TED UNIVERSITY

CMPE 492

Senior Design Project II

"TEDU GuidAR"

Spatial Computing for Indoor Navigation

Final Report

Supervisor: Tolga Kurtuluş Çapın
Jury Members: Fırat Akba, Kasım Murat Karakaya
Course Coordinator: Gökçe Nur Yılmaz

Authors and ID:
Berk Belhan 43906121950
Alperen Karadağ 14317165222
Altuğ Berke Akman 15349016582
Ceren Kızılırmak 14125057252

Project Page URL: https://berkbelhan.github.io/ar-navigation-senior-project/

GitHub Page URL: https://github.com/BerkBelhan/GuideAR

Table of Contents

1. Introduction

1.1 Project Overview

Indoor environments such as university campuses, museums, and airports frequently create navigation challenges due to complex multi-floor layouts and insufficient directional signage. TEDU GuidAR is an Augmented Reality (AR) and spatial computing application that addresses this problem by providing real-time, indoor navigation on the Meta Quest 3 device.

The system consists of virtual elements like directional arrows, path line, and informational labels that are rendered users' field of view. User can also see the real environment view using the Quest 3's passthrough cameras. Navigation data is stored in the cloud, reducing space related issues.

1.2 Scope and Objectives

The primary scope of GuidAR is the TED University campus building, specifically covering selected campus buildings where spatial anchors have been placed.

The main objectives are:

- Provide accurate, real-time user localization within the mapped indoor environment.
- Generate and display optimal navigation paths between any two Points of Interest (POIs) using the A* pathfinding algorithm.
- Having a dynamic rerouting when the user deviates from the planned path.
- Render AR navigation cues that correctly occlude behind physical obstacles for a convincing mixed-reality experience.
- Deliver a usable and intuitive interface operable entirely through hand-tracking gestures.
- Adding a language support for foreign newcomers.
- Placing interactive videos around the map to give a more special experience for the end user.

1.3 Final Status of the Project

As for the final status of GuidAR, has been successfully implemented and deployed on the Meta Quest 3 headset. The core navigation pipeline from destination selection through pathfinding to AR visualization is fully operational. Formal test cases were done, and scenarios are filmed, as a video, ready to be presented.

The NavMesh based mapping of the selected campus areas are complete and stable. Spatial anchor persistence allows the system to relocalize correctly on subsequent launches without requiring a new scan. HUD correctly displays distance and line instructions in real time.

2. Background Research

2.1 Use of Library and Internet Resources

Throughout the project, we have conducted researches in indoor navigation, Simultaneous Localization and Mapping (SLAM). We heavily relied on The Meta Horizon Platform SDK documentation, Meta XR All in One SDK, Multiset MetaQuest3 SDK. We referenced technical blogs such as Volpis (2025) for practical guidance on AR indoor navigation development. Lastly, IEEE citation standards (IEEE, 2009) were applied throughout the design reports.

2.2 Review of Similar Designs and Existing Solutions

Several related systems were reviewed during the design phase:

- GPS-to-AR Hybrid Systems: Alluhaidan et al. (2025) demonstrated that combining grid-based spatial representations with AR overlays significantly improves indoor wayfinding accuracy compared to GPS-only approaches.
- SLAM-Based University Navigation: Sukhareva, Tomchinskaya, and Serov (2021) proposed a SLAM-based indoor navigation system for university buildings, highlighting the importance of pre-mapped anchor points for reliable relocalization. This principle directly applied in GuidAR's SpatialAnchor design.
- Campus AR Navigation: Lokhande et al. (2025) developed a campus navigation system combining AR and VR, providing insights into effective POI metadata structures and user interface conventions for educational environments.

2.3 Component Information and Specifications

The following key hardware and software components were specified and integrated:

- Meta Quest 3: Primary runtime device. Equipped with colour passthrough cameras, depth sensors, IMU, barometer, and hand-tracking hardware. Runs in an Android-based operating system.
- Meta Quest 3 Depth API: Used by the Occlusion Engine to generate depth masks, enabling virtual arrows to be correctly occluded by physical objects.
- Unity Engine: Cross-platform game engine used as the primary development environment for scene management, rendering, and scripting.
- Android SDK: Required for building and sideloading the application onto the headset.

2.4 Basic Engineering Principles

GuidAR's has several foundational engineering principles:

- Model-View-Controller (MVC) Architecture: The system is cleanly separated into a Model layer (navigation graph and POI data), a View layer (AR rendering and UI), and a Controller layer (sensor fusion, pathfinding, and input handling).
- Navigation Mesh (NavMesh): The traversable floor area is discretised into convex polygons baked from the Quest 3's Scene Mesh. Agent constraints (shoulder width, step height) prevent paths from passing through furniture or too close to walls.
- Cloud Data Storage: All navigation data is stored and managed in the cloud, enabling real-time access, synchronization across devices, and eliminating reliance on local device storage.

3. Final Architecture and System Design

3.1 High-Level System Architecture

The design of TEDU GuidAR is based on the Model–View–Controller (MVC) architectural pattern to ensure modularity, maintainability, and scalability throughout the development process. The architecture separates the system into three primary layers: the Model layer, responsible for data management and core application logic; the View layer, responsible for augmented reality visualization and user interface rendering; and the Controller layer, responsible for user interaction handling and coordination between system components. This separation of concerns enables the integration of AR-specific functionalities such as spatial localization, real-time pathfinding, gesture-based interaction, and headset-based rendering while maintaining a structured and extensible software design.

