Air University Multan Campus

SOFTWARE DESIGN DOCUMENT

(SDD DOCUMENT)

for

<AUMC GuideVR>

Version 1.0

By

Student Name 1	**233679_Abdul_Rafay**
Student Name 2	**233711_Abdul_Rehman**

Supervisor

Hafiz Muhammad Anas Ali

Bachelor of Science in Software Engineering (2023-2027)

Table of Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Abdul Rafay and Abdul Rehman	Initial version of the Software Design Document for AUMC GuideVR, including design methodology, architecture, major process flows, and all core UML diagrams.	15/06/2025

1. Introduction

Navigating large university campuses like Air University Multan Campus (AUMC) can be overwhelming, especially for new students, visitors, and even some faculty members. **AUMC GuideVR** addresses this challenge through an innovative virtual reality-based navigation solution. The application offers immersive visual guidance, an interactive map, and location-specific assistance to enhance campus accessibility and user experience.

1.1 Purpose of the Document

This Software Design Document (SDD) outlines the internal design, architectural layout, and implementation strategy for the *AUMC GuideVR* system. It serves as a bridge between the Software Requirements Specification (SRS) and the actual development of the system. This document is intended to guide developers, architects, and testers in understanding how the system will be structured and how its various components will interact.

1.2 Product Scope

AUMC GuideVR is an innovative VR-based campus navigation application designed to assist users in easily navigating the Air University Multan Campus (AUMC). Through the use of virtual reality, it provides real-time, 3D visual guidance to users—including students, faculty, and visitors—to locate essential areas such as classrooms, hostels, cafeterias, auditoriums, and administrative offices. Key features include floor-based navigation, search and distance calculation, zoom and rotation support, drone-view layout, and classroom identification based on schedules.

This project aims to reduce confusion, save time, and improve campus accessibility, especially for new students and parents unfamiliar with the campus environment.

However, while every effort will be made to implement all major features, the inclusion of certain high-complexity modules (such as real-time drone view simulation, classroom tracking, or 360-degree VR overlays) may be limited by technical, budgetary, or timeline constraints. In such cases, static alternatives, simplified versions, or placeholder modules will be developed to maintain the application's integrity and usability.

1.3 Intended Audience

This document is intended for the following user categories:

- **Students:** To understand how the system will assist in navigating the campus and accessing location-based information.
- **Faculty Members:** To explore how the system can support efficient movement across campus and share announcements or directions.
- **Visitors:** To benefit from guided navigation for specific locations within the campus.

2. Design Methodology and Software Process Model

For the development of AUMC GuideVR, a combination of Object-Oriented Design (OOD) and the Agile-Scrum methodology has been chosen. This approach aligns well with the system's modular nature and the evolving requirements during VR scene development, UI enhancement, and integration of components like bookmarks, search, and drone view.

2.1 Object Oriented Design

The system follows an **object-oriented design** approach rather than a traditional procedural design. Each core module of AUMC GuideVR (e.g., navigation, search, user session, POI manager) is treated as a separate object or class with its own properties and methods. This structure makes the application easier to:

- Understand
- Extend
- Maintain
- Test

2.1.1 Why OOP over Traditional Design?

- OOP allows better encapsulation of complex features like 3D route calculation or drone-view toggling.
- It supports reusability, meaning components like bookmarks or scene renderers can be reused across modules.
- Traditional design (top-down or linear) lacks the flexibility needed for a dynamic VR system, especially when features are added incrementally or adjusted after testing.

2.2 Software Process Model: Agile with Scrum

We have selected the **Agile process model**, specifically using the **Scrum framework**, for the following reasons:

2.2.1 Why Agile-Scrum?

- The system is being built in increments, not as one massive block.
- VR-based navigation systems require frequent testing, visual tuning, and user feedback — something Agile supports really well.
- Scrum allows us to break development into short, manageable sprints (2–3 weeks each), with reviews at the end of every sprint.

- Changes to the interface, labels, or POIs can be made mid-project without disrupting the entire timeline.

2.2.2 How Scrum Helps AUMC GuideVR?

- Daily stand-ups or review meetings help keep all team members in sync.
- Backlog items like "Add 360 view to Admin Block" or "Implement Search Suggestions" can be prioritized.
- Stakeholders can see demos after each sprint — great for validating visual and functional aspects of the VR experience.

2.3 Comparison with Other Models

Model	Why Not Preferred for AUMC GuideVR
Waterfall	Too rigid; does not allow mid-way feature improvements or visual UX changes
V-Model	Too test-heavy at early stages; not flexible for UI/VR flow revisions.
Spiral	More suitable for high-risk, complex systems — overkill for this academic VR project.
RAD	Good for small-scale prototypes but lacks structure for modular system design.

2.4 Benefits:

- Rapid feedback from actual users (students, faculty)
- Easy handling of UI changes or visual tweaks in 3D scenes
- Parallel development of components (e.g., while one developer works on bookmarks, another works on indoor scenes)
- Easier debugging and modular testing
- Smooth implementation of VR interaction patterns (zoom, rotate, pan)

3. System Overview

AUMC GuideVR is a web-based virtual reality application designed to guide users (students, faculty, and visitors) through the physical layout of **Air University Multan Campus**. The system presents an immersive 3D navigation experience, allowing users to explore the campus in virtual space without needing real-time camera input or GPS data.

Unlike traditional navigation apps or camera-based AR tools, AUMC GuideVR simulates the entire campus environment using VR scenes, 3D models, and pre-labeled rooms, allowing users to move through the university as if they are virtually present on-site.

a) Functionality at a Glance:

The core purpose of AUMC GuideVR is to help **students**, **faculty**, and **visitors** find their way around campus in a simple, interactive, and engaging manner. The system supports the following key features:

- **Indoor navigation** with floor-based path visualization
- **Outdoor navigation** using a bird's-eye (drone) view of the campus
- **Search and explore** functionality to quickly locate specific classrooms, offices, cafeterias, or labs
- **Bookmarking/favorites** to save and return to frequently visited places
- **360-degree** viewing of indoor and outdoor scenes
- **Zoom and rotation** controls for better spatial awareness in the VR environment
- A **responsive user interface** optimized for both desktop and mobile browsers

b) System Context and Design:

AUMC GuideVR operates entirely within a web browser using **WebXR**, **A-Frame**, and **Three.js** for rendering 3D scenes. It does not require camera or GPS access, which distinguishes it from AR (Augmented Reality) systems. Instead, users interact with a **pre-built virtual model** of the university campus that includes labeled buildings, rooms, and outdoor areas.

From a design standpoint, the system follows a **modular structure** where each major functionality (navigation, search, bookmarks, etc.) is developed as a separate component. This allows easier maintenance, testing, and future scalability.

c) Background and Need:

Navigating large campuses like AUMC can be overwhelming — especially for new students and first-time visitors. Without clear visual guidance, people often find it difficult to locate departments, classrooms, or administrative offices. AUMC GuideVR addresses this gap by offering a realistic virtual exploration experience, making it easier for users to:

- Plan their path
- Locate destinations
- Understand the campus layout visually

This system not only improves accessibility but also saves time and reduces confusion, particularly during orientation weeks, admission sessions, and campus events.

3.1 Architectural Design

The architecture of the **AUMC GuideVR** system is designed to be modular, scalable, and easy to maintain. To support browser-based virtual reality navigation, we have chosen a **three-tier architecture**, which separates the system into three logical layers:

3.1.1 Three-Tier Architecture Overview

- **Presentation Layer (Frontend – VR Interface):**
  - Built using HTML5, CSS3, JavaScript, and A-Frame/WebXR for VR rendering
  - This layer handles user interaction, scene loading, and rendering of 3D elements
  - Includes the interface for search, floor switching, bookmarks, and VR navigation modes
- **Application Logic Layer (Backend APIs):**
  - Built using Node.js with Express
  - Manages communication between the frontend and the database
  - Handles search queries, bookmark saving, POI retrieval, and feedback submission
- **Data Layer (Database):**
  - Stores structured data including room numbers, faculty/office locations, bookmarks, building metadata, and POIs
  - Recommended databases: MongoDB or Firebase (NoSQL) for flexibility
  - Supports secure data fetch, write, and update operations through backend APIs

3.1.2 Module Relationships (How they work together)

The system is decomposed into several major functional modules, each responsible for one aspect of the application. These modules are **loosely coupled** and **highly cohesive**, meaning they perform distinct tasks but work together seamlessly.

Module	Responsibility
VR Navigator	Handles indoor and outdoor path rendering, floor switching, and camera movement within the 3D scene
Search & Explore	Provides intelligent search suggestions and POI filtering
Bookmark Manager	Saves and loads user-saved locations from local browser storage
360° Viewer	Renders a panoramic view of selected buildings or open areas
Drone View	Provides a top-down map for overview-based navigation
Data API Connector	Facilitates secure data communication between frontend and backend
Admin Tools (optional future module)	Allows upload/update of new POIs, floor plans, and labels

These modules interact through well-defined interfaces and pass data (e.g., selected destination, POI details) between layers. The **backend logic** processes the requests and responds with accurate navigation data, which the **frontend** then uses to update the VR scene.

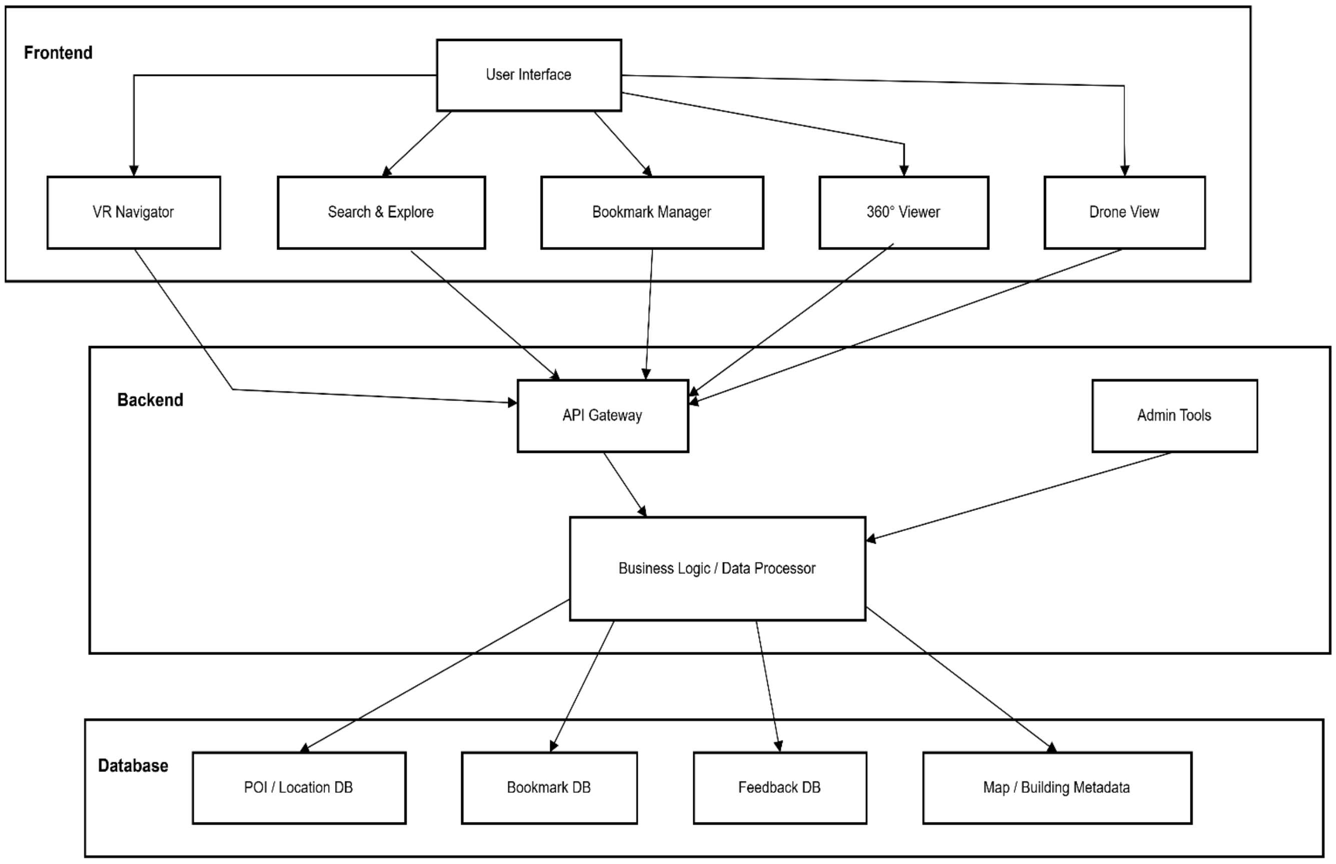
3.1.3 Why this Architecture?

