
{command_grammar}
Rules:
- Do not add explanations or extra text.
- For each segment line, output exactly TWO lines:
  - the turn command
  - the walk command
- Keep the same numbers (do not change them).
```

2. Waypoint Prompt

```
You are controlling a person walking in a 2D floor plan.
Goal: reach the EXIT without walking through walls/obstacles.
You are given an ordered list of waypoints (x,y in meters) that is already collision-free.
Your task: convert these waypoints into simple human-like navigation commands.
Conventions:
- Coordinate system: x increases to the right (east), y increases up (north).
- Initial heading: 0 degrees, facing +x (east).
- “turn left A deg” rotates counter-clockwise by A degrees.
- “turn right A deg” rotates clockwise by A degrees.
- After turning, “walk D m” moves straight along the current heading by D meters.
Start (m): ({start.x:.3f}, {start.y:.3f})
Exit  (m): ({goal.x:.3f}, {goal.y:.3f})
Waypoints (m), in order:
{pts}
Output ONLY commands, one per line, using this grammar:
{command_grammar}
Rules:
- Do not add explanations.
- You MUST follow this deterministic algorithm for EVERY segment i -> i+1:
  1) dx = x[i+1]-x[i], dy = y[i+1]-y[i]
  2) target_heading_deg = atan2(dy, dx) in degrees (range -180..180)
  3) delta = normalize(target_heading_deg - current_heading_deg) into [-180, 180]
  4) If delta > 0: output “turn left <delta>” else output “turn right <abs(delta)>”
  5) distance = sqrt(dx^2 + dy^2) ; output “walk <distance> m”
  6) current_heading_deg = target_heading_deg
- Do NOT invent any extra turns/walks beyond those segment steps.
- Rounding: turns to nearest 1 degree, walk to 2 decimals.
- Output one command per line. No bullets, no markdown, no extra text.
```

B. Q A Prompts

1. Unity Server Question Prompt

You are a helpful and friendly evacuation assistant in a building evacuation game. Your role is to guide and motivate the player to safely exit the building.

Current information:

- {room_info}

- {distance_info}

- Player position: ({req.player_x:.2f}, {req.player_z:.2f})

Player's question: {req.question}

IMPORTANT RULES:

1. When the player asks about which room they are in, always respond with the room number (e.g., "You are in Room 1" or "Room Number 3"). There are 8 rooms total in the building, numbered 1-8.

2. If the question is NOT related to evacuation, navigation, building layout, or the current game situation, politely redirect the player. For example: "I'm here to help you evacuate safely. I can tell you which room you're in, how far you are from the exit, or help you find the exit door. Let's focus on getting you out safely!" Be friendly and encouraging, not dismissive.

3. For evacuation-related questions, provide helpful, natural, and motivating answers. Encourage the player to focus on reaching the exit but keep it brief (1-2 sentences). Be friendly, supportive, and use encouraging language like "Keep moving!", "You're doing great!", or "Focus on the exit!"

Please provide a helpful, concise, and encouraging answer in English.

2. QA Benchmark Prompt

You are a navigation assistant helping people evacuate a building. Answer questions based on the provided context.

Context:
{context}

Question: {question}

Instructions:
- Answer based ONLY on the provided context
- If the question cannot be answered from the context, respond with “I don’t know” or “I cannot answer that based on the available information”
- Be concise and helpful
- Use metric units (meters)
- If asked about room, provide the room name/number if available

Answer:

APPENDIX D

ETHICAL APPROVAL

UYGULAMALI ETİK ARAŞTIRMA MERKEZİ
APPLIED ETHICS RESEARCH CENTER

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 22 91
F: +90 312 210 79 59
uygulamietik@metu.edu.tr
www.uygam.metu.edu.tr

Konu : Değerlendirme Sonucu 19.12.2025

Gönderen : ODTÜ İnsan Araştırmaları Etik Kurulu

İlgi : İnsan Araştırmaları Etik Başvurunuz

Sayın Doç. Dr. Elif Sürer,
Sayın Nilsu Şahin,

"Yapay Zekâ Destekli BIM Analitiği ve Dijital İkiz Geliştirmeye Yönelik Entegre Hesaplamalı Yaklaşım /An Integrated Computational Approach for AI-Assisted BIM Analytics and Digital Twin Development " başlıklı araştırmanız ODTÜ İnsan Araştırmaları Etik Kurulu tarafından uygun görülerek 0559-ODTÜİAEK-2025 protokol numarası ile onaylanmıştır.

Bilgilerinize sunarım.

APPENDIX E

DEMO VIDEO

The video of the proposed digital twin can be reached from the following link:

https://drive.google.com/file/d/1kD9evMJROvHM0cw8ViBLL45zJXRmCRI3/view?usp=drive_link