3.2 Detailed Design Description

The central component of the system is the NavigationController, which coordinates the overall navigation workflow by managing destination selection, storing the active Point of Interest (POI), requesting routes from the PathFinder, and updating the HUDView with navigation information such as upcoming destinations, distance, and directional guidance. User interactions are processed through the InputHandler, which translates Meta Quest 3 hand-tracking gestures into navigation commands and forwards them to the navigation controller.

Route generation is handled by the PathFindercontroller using the A* pathfinding algorithm on the building graph structure, returning an ordered list of navigation nodes representing the optimal route. To ensure robustness during real-world navigation, the ReroutingService continuously monitors the user's position relative to the generated route and automatically recalculates the path whenever the user deviates beyond an acceptable threshold. Spatial alignment between the virtual environment and the physical indoor space is managed by the MapLocalizationManager, which synchronizes the user's real-world position with the digital map through transformation data stored within the LocalSpatialAnchor.

The system's environment preparation process is handled by the MapMeshDownloader, which is responsible for downloading navigation mesh files and map datasets stored on a cloud server prior to application usage as a one-time setup operation. Storing navigation meshes remotely enables new indoor environments and updated map data to be distributed without requiring the entire application to be rebuilt or reinstalled. Once imported into the Unity project and attached to the designated MapSpace object, these meshes are utilized by the MapLocalizationManager and LocalSpatialAnchor systems to align the user's real-world position with the corresponding virtual indoor environment during the localization process.

Navigation destinations are represented through the Point Of Interest (POI) model, which extends graph nodes with searchable metadata including room numbers, staff information, and office hours to provide contextual information alongside navigation functionality. On the visualization side, the HUDView delivers persistent heads-up navigation feedback within the AR environment, while the OcclusionEngine utilizes the Meta Quest Depth API to generate depth masking effects that allow real-world objects to properly occlude virtual navigation elements, improving immersion and depth perception.

The application's interface management is handled through the MainMenuManagement and NavigationManagement components, which connect the main menu, settings menu, and credits interface using Unity UI Toolkit documents while integrating Quest 3 headset input controls. Overall, the architecture was designed to provide a clear separation between application logic, visualization systems, and interaction management, allowing individual subsystems such as localization, rendering, pathfinding, and interface control to be developed, maintained, and extended independently.

3.3 Component-Level Design

Package	Class	Responsibility
Model	**MapLocalization Manager**	Map Localization Manager handles the process of aligning the user's real-world position with the digital map used by the application
Model	**VideoTrigger**	Video Trigger used automatically to control video playback based on the feedback of Box Collider.
Model	**MapMeshDownloader**	Map Mesh Downloader downloads and imports the 3D mesh data of a selected map set into the Unity scene. The script downloads the corresponding .glb mesh file, saves it inside the project files, and attaches the imported mesh under the assigned MapSpace object
Model	**MainMenu Management**	Main Menu Management controls the main menu interface of the application. It connects the main menu, settings menu, and credits menu through Unity UI Toolkit documents
Model	**Navigation Management**	Navigation Management manages the settings menu and basic application controls for the Quest 3 navigation application. It connects the UI Toolkit settings interface with headset input.
Model	**NavigationController**	Navigation Controller manages all of the flow; from selecting destination list, storing the POI, calling the PathFinder controller and managing HUDView operations.
Model	**Point Of Interest (POI)**	Node extension storing searchable room metadata (number, staff, office hours).

Model	**LocalSpatialAnchor**	Transformation matrix for aligning virtual content to physical room geometry.
View	**OcclusionEngine**	Depth-API depth mask enabling virtual elements to be hidden by physical objects.
View	**HUDView**	Persistent heads-up display: next POI name, distance, and turn instructions.
Controller	**PathFinder**	A* pathfinding on BuildingGraph; returns ordered Node list for the active route.
Controller	**ReroutingService**	Triggers path recalculation if user exceeds beyond destined path.
Controller	**InputHandler**	Translates Meta XR hand-tracking gestures into destination selection commands.