- A **three-tier structure** allows clean separation of UI, logic, and data, making the system easier to:

  - Maintain
  - Expand in the future
  - Reuse across campuses

- It supports a **component-based, object-oriented design**, where each module can be modified or updated independently
- The use of **web-based VR frameworks (A-Frame/WebXR)** works best when decoupled from backend logic, which this architecture supports well

3.1.4 Diagram



3.2 Process Flow

To clearly understand how the system operates, this section outlines the flow of **major processes** within the AUMC GuideVR system. These activity flows illustrate the interaction between the user and the system for key features such as indoor navigation, location search, and bookmarking functionality.

The purpose of these diagrams is to provide a high-level overview of how the system guides users through its core operations and how different components collaborate to deliver the expected VR navigation experience.

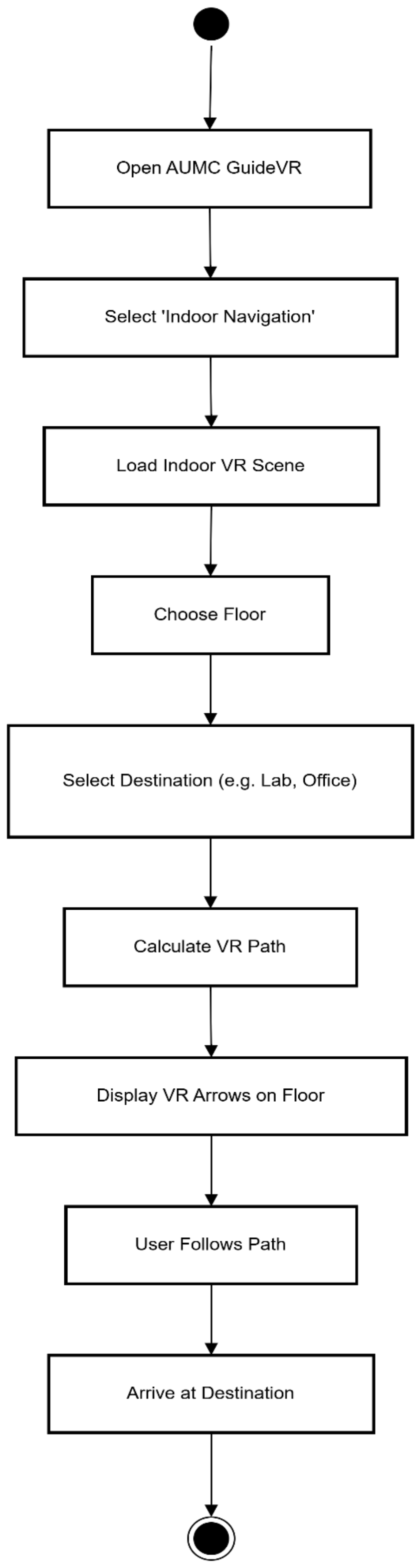
3.2.1 Indoor Navigation

This process represents how a user initiates indoor VR navigation within a building on the AUMC campus.

a) Process Description:

- The user opens the AUMC GuideVR application.
- They select the “Indoor Navigation” option from the main menu.
- The system loads the indoor VR scene.
- The user selects the desired building floor.
- A specific classroom, lab, or office is chosen as the destination.
- The system calculates a virtual navigation path.
- The path is displayed through VR arrows or indicators.
- The user follows the path until they reach their selected destination.

b) Activity Diagram:



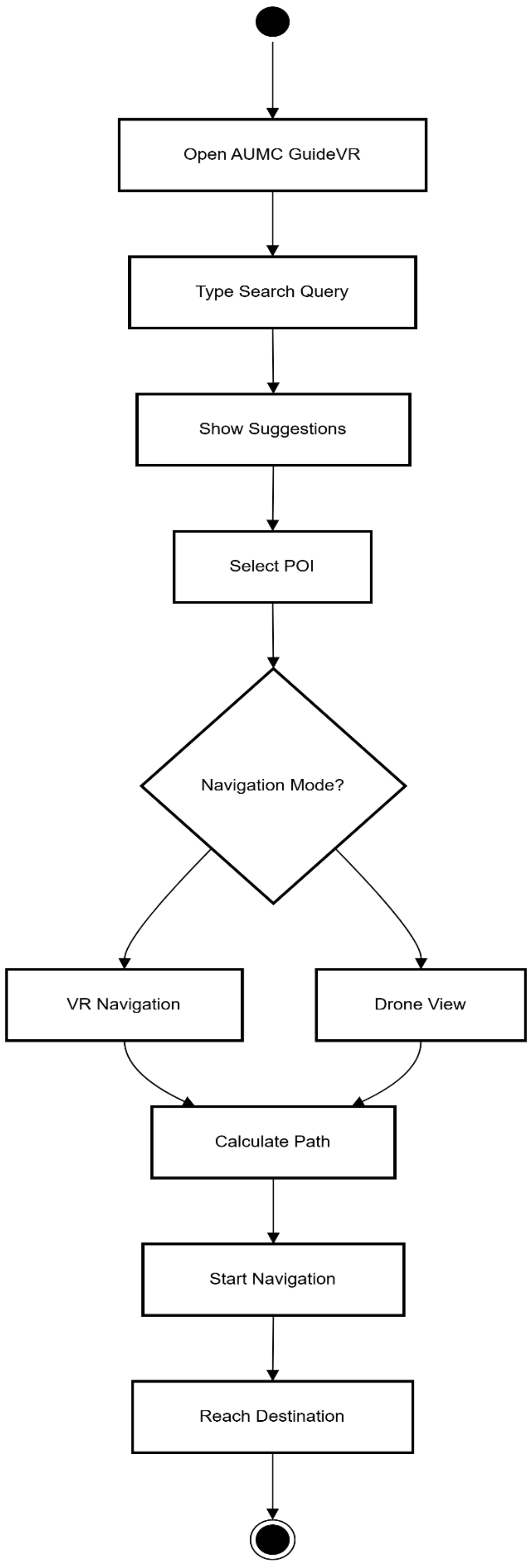
3.2.2 Search and Navigate

This process outlines how the user searches for a place (e.g., library, finance office, IT department) and starts navigation directly from the search results.

a) Process Description:

- The user launches the AUMC GuideVR system.
- They enter a keyword into the search bar.
- The system displays auto-suggestions based on the query.
- The user selects a location from the results.
- A prompt allows the user to choose a navigation mode (VR mode or Drone View).
- Based on the selection, the system calculates a suitable path.
- Navigation begins and continues until the destination is reached.

b) Activity Diagram:



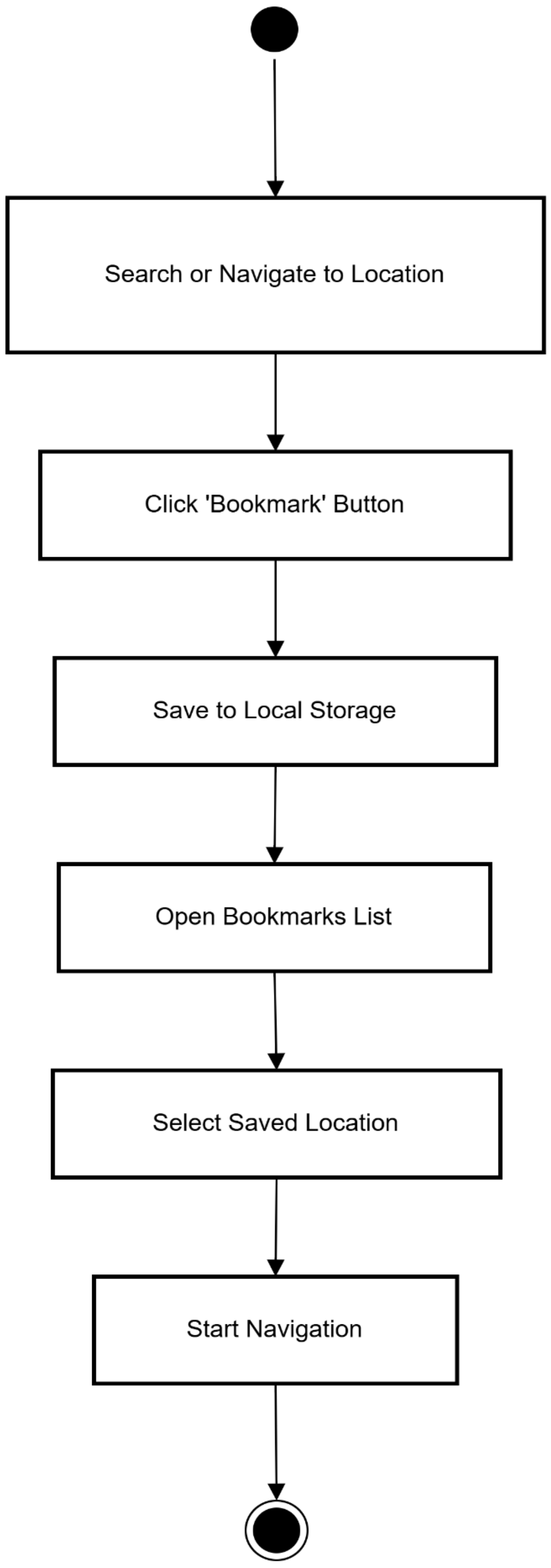
3.2.3 Bookmarking a Location

This process details how users can save important or frequently visited locations and later use them for quick navigation.

a) Process Description:

- The user either searches for or navigates to a location.
- They choose to bookmark the current location.
- The system saves the location in the browser’s local storage.
- At a later point, the user accesses the “Bookmarks” list.
- They select a saved location from the list.
- The system initiates navigation to that bookmarked point.