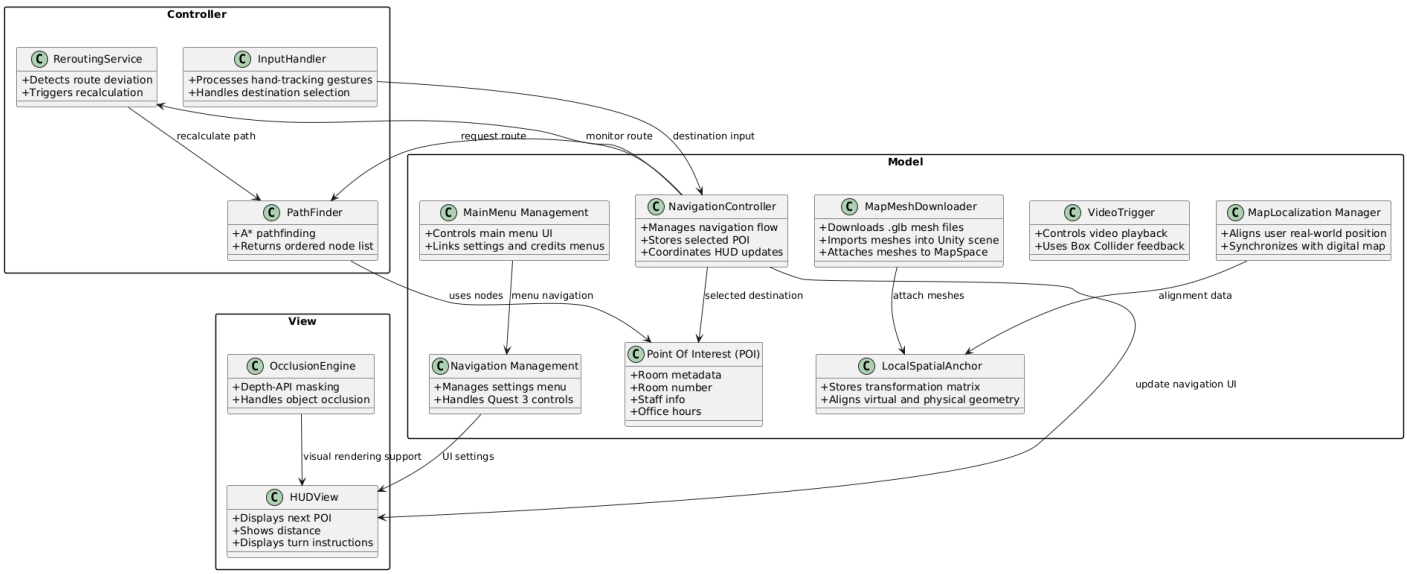

Figure 1. GuidAR System Architecture

Architectural Class Diagram

3.4 Runtime Application Components

3.4.1 Main Menu System

The main menu serves as the primary interaction hub of the TEDU GuidAR application. It allows users to start navigation sessions, select destinations, adjust settings such as audio volume and interface preferences, and access application information. The menu was designed specifically for mixed reality interaction on the Meta Quest 3, supporting intuitive controller-based input and maintaining readability within the AR environment.

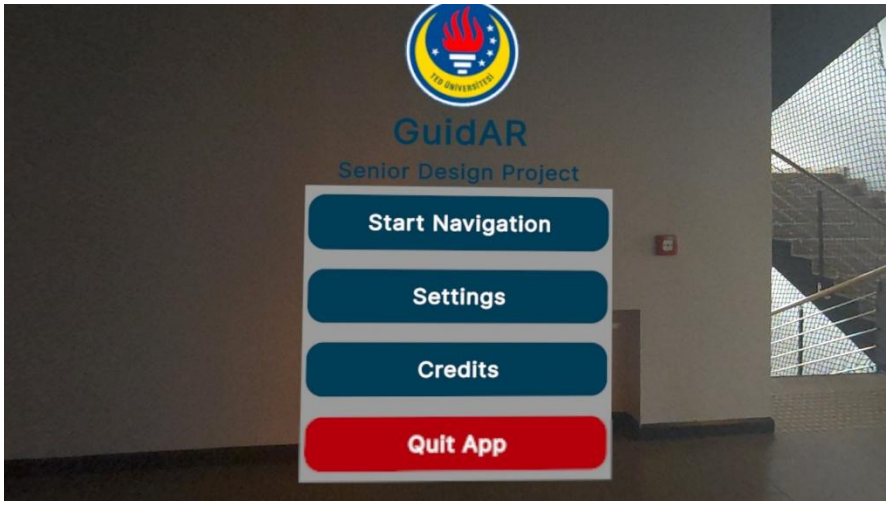

Figure 2. GuidAR Main Menu UI

3.4.2 Points of Interest (POIs)

Points of Interests represent important indoor locations such as classrooms, offices, laboratories, exits, and informational areas. Each POI is integrated into the navigation graph and contains metadata including location identifiers and category information. POIs allow users to select destinations easily while also enhancing navigation clarity by providing recognizable environmental reference points.

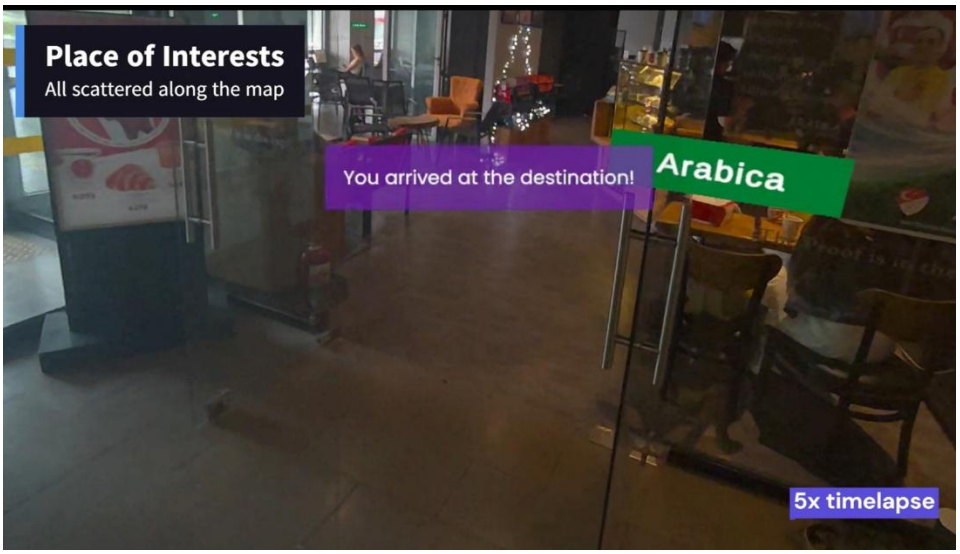

Figure 3. POI Destination Reached Popup Display

3.4.3 AR Navigation Arrows

The AR navigation arrow system provides real-time directional guidance by overlaying virtual arrows directly onto the user's physical surroundings. These arrows dynamically update according to the calculated route and user position, guiding the user through corridors, intersections, and floor transitions. The navigation cues were designed to remain visually clear while minimizing clutter within the mixed reality experience.

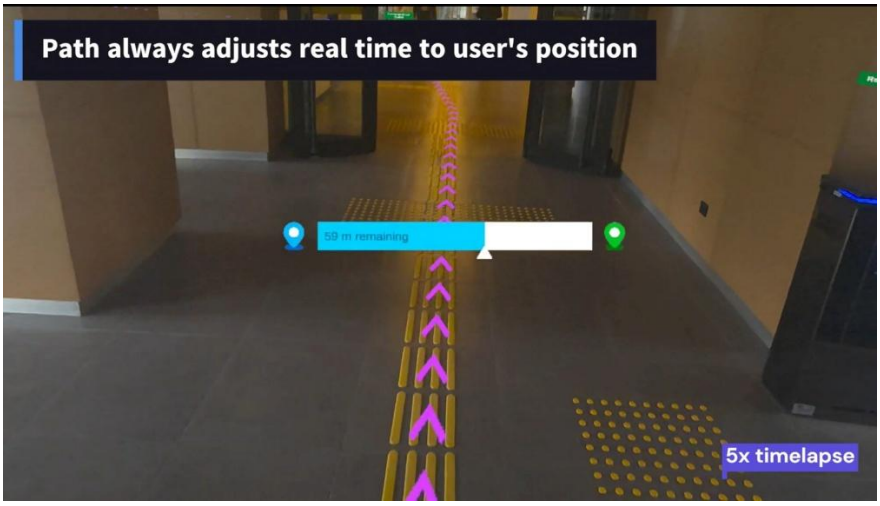

Figure 4. Navigation Arrows on Display

3.4.4 Informative Video Trigger System

The informative video trigger system activates contextual multimedia content when users approach designated locations within the environment. These videos provide additional information about specific areas, departments, or facilities and enhance user engagement during navigation. Trigger zones are spatially mapped within the indoor environment and automatically initiate playback without interrupting the navigation workflow.

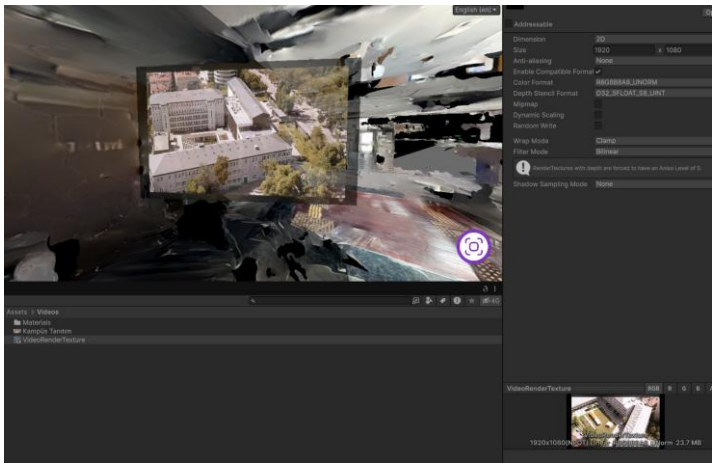

Figure 5. Campus Overview Video Object Placed on the Map

3.5 Final Implementation Details

The final implementation of the AR indoor navigation system was developed in Unity using C# and optimized for the Meta Quest 3 platform. The application integrates Meta XR SDK features including hand tracking, spatial anchoring, passthrough mixed reality, and Depth API-based occlusion to create an immersive indoor navigation experience. Prior to application usage, indoor environment meshes and map datasets are prepared and downloaded through the MapMeshDownloader component during the setup process, allowing the whole navigation mesh to be imported into the Unity project and attached to the designated MapSpace objects.

During operation, the MapLocalizationManager and LocalSpatialAnchor systems align the virtual environment map set with the user's physical surroundings to maintain accurate spatial positioning and stable AR content placement. Indoor routes are generated using an A* pathfinding implementation within the PathFinder controller, while the ReroutingService continuously monitors the user's movement and recalculates navigation paths whenever the user deviates significantly from the intended route. User interaction is handled through Meta Quest hand-tracking functionality, with the InputHandler translating gestures into navigation commands and destination selections.

Navigation guidance is displayed through the persistent HUDView, which presents directional instructions, remaining distance, and upcoming destination points directly within the augmented reality environment. Additionally, Unity UI Toolkit was used to implement the application's main menu, settings menu, and credits interface, enabling seamless headset-based interaction. The final implementation emphasizes modularity, maintainability, and responsive real-time operation while supporting scalable deployment across multiple indoor environments.

4. Tools and Technologies

4.1 Hardware Tools and Environments

The Meta Quest 3 was the primary hardware platform used in the project. Its built-in sensors and mixed-reality capabilities played an important role in supporting localization, interaction, and AR visualization.

- Colour Passthrough Cameras: The passthrough cameras provided a real-time view of the physical environment. This allowed virtual navigation elements, such as arrows, paths, and labels, to be displayed over the real-world scene and enabled a mixed-reality navigation experience.
- Depth Sensor: It was used together with the Meta Quest Depth API by the OcclusionEngine. This made it possible to create depth-based masks which allowed virtual navigation elements to be hidden behind physical objects when necessary.
- Inertial Measurement Unit (IMU): The IMU provides accelerometer and gyroscope data that supports the headset's motion tracking system. This tracking information helped the application follow the user's movement in the indoor environment.

Development was carried out on Android workstations running Unity Editor with the Meta XR All-in-One SDK package.