b) Activity Diagram:



4. Design Models

4.1 Models and Diagrams

4.1.1 Use Case Diagram

a) Description:

The Use Case Diagram for the AUMC GuideVR system represents the interaction between the system and its primary users: students, faculty, and visitors. It visually outlines the functional requirements of the system based on user goals, and organizes these into three main modules:

- Indoor Navigation
- Outdoor Navigation
- Bookmark & Saved Location Management

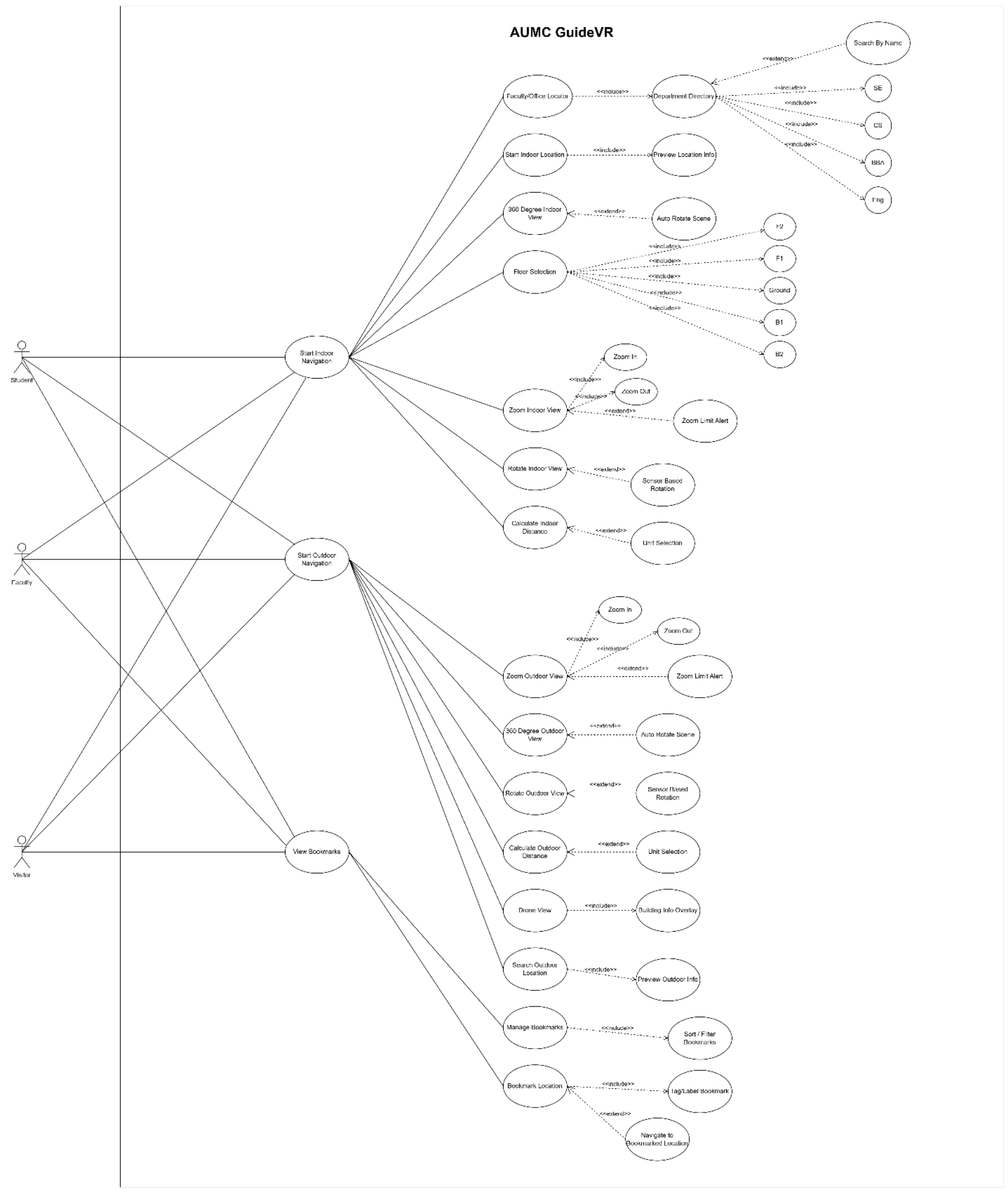
Each module consists of core functional use cases (like searching, viewing 360 scenes, zooming, rotating, etc.) that are directly triggered by the actors. These functional use cases are further extended with optional or reusable features (e.g., unit selection, zoom limit alerts, tag labeling) using <<include>> and <<extend>> relationships to ensure modularity and reusability.

This model helps in:

- Understanding what actions users can perform
- Visualizing system boundaries
- Identifying sub-features that can be optionally invoked
- Laying the foundation for sequence diagrams and detailed process modeling

The diagram strictly follows the Object-Oriented Approach, using actors, use cases, and UML-compliant relationships like <<include>> and <<extend>>.

b) Diagram:


AUMC GuideVR
Student
Faculty
Visitor
Start Indoor Navigation
Start Outdoor Navigation
View Bookmarks
Faculty/Office Locator
Department Directory
Search By Name
SE
CS
BBA
Eng
Start Indoor Location
Preview Location Info
360 Degree Indoor View
Auto Rotate Scene
Floor Selection
F2
F1
Ground
B1
B2
Zoom Indoor View
Zoom In
Zoom Out
Zoom Limit Alert
Rotate Indoor View
Sensor Based Rotation
Calculate Indoor Distance
Unit Selection
Zoom Outdoor View
Zoom In
Zoom Out
Zoom Limit Alert
360 Degree Outdoor View
Auto Rotate Scene
Rotate Outdoor View
Sensor Based Rotation
Calculate Outdoor Distance
Unit Selection
Drone View
Building Info Overlay
Search Outdoor Location
Preview Outdoor Info
Manage Bookmarks
Sort / Filter Bookmarks
Bookmark Location
Tag/Label Bookmark
Navigate to Bookmarked Location
<<include>>
<<extend>>

4.1.2 Class Diagram

1) Description:

The class diagram for AUMC GuideVR represents the overall object-oriented design structure of the application. It visualizes how core components, functional modules, and user roles are logically organized and interconnected through classes, attributes, methods, and relationships.

This system is designed using Object-Oriented Programming (OOP) principles. The diagram reflects modular design, reusability, and extensibility — all of which are necessary for building a scalable browser-based virtual reality navigation experience.

The system is decomposed into the following major components:

1. User Roles (Actors)

- A base class User is created with shared attributes like userID, name, and role, and a common method navigate().
- It is inherited by three actor classes: Student, Faculty, and Visitor, representing the end-users of the system.
- This approach allows flexibility in defining role-specific behavior later, if needed.

2. Main System Controller

- The AUMC GuideVR class acts as the central system that manages and triggers different navigation modules.
- It has three key methods:
  - startIndoorNavigation()
  - startOutdoorNavigation()
  - startBookmarkNavigation()
- It aggregates the modules IndoorNavigation, OutdoorNavigation, and BookmarkManager.

3. Indoor Navigation Module

- IndoorNavigation handles all indoor-specific functionality like:
  - Searching for rooms
  - Locating faculty
  - Floor switching
- It is associated with:
  - IndoorVRView: for rotating, zooming, and showing 360° indoor scenes
  - IndoorDistanceCalculator: for measuring distances indoors
  - UnitSelector: for selecting metric or imperial units
  - ZoomLimiter: for restricting excessive zoom levels
  - Room and Location classes for physical mapping

4. Outdoor Navigation Module

- OutdoorNavigation is designed similarly, with methods to:
    - Search outdoor places
    - Show drone view
    - Launch 360° outdoor view
- It connects to:
    - OutdoorVRView (for rotating, zooming, etc.)
    - OutdoorDistanceCalculator
    - UnitSelector and ZoomLimiter
    - Location class for geographic coordinates

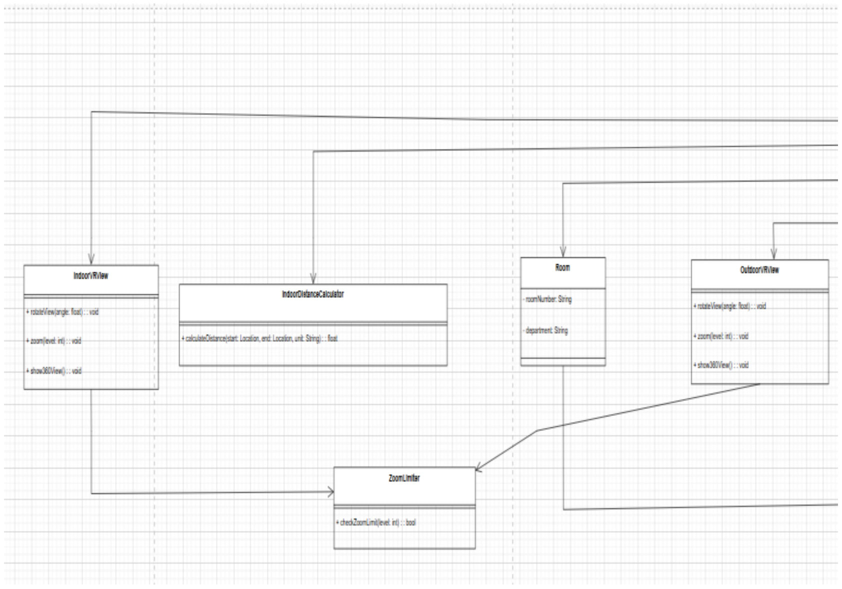
5. Bookmark & Saved Location Module

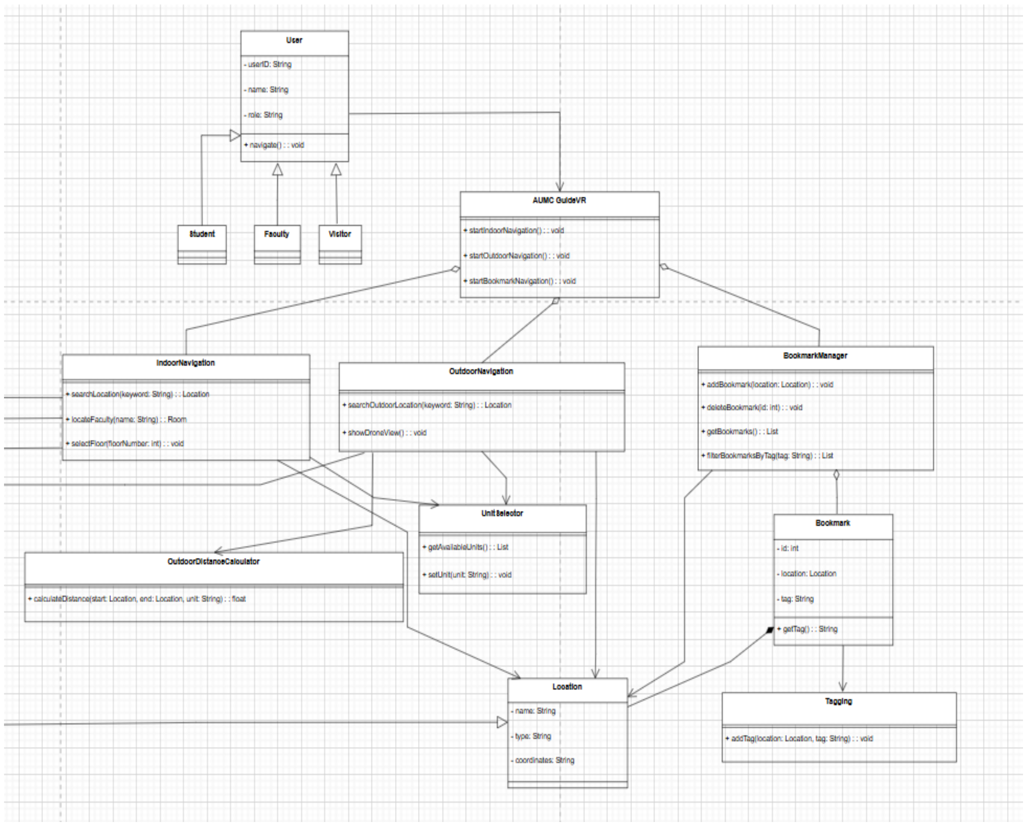
- BookmarkManager allows users to:
    - Save locations (Bookmark)
    - Filter them by tag
    - Navigate directly from bookmarks
- Bookmark is composed of a Location (meaning it cannot exist without a location) and holds a tag value.
- The Tagging class is used to add or manage tags for each saved place.