4.2 Software Frameworks and Libraries

- **Unity Engine:** Used for scene management, 3D rendering, NavMesh baking, scripting (C#), and build generation. Unity's NavMesh system provided the floor discretisation and A*-compatible pathfinding surface.
- **Meta Presence Platform SDK / MR Utility Kit:** Supplied Spatial Anchor creation and Scene Mesh generation for mixed-reality passthrough rendering.
- **Meta XR Interaction SDK:** Delivered hand-tracking input events, gesture recognizers, and interactable component abstractions consumed by the InputHandler class.
- **Meta Quest Depth API:** Provided per-frame depth estimates used by the OcclusionEngine to render the depth mask.
- **Android SDK (via Unity Build Support):** Required for compiling and deploying the application binary to the Quest 3 device.
- **MultiSet Quest SDK (Room Scanning):** Used during the environment-mapping phase to scan and align room geometry data, which was subsequently used to define spatial anchor placement.

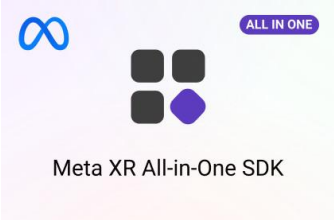

Figure 6. Meta XR SDK Logo

Figure 7. MultiSet Quest SDK Logo

4.3 New Technologies Acquired During the Project

Several technologies were new to the team prior to this project and required dedicated learning effort:

- **Mixed-Reality Passthrough Rendering:** The team learned how to configure passthrough-based mixed-reality rendering in Unity.
- **Depth API Occlusion:** Integrating depth-based occlusion was a new technique for the team. The Depth API was used to support realistic interaction for virtual navigation elements by allowing real-world objects to visually occlude AR content when necessary.
- **Cloud-Based Map and Mesh Management:**
  The team learned how to manage map and mesh data through cloud storage and import it into the Unity project. The MapMeshDownloader component was used to download the required mesh data and attach it to the related MapSpace object during the setup process.
- **NavMesh-Based Indoor Navigation:**
  The team learned how to prepare and use NavMesh-based navigation for indoor AR environments. This included working with imported map geometry, defining walkable areas, handling problematic regions in the mesh, and generating navigation paths suitable for indoor guidance.

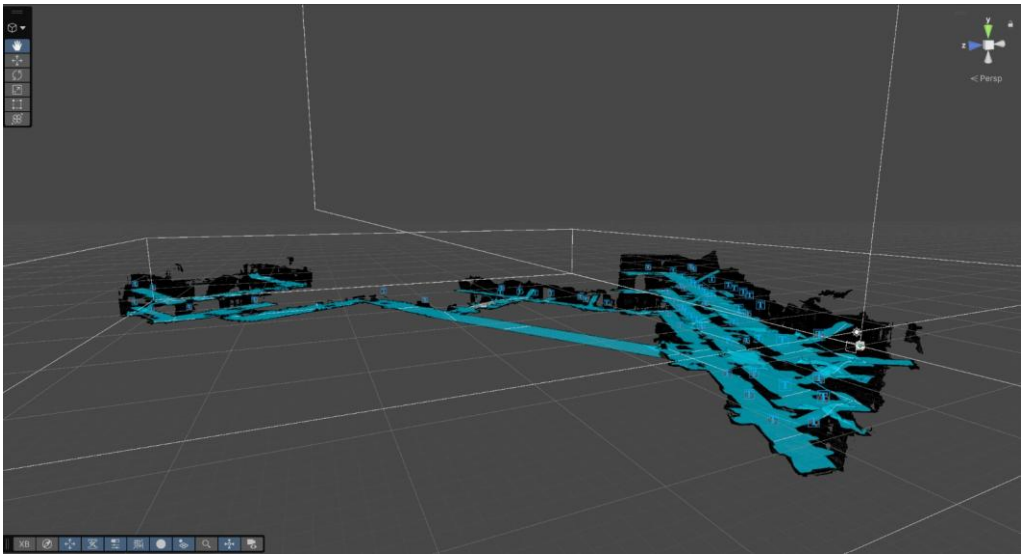

Figure 8. NavigationMesh of the Scanned TEDU Campus Area (A,B and K Blocks)

- **Spatial Localization and Relocalization:**

  Spatial localization and relocalization were important concepts learned during the project. The team worked on aligning the virtual map with the real indoor environment. This required understanding how spatial reference data and map positioning.

- **Interactive Informative Video Integration**

  Informative videos were integrated into selected locations across the indoor map to provide users with additional contextual information during navigation. These videos enhance the overall user experience by offering guidance, location-specific details, and interactive educational content directly within the AR environment.

5. Engineering Impact and Contemporary Issues

5.1 Impact of Engineering Solutions

GuidAR's engineering solutions create meaningful impact across technological, economic, environmental, and societal dimensions by combining real-time localization, AR visualization, and indoor navigation on consumer mixed-reality hardware.

5.1.1 Global Impact

Indoor navigation remains a major challenge in complex buildings such as universities, hospitals, airports, shopping malls, and large public facilities. GuidAR demonstrates that accurate indoor navigation can be achieved using consumer-grade mixed reality hardware without requiring expensive external infrastructure. The system utilizes Meta Quest 3 passthrough capabilities together with spatial localization and point-cloud-based mapping technologies to provide real-time AR navigation guidance. By combining VPS-based localization and custom navigation meshes, GuidAR offers a scalable indoor navigation approach that can be adapted to many different indoor environments worldwide. Additionally, the project introduces a customizable POI (Point of Interest) architecture where locations such as classrooms, lounges, laboratories, and offices are represented as individual objects with configurable metadata, categories, and visual properties. This modular structure allows future deployments to easily adapt the system to different institutions and building types.

5.1.2 Economic Impact

GuidAR follows a local-first architecture where navigation graphs, POI data, localization references, and spatial mapping information are stored directly on the device instead of relying on continuous cloud communication. This significantly reduces operational costs by eliminating the need for permanent server infrastructure, cloud rendering, or subscription-based localization services. Since localization and route calculation are performed locally using modified Meta SDK packages and MultiSet VPS components, institutions can deploy the system with minimal recurring maintenance expenses. Furthermore, the project demonstrates that advanced AR navigation systems can be implemented using commercially available devices instead of specialized industrial hardware, reducing deployment costs for universities and public facilities.

5.1.3 Environmental Impact

The system minimizes network usage by processing localization, route calculation, and rendering directly on the Meta Quest 3 device. Since continuous cloud synchronization is avoided, the overall communication and server energy consumption are reduced. GuidAR also prioritizes smaller,

controlled mapping environments instead of extremely large-scale spatial scans. This design decision reduces computational overhead during localization matching and NavMesh calculations. As a result, the system achieves more stable performance while lowering unnecessary computational resource consumption. In addition, the use of optimized navigation meshes and localized pathfinding reduces redundant rendering and processing operations during navigation sessions.

5.1.4 Societal Impact

GuidAR improves accessibility and usability inside complex indoor environments. Users who are unfamiliar with a building layout can navigate more efficiently through real-time AR guidance instead of relying on static maps or verbal instructions. The AR interface presents navigation cues directly within the user's physical field of view using the Quest 3 passthrough API, allowing users to maintain awareness of their surroundings while navigating. This reduces the distraction commonly associated with smartphone-based navigation systems. The system also supports intuitive destination selection through an interactive menu structure connected across scenes. Users can select destinations from categorized POI lists, after which the navigation engine automatically generates routes using the A* pathfinding algorithm and visualizes them through AR-rendered guidance paths. Additionally, automated event-trigger systems were implemented to improve user interaction. For example, specific video content can automatically open and close based on trigger conditions during navigation, enhancing contextual guidance and user engagement.

5.2 Discussion of Contemporary Issues Related to the Topic

5.2.1 Privacy and Sensor Data

GuidAR continuously processes environmental data captured through the Meta Quest 3 passthrough cameras and spatial sensors. Since localization depends on comparing real-time environmental features against previously generated point-cloud maps, large amounts of visual and spatial data are processed during operation. To address privacy concerns, all localization and navigation computations are executed locally on-device. Visual data is neither permanently stored nor transmitted to external servers. This local-first architecture reduces risks related to external data exposure and unauthorized environmental monitoring. However, privacy concerns surrounding spatial computing technologies remain an active contemporary issue. Regulations such as the EU AI Act and emerging biometric data policies increasingly focus on persistent environmental sensing systems and spatial data collection practices.

5.2.2 Platform Dependency and Vendor Lock-In

The current implementation heavily depends on Meta-specific technologies including the Meta Presence Platform, Passthrough API, Spatial Anchors, Scene Mesh, Depth API, and XR Interaction SDK. Additionally, the system integrates modified versions of Meta-provided packages together with MultiSet VPS and localization packages to improve localization accuracy and reduce positioning errors. While these optimizations improve system performance, they also increase dependency on vendor-specific ecosystems. As a result, significant redevelopment would be required if these APIs were deprecated or if the project were migrated to alternative XR hardware platforms. Although standards such as OpenXR aim to improve cross-platform compatibility, many advanced mixed-reality features still remain platform-dependent.

5.2.3 Localization Accuracy and Spatial Reliability

Indoor localization remains one of the most difficult technical challenges in AR navigation systems. Environmental changes, lighting variations, reflective surfaces, and incomplete spatial scans can negatively affect localization accuracy. To improve reliability, GuidAR combines multiple technologies including point-cloud comparison, localization packages, MultiSet VPS integration, custom navigation meshes and manually optimized planes in geometrically unclear regions. Additional modifications were also applied to third-party packages to reduce positioning errors and improve tracking stability. Despite these improvements, maintaining accurate localization in dynamic indoor environments continues to be a major contemporary challenge for spatial computing systems.

5.2.4 Digital Divide and Accessibility

Although mixed reality technologies are becoming more accessible, devices such as the Meta Quest 3 remain relatively expensive for widespread public adoption. This creates a digital divide where advanced spatial computing applications may only be accessible to institutions or users with sufficient financial resources. Large-scale adoption of AR navigation systems therefore remains dependent on future reductions in headset cost and improvements in lightweight wearable hardware.

5.2.5 Human Factors and User Safety

Extended use of head-mounted mixed reality devices may introduce ergonomic and cognitive concerns including eye fatigue, motion discomfort, reduced peripheral awareness, and user distraction. GuidAR attempts to minimize these issues through a simplified HUD structure that displays only contextually relevant navigation information. Navigation paths are visualized using minimal AR overlays instead of dense interface elements to reduce cognitive load during movement.

Nevertheless, long-term usability and ergonomic evaluation remain important future research areas for mixed reality navigation systems.

5.2.6 Data Maintenance and Scalability

The current implementation stores POIs, navigation meshes, destination lists, and localization data locally within the application environment. As buildings evolve over time, manually updating room information, navigation paths, and spatial references may become difficult at larger scales. Future versions of the system may require authenticated remote synchronization mechanisms or hybrid local-cloud update architectures to maintain accurate building information while still preserving the privacy-oriented local-first design philosophy.