6. Shared Components

- Location: Base class representing any physical place (indoor or outdoor)
- Room: Inherits from Location and adds attributes like room number and department
- UnitSelector, ZoomLimiter, and Tagging are support utilities used across modules to support feature extensions (e.g., measurement units, limits, labels)

2) Diagram:


IndoorVRView
+ rotateView(angle: float) : : void
+ zoom(level: int) : : void
+ show360View() : : void
IndoorDistanceCalculator
+ calculateDistance(start: Location, end: Location, unit: String) : : float
Room
- roomNumber: String
- department: String
OutdoorVRView
+ rotateView(angle: float) : : void
+ zoom(level: int) : : void
+ show360View() : : void
ZoomLimiter
+ checkZoomLimit(level: int) : : bool

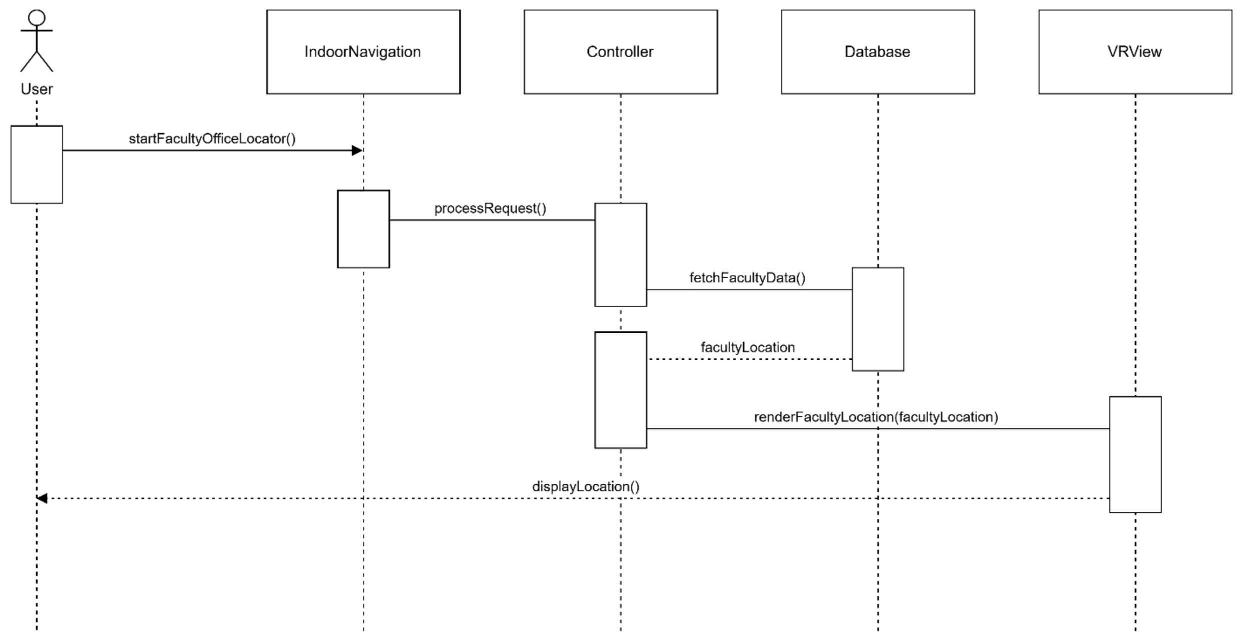

User
- userID: String
- name: String
- role: String
+ navigate() : : void
Student
Faculty
Visitor
AUMC GuideVR
+ startIndoorNavigation() : : void
+ startOutdoorNavigation() : : void
+ startBookmarkNavigation() : : void
IndoorNavigation
+ searchLocation(keyword: String) : : Location
+ locateFaculty(name: String) : : Room
+ selectFloor(floorNumber: int) : : void
OutdoorNavigation
+ searchOutdoorLocation(keyword: String) : : Location
+ showDroneView() : : void
BookmarkManager
+ addBookmark(location: Location) : : void
+ deleteBookmark(id: int) : : void
+ getBookmarks() : : List
+ filterBookmarksByTag(tag: String) : : List
UnitSelector
+ getAvailableUnits() : : List
+ setUnit(unit: String) : : void
OutdoorDistanceCalculator
+ calculateDistance(start: Location, end: Location, unit: String) : : float
Bookmark
- id: int
- location: Location
- tag: String
+ getTag() : : String
Location
- name: String
- type: String
- coordinates: String
Tagging
+ addTag(location: Location, tag: String) : : void

4.1.3 Sequence Diagram

1) Module: Indoor Navigation

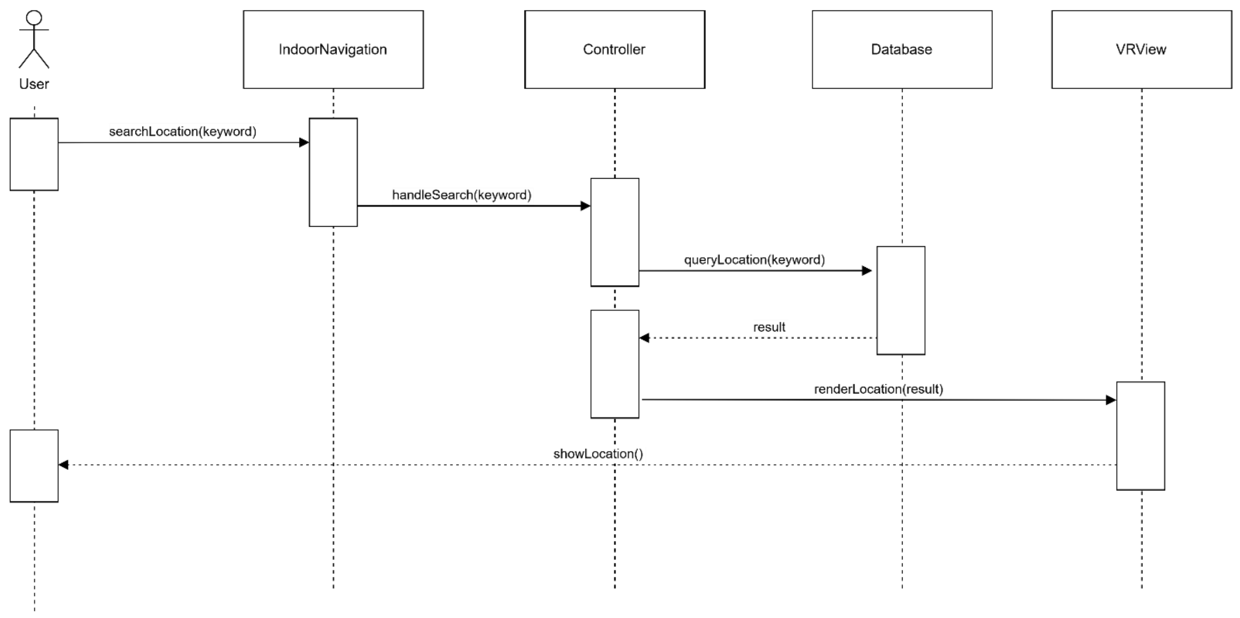
a) Faculty / Office Locator

This sequence diagram illustrates the interaction between the user and the system when locating a specific faculty member or office. The user initiates a request to search for a faculty room. The system controller communicates with the database to retrieve the room's location and forwards it to the VR view, which then displays the visual guidance inside the virtual environment.



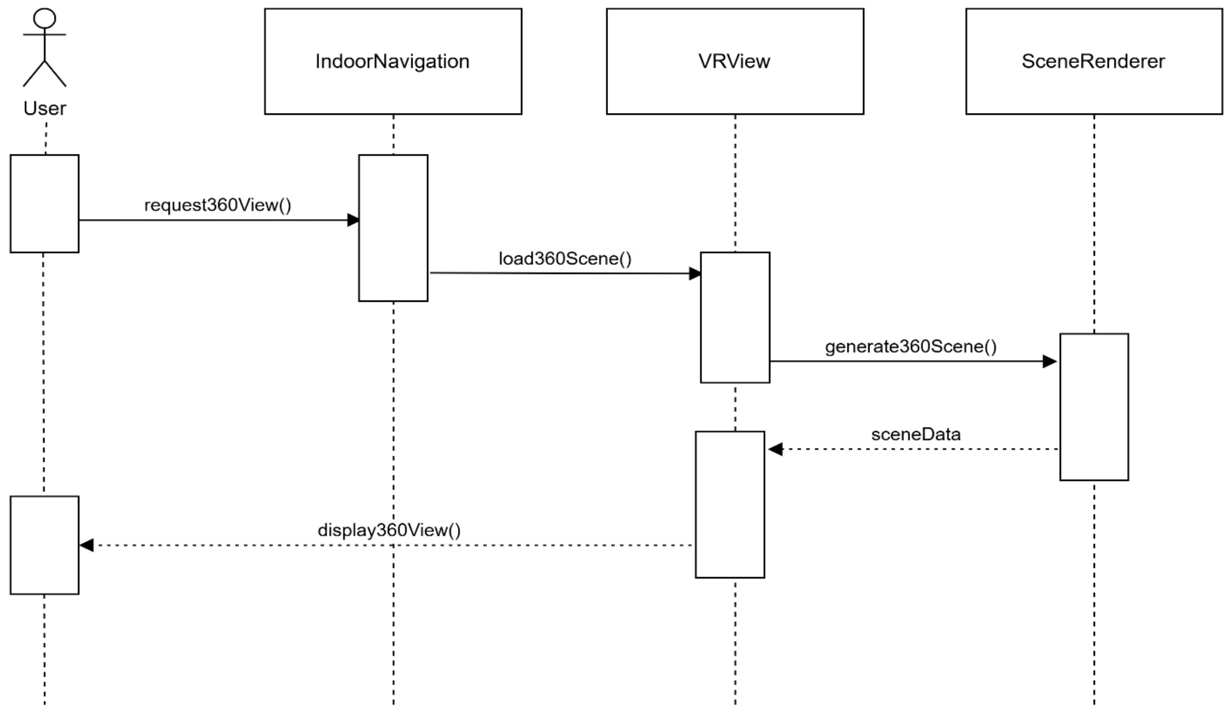
b) Search / Indoor Location

This diagram captures the process of searching for an indoor location such as a lab, classroom, or admin office. The user provides a search keyword, which is processed by the controller. The controller queries the database and sends the results to the VR interface to be visually rendered in the virtual campus.



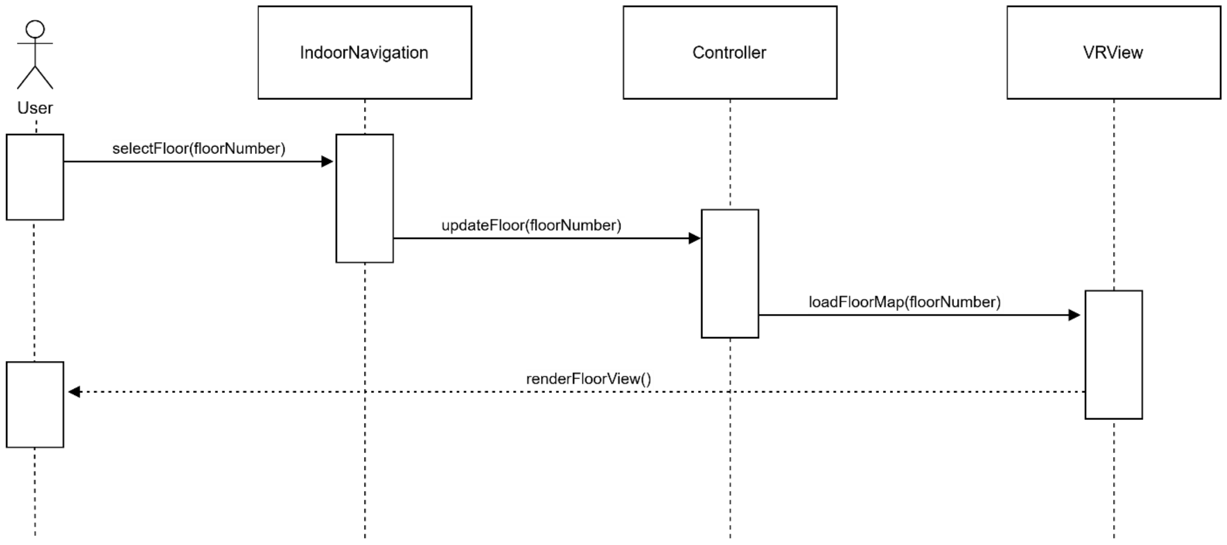
c) 360° Indoor View

This sequence depicts how a 360-degree indoor scene is generated and displayed. The user requests a 360° view of a particular location. The system loads the respective scene using the VR view and renders the immersive view via the SceneRenderer. This enhances user orientation within indoor spaces.



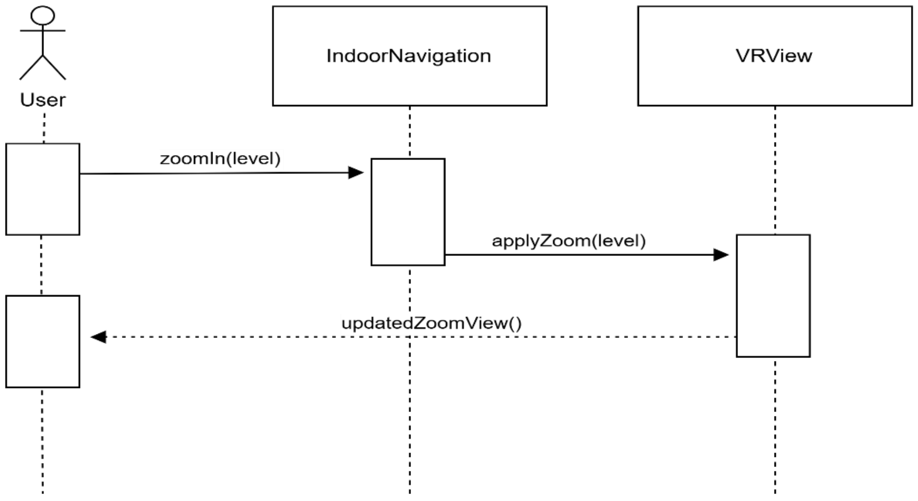
d) Floor Selection

This sequence illustrates the logic behind selecting a specific floor within a multi-story building. When a user selects a floor, the request is processed by the controller, which then updates the current floor and triggers the VR interface to display the relevant map layer for that floor.



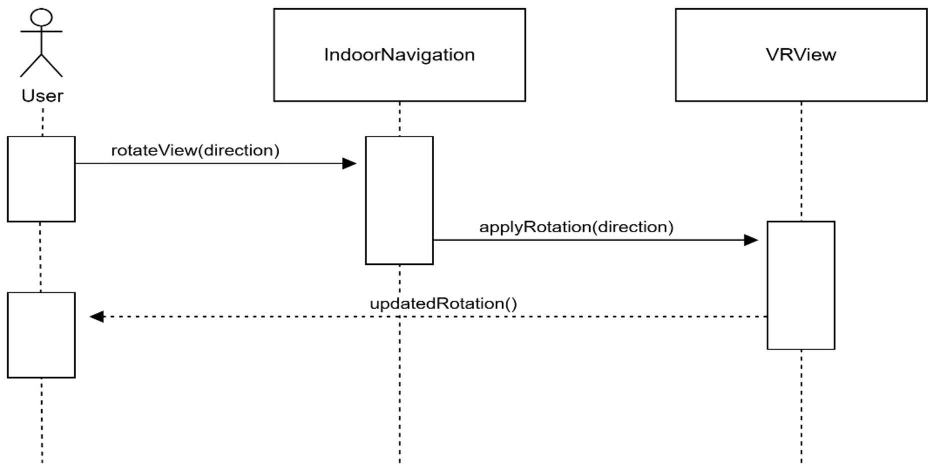
e) Zoom Indoor View

The zoom sequence represents how users interact with the virtual view to focus on a specific part of the building. Upon triggering a zoom level, the system relays this command to the VR view, which updates the visual rendering to reflect the new zoom state. Zooming is especially helpful in crowded or detailed areas.



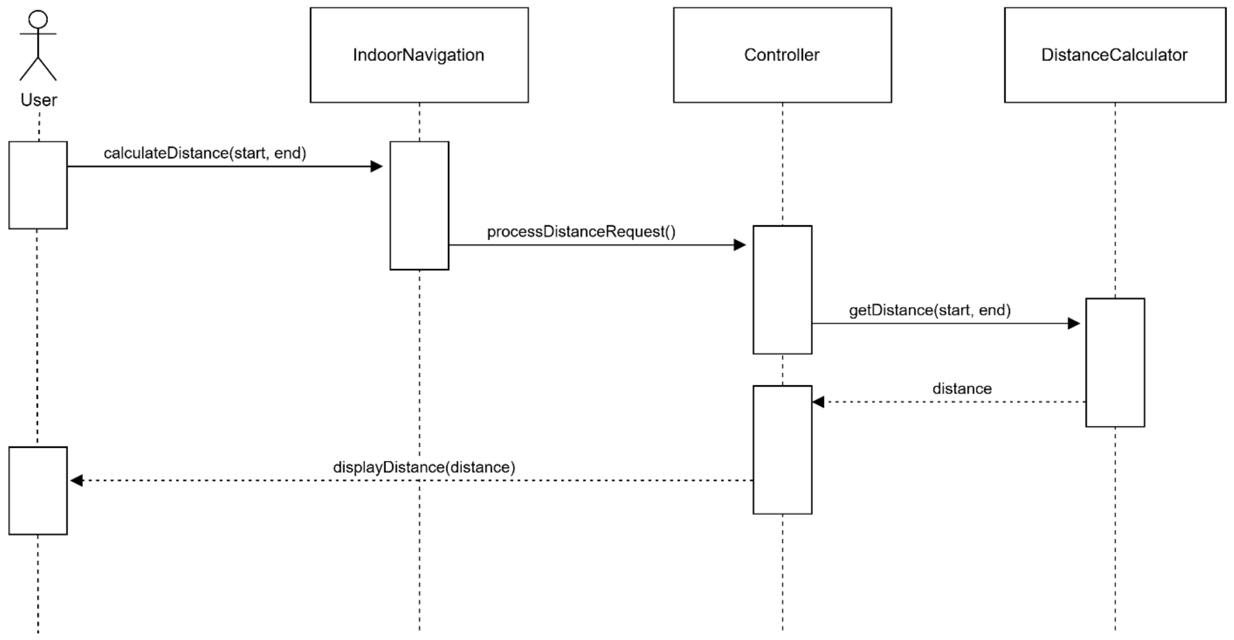
f) Rotate Indoor View

This diagram showcases how rotation works in the VR environment. A user request to rotate the view is passed through the Indoor Navigation module to the VRView component, which adjusts the visual orientation accordingly. This helps align the user's perspective with the direction of navigation.



g) Calculate Indoor Distance

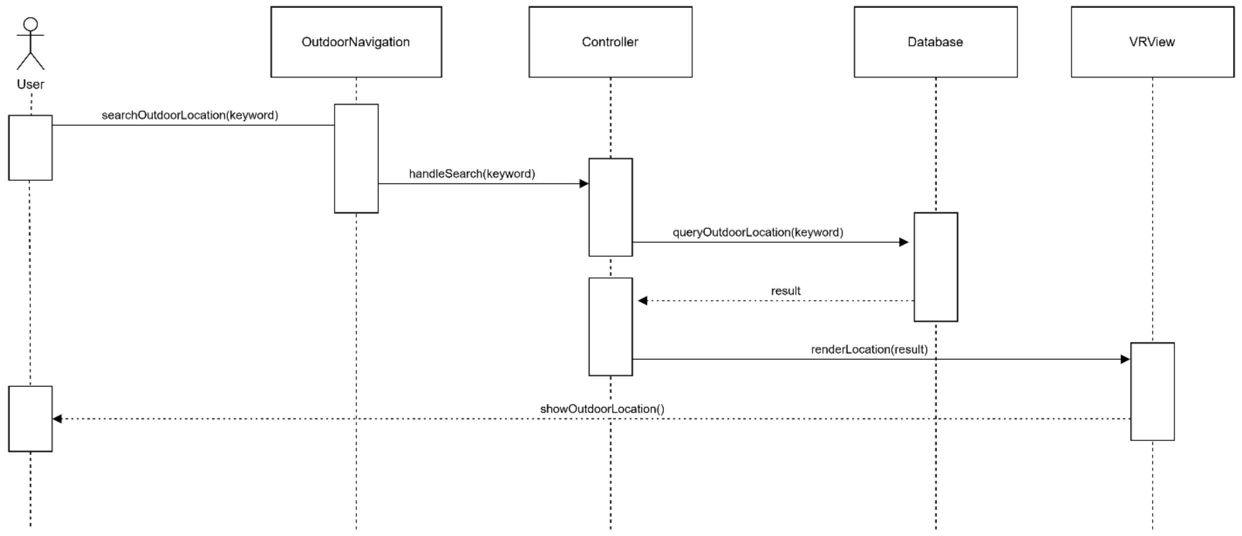
This sequence shows the flow of calculating the distance between two indoor points. The user specifies a start and end location, which the controller forwards to the DistanceCalculator. The calculated distance is then returned and displayed to the user, optionally supporting units like meters or feet.