6. Testing and Results

6.1 Overview of the Test Plan

The GuidAR test plan followed a layered testing methodology. The purpose of the testing process was to verify that the main navigation flow worked correctly, from destination selection to path generation, movement tracking, rerouting, menu interaction, and relocalization.

The testing process focused on the following main areas:

6.1.1 Unit Testing

Individual components such as destination selection, PathFinder, HUD updates, menu interaction, and rerouting logic were tested separately to verify that each function produced the expected output.

6.1.2 Integration Testing

The interaction between system components was tested. This included the connection between destination selection and NavigationController, PathFinder and route generation, user movement tracking and HUD updates, and the rerouting mechanism when the user moved away from the planned path.

6.1.3 System Testing

End-to-end navigation scenarios were executed on the Meta Quest 3 in real indoor environments. These tests covered the complete navigation process, including selecting a POI, generating a path, following the route, reaching the destination, displaying the destination-reached pop-up, and interacting with the menu.

6.1.4 Relocalization Testing

Relocalization was tested separately because spatial alignment is critical for AR navigation. The tests examined whether POIs and navigation elements remained correctly aligned after connection loss or tracking loss.

6.1.5 User Interaction Testing

The usability of the menu and navigation interface was tested through headset-based interaction. The goal was to confirm that users could start the navigation process and select destinations.

All tests were performed on the Meta Quest 3 headset in real indoor conditions to reflect the actual deployment environment.

6.2 Test Case Executions

The following table presents the eight test cases executed during the system testing phase, with their results.

ID	Test	Input	Expected	Result	Notes
TC-01	Destination Selection	User selects a POI from the “Destination List”.	System receives the chosen destination, NavigationController stores the target POI, and navigation initialises correctly.	**Pass**	Destination was reliably registered across all trials.
TC-02	Path Generation	User's current localized position and destination POI are passed to PathFinder.	A correct, collision-free path is computed using A* and shortest path is calculated.	**Pass**	A* produced optimal routes as desired.
TC-03	Destination reached pop-up	User reached the selected destination.	Destination reach pop-up is displayed	**Pass**	Destination reach pop-up is displayed
TC-04	User Movement Tracking	User walks along the calculated path.	SpatialController continuously updates the user's pose; HUDView displays decreasing distance to the next waypoint in real time.	**Pass**	The HUD distance counter updated at every relocalization frame.
TC-05	Automatic Rerouting	User deliberately deviates from the planned path beyond the configured threshold.	ReroutingService detects the deviation and triggers NavigationEngine to calculate and display a new route from the current position.	**Pass**	The application was able to reroute accurately.
TC-06	Menu UI Interaction	User interacts with the menu.	By interaction with the menu user can start the navigation process.	**Pass**	All UI interactions responded correctly.

TC-07	Relocalization	User tries to relocalize after lost connection.	Relocalization process works correctly. POI's are aligned as it was before.	**Fail**	Places with possible connectivity issues tends to fail the relocalization process. (Often results in misalignment of POI's)
TC-08	Relocalization	User tries to relocalize after lost position.	Relocalization process works correctly. POI's are aligned as it was before.	**Fail**	Relocalization is suboptimal under crowded areas. (Often results in misalignment of POI's)

6.3 Assessment of Test Results

- The test results show that the core navigation functions of GuidAR were successfully implemented and performed as expected in real indoor conditions. Out of eight executed test cases, six passed and two failed. The failed cases mainly related to relocalization reliability rather than the main navigation flow. The passing test cases demonstrate that the main navigation pipeline works correctly. Destination selection, path generation, destination arrival detection, movement tracking, automatic rerouting, and menu interaction all produced the expected results. The system was able to receive the selected POI, calculate a path using the A* algorithm, display navigation feedback through the HUD, update the remaining distance while the user moved, and reroute when the user deviated from the planned path. These results indicate that the essential navigation and interaction features are functional.

7. Conclusion

7.1 Summary of Project Outcomes

GuidAR successfully demonstrates that consumer-grade mixed reality hardware can support reliable and real-time indoor navigation without requiring continuous cloud infrastructure or external positioning systems. Throughout the project, a complete AR indoor navigation pipeline was designed and implemented on the Meta Quest 3 platform. The system combines spatial localization, point-cloud-based mapping, NavMesh-driven navigation, and real-time AR rendering to guide users inside complex indoor environments. Users can select destinations through a gesture-driven interface, after which the system generates navigation routes using the A* pathfinding algorithm and visualizes them directly within the physical environment through passthrough-based AR overlays. Navigation paths, directional cues, POIs, and destination indicators are spatially aligned with the real environment using localization and anchor-based positioning systems.

One of the major outcomes of the project was the development of a modular POI architecture. Locations such as classrooms, lounges, laboratories, and offices were implemented as customizable object types with configurable metadata and visual properties. This structure allows the system to scale more easily across different environments and use cases.

The project also integrated and modified multiple third-party packages including Meta SDK components, localization systems, and MultiSet VPS technologies in order to improve localization accuracy and reduce positioning errors. Additional engineering solutions such as custom navigation meshes, manually optimized planes for unclear areas, automatic event-trigger systems, and scene-based navigation flows were implemented to improve system reliability and usability.

The local-first system architecture proved highly effective during real-world testing. Since navigation, localization, and rendering operations are processed directly on-device, the system remained functional even in environments without stable internet connectivity while maintaining low-latency navigation updates.

Although automatic rerouting functionality (TC-05) was not fully stabilized at the final stage of development, the issue was isolated to asynchronous state management and does not represent a limitation of the overall system architecture. The implemented framework remains extensible and provides a strong foundation for future improvements and larger-scale deployments.

7.2 Lessons Learned

Several important lessons emerged during the project:

- Accuracy Over Coverage: Attempting to map large, complex environments introduced tracking instability. Focusing on smaller, controlled areas with clearly placed spatial anchors produced far more reliable relocalization. Future work should extend coverage incrementally rather than attempting large-scale mapping in a single pass.
- Event-Driven Concurrency: The TC-05 failure highlighted the dangers of polling-based state checks in a real-time system where sensor updates arrive asynchronously. Future components that react to continuous sensor streams should adopt an event-driven or reactive pattern from the outset.
- Platform API Maturity: Some Meta SDK APIs used in the project were in active development and underwent breaking changes during the project timeline, requiring reactive re-integration work. Teams building on rapidly evolving XR SDKs should pin API versions and maintain a regression test suite to detect breakage early.
- User Testing Early: Informal observations during beta testing revealed UX friction points—such as ambiguous arrow directions at corridor junctions—that were not captured by the formal test cases. Incorporating user observation sessions earlier in the development cycle would have allowed these issues to be addressed before the final release.
- Modular Design Pays Off: The MVC separation made it straightforward to replace or modify individual components (e.g., swapping the pathfinding heuristic or changing the AR renderer's arrow assets) without cascading changes across the codebase, validating the architectural decision.

8. References

[1] IBM, "UML - Basics," June 2003. [Online]. Available: http://www.ibm.com/developerworks/rational/library/769.html. [Accessed 17-Mar-2026].

[2] IEEE, "IEEE Citation Reference," September 2009. [Online]. Available: https://m.ieee.org/documents/ieeecitationref.pdf. [Accessed 17-Mar-2026].

[3] Volpis, "A complete guide to developing augmented reality indoor navigation applications," Oct. 23, 2025. [Online]. Available: https://volpis.com/blog/guide-to-developing-augmented-reality-indoor-navigation-applications/ [Accessed 16-Mar-2026].

[4] Google ARCore Documentation, Geospatial and Indoor Mapping APIs. [Accessed 17-Mar-2026].

[5] Apple ARKit Documentation, Visual-Inertial Odometry and Room Plan APIs. [Accessed 17-Mar-2026].

[6] E. Sukhareva, T. Tomchinskaya, and I. Serov, "SLAM-based indoor navigation in university buildings," 2021. [Online]. Available: https://ceur-ws.org/Vol-3027/paper63.pdf [Accessed 17-Mar-2026].

[7] Meta Horizon Platform SDK Documentation. Meta for Developers. [Online]. Available: https://developers.meta.com/horizon/documentation/unity/ps-platform-intro/ [Accessed 17-Mar-2026].

[8] D. K. Parab, P. P. Deshpande, and S. Khanvilkar, "Indoor navigation system using augmented reality," International Congress on Engineering and Computer Science, 2024. Semantic Scholar. [Accessed 17-Mar-2026].

[9] M. Vaqar, "Augmented reality-based navigation system for indoor environments," International Journal of Research in Engineering, vol. 6, no. 2, pp. 60-65, 2024. https://doi.org/10.33545/26648776.2024.v6.i2a.110 [Accessed 17-Mar-2026].

[10] A. S. Alluhaidan et al., "From GPS to AR: Leveraging augmented reality and grid-based systems for improved indoor navigation," IEEE Access, vol. 13, pp. 55210-55225, 2025. https://doi.org/10.1109/ACCESS.2025.10930457 [Accessed 15-Mar-2026].

[11] L. Zheng and H. Wang, "A review of research on SLAM technology based on the fusion of LiDAR and vision," Sensors, vol. 25, no. 5, p. 1447, 2024. https://doi.org/10.3390/s25051447 [Accessed 16-Mar-2026].

[12] J. Lokhande et al., "Campus navigation using augmented reality and virtual reality," International Journal for Multidisciplinary Research, vol. 7, no. 6, 2025. [Online]. Available: https://www.ijfmr.com/research-paper.php?id=19504 [Accessed 16-Mar-2026].

[13] Unity Documentation, Unity Technologies. [Online]. Available: https://docs.unity3d.com/Manual/index.html [Accessed 17-Mar-2026].

[14] Meta Quest 3 Specifications, Meta. [Online]. Available: https://www.meta.com/quest/quest-3/ [Accessed 17-Mar-2026].

[15] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," Pattern Recognition, vol. 47, no. 6, pp. 2280-2292, 2014. https://doi.org/10.1016/j.patcog.2014.01.005 [Accessed 17-Mar-2026].

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in European Conference on Computer Vision (ECCV), 2014, pp. 834-849. https://doi.org/10.1007/978-3-319-10605-2_54 [Accessed 17-Mar-2026].

[17] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), 2007, pp. 225-234. https://doi.org/10.1109/ISMAR.2007.4538852 [Accessed 17-Mar-2026].

[18] OpenXR Specification, The Khronos Group. [Online]. Available: https://www.khronos.org/openxr/ [Accessed 17-Mar-2026].

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Accessed 17-Mar-2026].

[20] Oculus Interaction SDK Documentation, Meta. [Online]. Available: https://developers.meta.com/horizon/documentation/unity/unity-isdk-interaction-sdk-overview/