2) Module: Outdoor Navigation

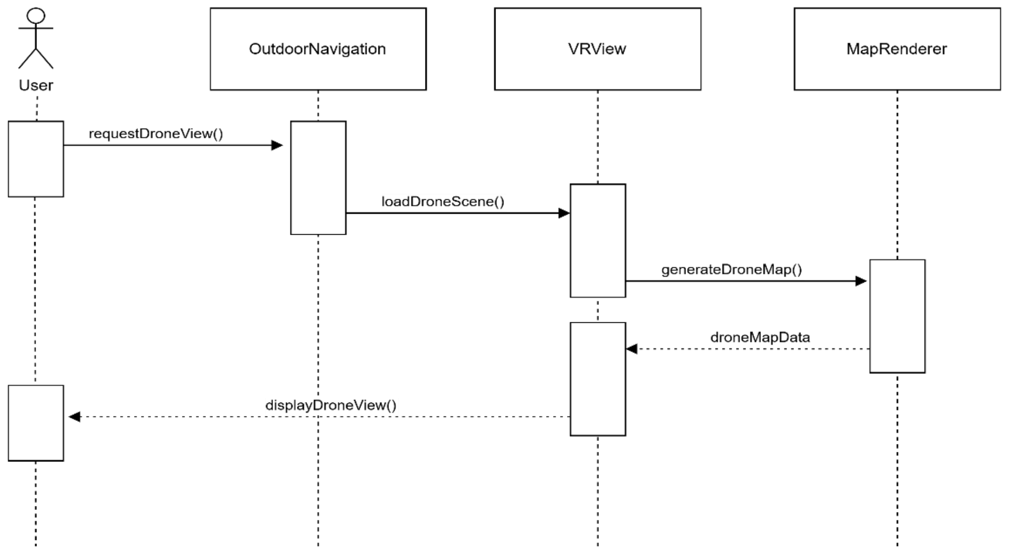
a) Search Outdoor Location

This sequence diagram shows how the system handles an outdoor location search initiated by the user. The user enters a keyword, which is passed to the controller. The controller queries the database and sends back the matched outdoor location result. The VR view then renders the visual output to the user within the virtual map environment.



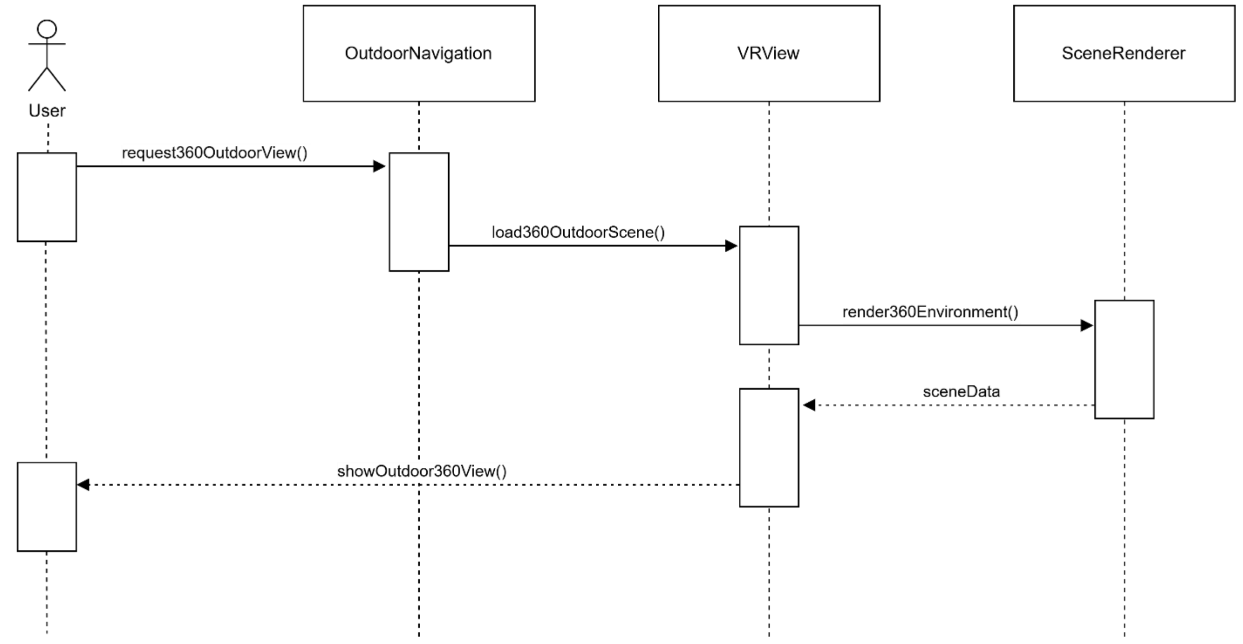
b) Drone View

This diagram represents how a drone-like bird's-eye view of the campus is generated. When the user selects the drone view option, the Outdoor Navigation module activates the VR interface, which then calls the MapRenderer to create a top-down 3D campus visualization. The generated view is returned to the user for visual exploration.



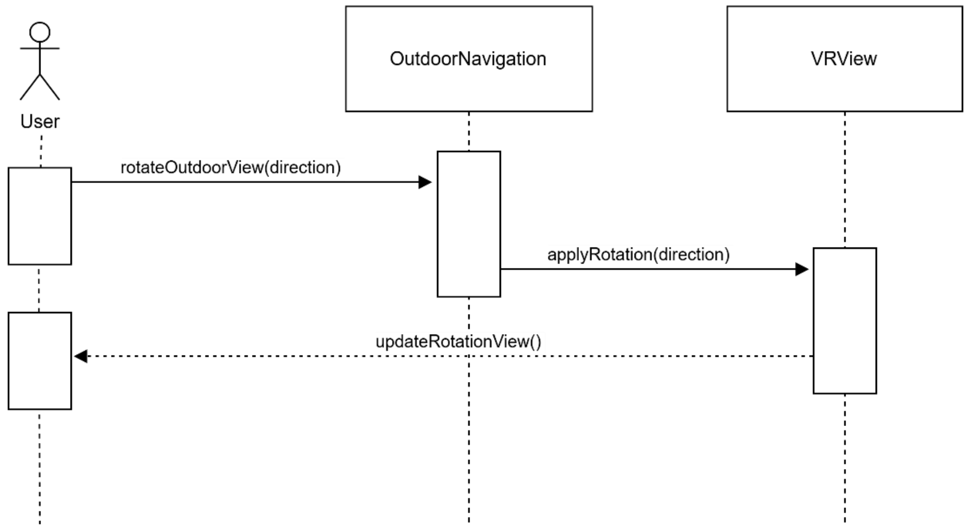
c) 360° Outdoor View

This diagram details the steps to provide a panoramic 360° view of an outdoor scene. Upon request, the system loads the corresponding scene via the VR view and uses the SceneRenderer to generate immersive visuals. This helps users get a complete visual understanding of their surroundings in outdoor areas like courtyards, parking, or gardens.



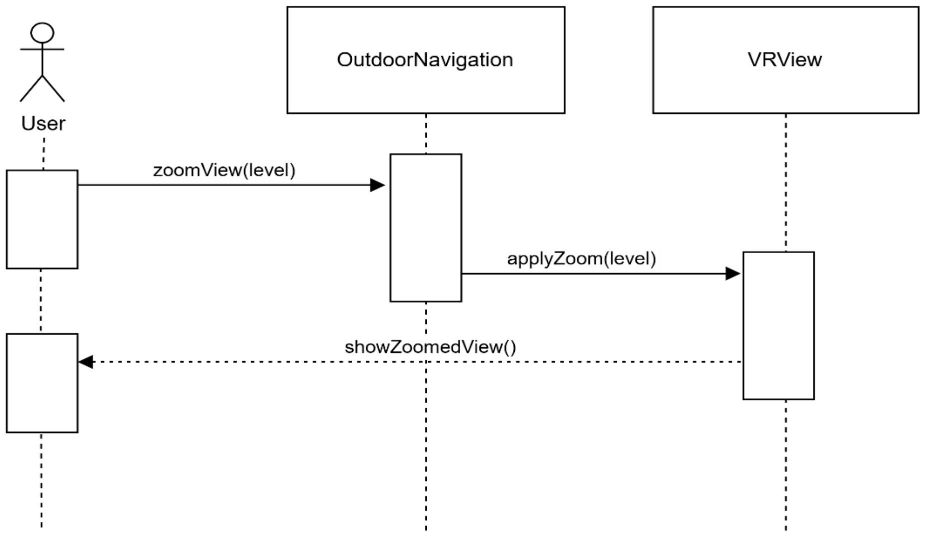
d) Rotate Outdoor View

This sequence illustrates how rotation functionality is implemented for outdoor scenes. When a user rotates the view, the Outdoor Navigation module calls the VRView to apply the requested angle, which is then reflected visually to help users align their direction outdoors.



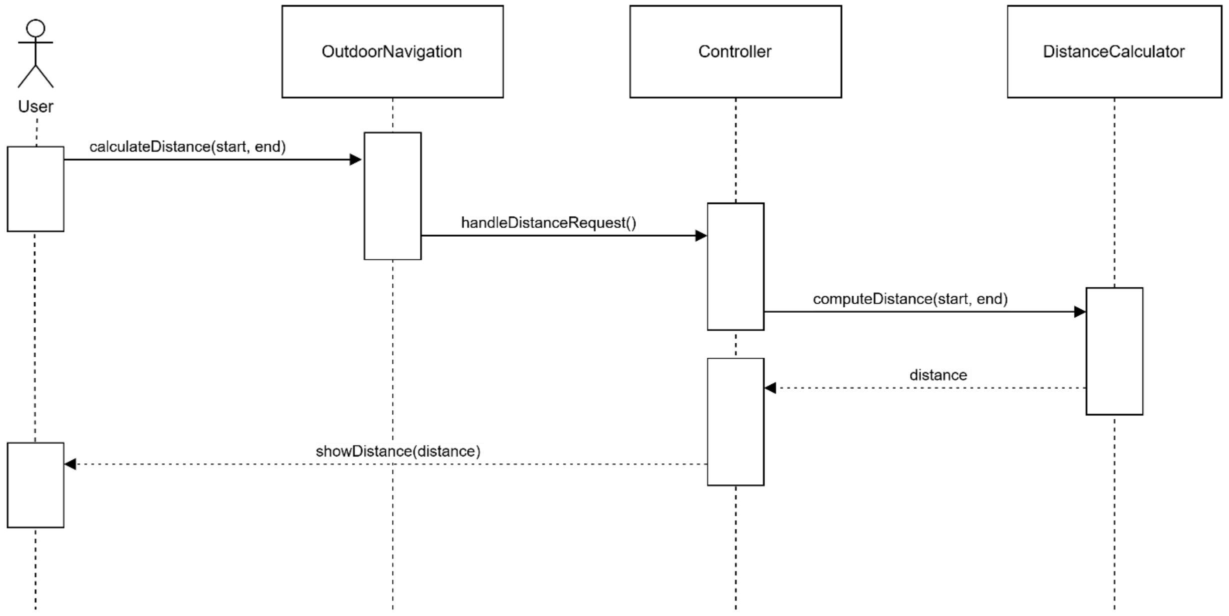
e) Zoom Outdoor View

This diagram shows how the system enables zooming in outdoor areas. The user initiates a zoom level change, which is passed to the VR view via the Outdoor Navigation module. The updated zoomed scene is then displayed, helping users focus on specific map or building sections.



f) Calculate Outdoor Distance

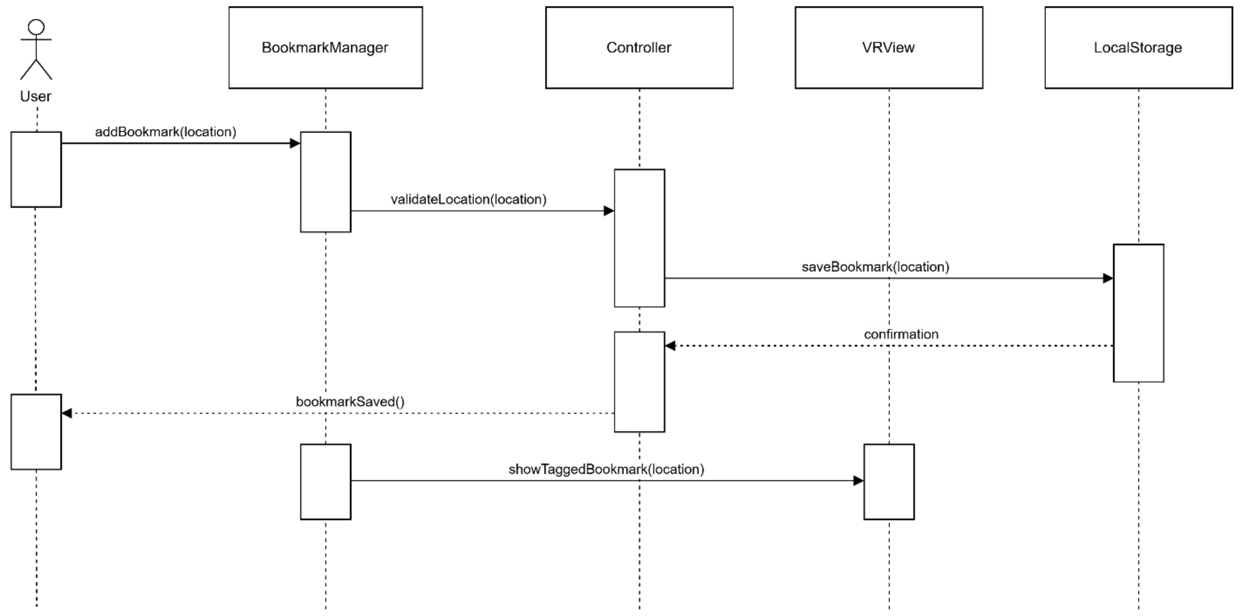
This diagram outlines how the system calculates the distance between two outdoor points. The request is processed by the controller, which forwards it to the DistanceCalculator utility. Once the distance is computed, it is sent back to the user in the selected unit (e.g., meters or feet).



3) Module: View Bookmark Locations

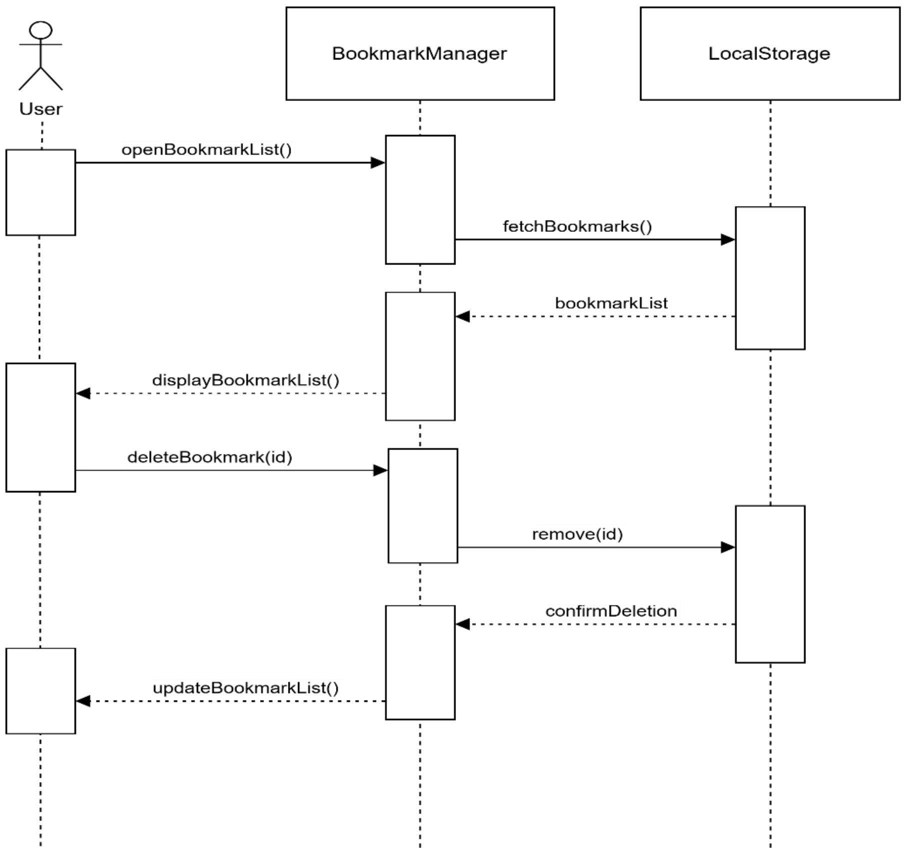
a) Bookmark Locations

This diagram illustrates the process of bookmarking a location within the AUMC GuideVR system. The user selects a location and requests to bookmark it. The BookmarkManager validates the location and sends it to the controller, which stores the data in the device's local storage. Upon successful saving, a confirmation is sent back to the user. Optionally, the location is visually tagged and displayed on the VR interface for quick recognition.



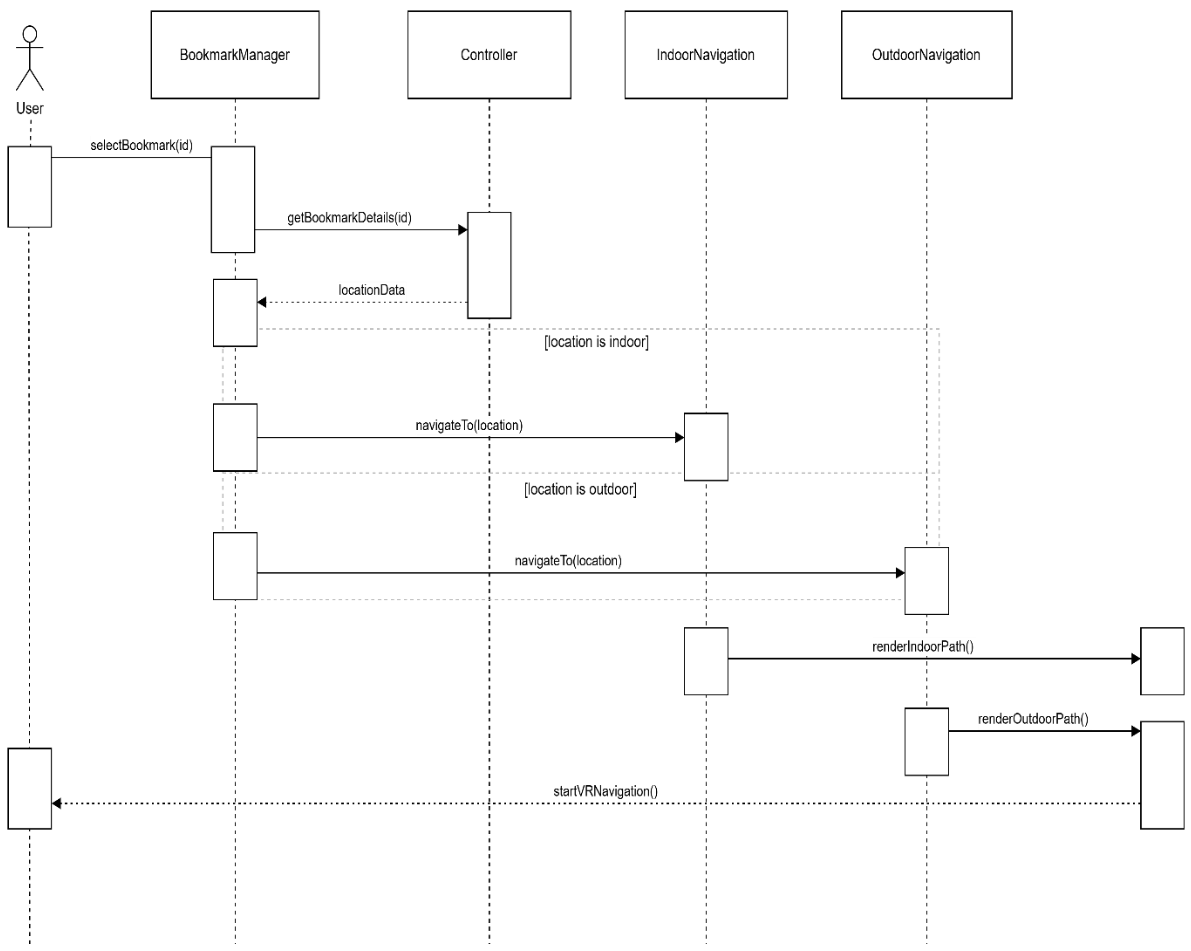
b) Manage Bookmarks

This sequence captures the flow of viewing and managing saved bookmarks. When the user opens their saved bookmarks, the BookmarkManager fetches the stored data from local storage and displays the list. Users may then choose to delete a bookmark, in which case a deletion request is sent, and the updated list is shown after confirmation. This provides flexibility and control over saved locations without requiring backend access.



c) Navigate to Bookmarked Location

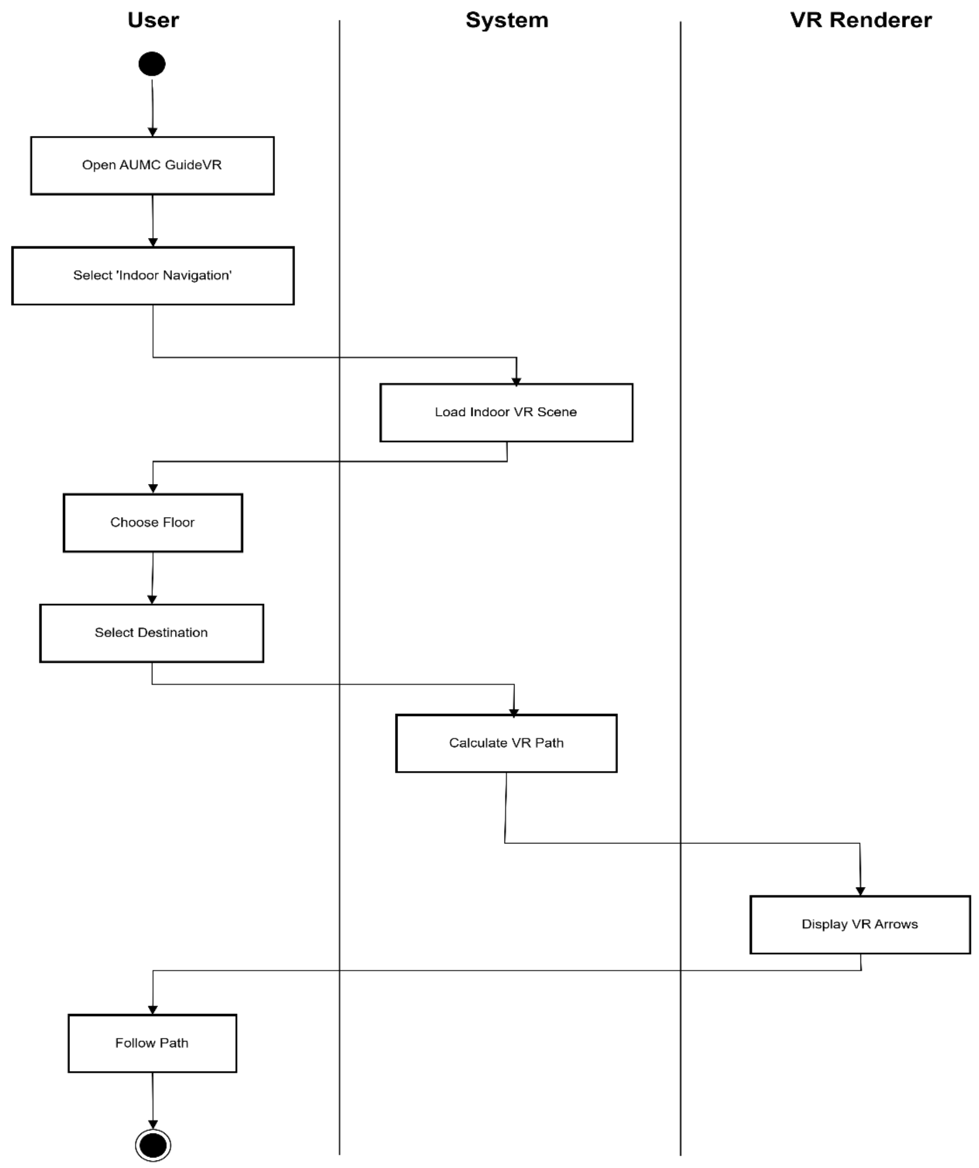
This diagram shows how the system handles navigation to a previously bookmarked location. Afterselecting a bookmark, its details are retrieved via the controller. The system checks whether the location is indoor or outdoor, and accordingly routes the request to the appropriate navigation module. The chosen module then initiates the path rendering using the VR interface, allowing the user to begin seamless VR-based navigation to their saved location.



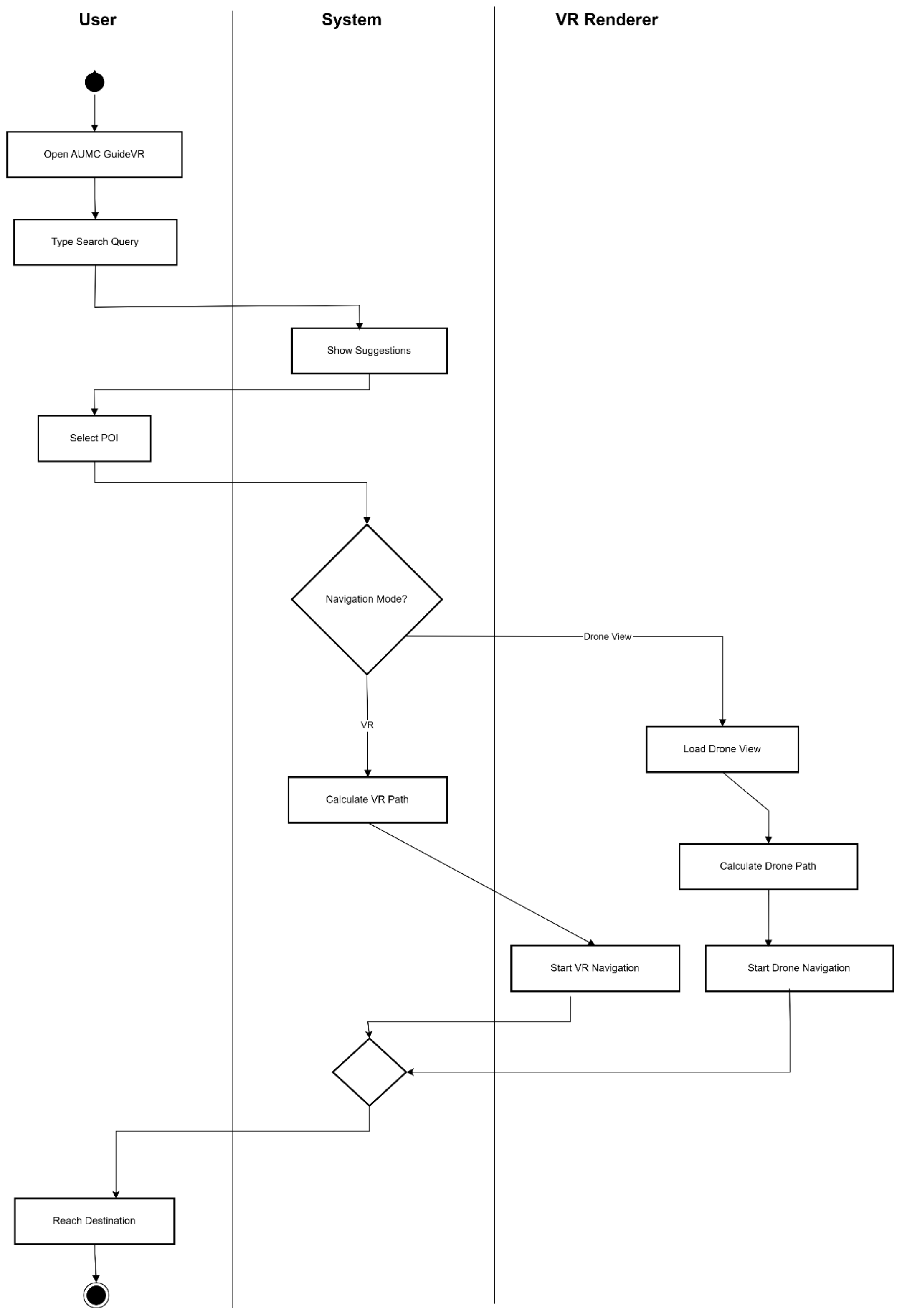
User
BookmarkManager
Controller
IndoorNavigation
OutdoorNavigation
selectBookmark(id)
getBookmarkDetails(id)
locationData
[location is indoor]
navigateTo(location)
[location is outdoor]
navigateTo(location)
renderIndoorPath()
renderOutdoorPath()
startVRNavigation()

4.1.4 Swim Lane Diagram

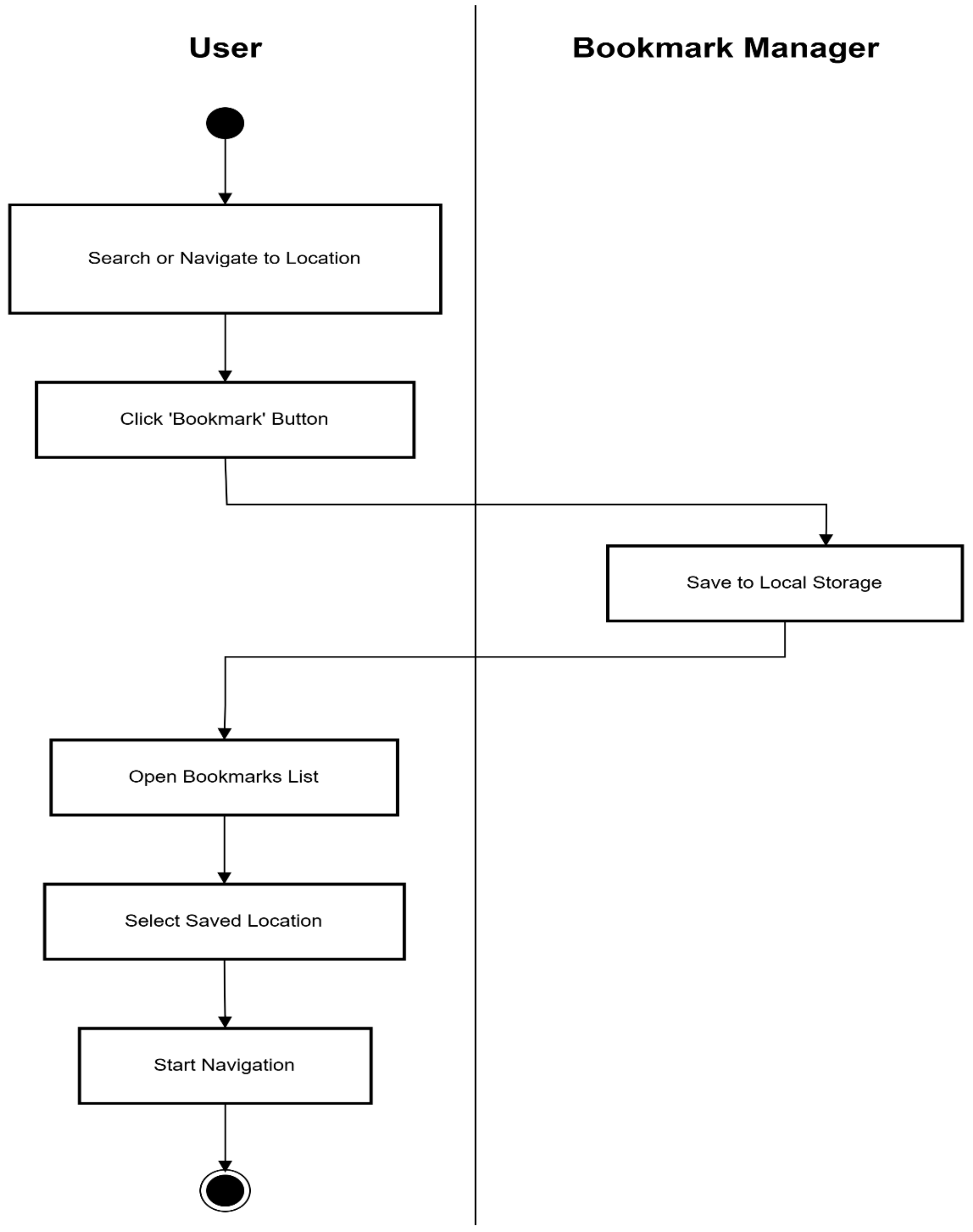
1) Indoor Navigation



2) Search and Navigate



3) Bookmark Location


User
Bookmark Manager
Search or Navigate to Location
Click 'Bookmark' Button
Save to Local Storage
Open Bookmarks List
Select Saved Location
Start Navigation