

**CSE6224 SOFTWARE REQUIREMENTS ENGINEERING**

**PROJECT PART 1**

|  |  |
| --- | --- |
| **NAME** | **STUDENT ID** |
| Iusupov Alimzhan | 1231301318 |
| GALAL OSMAN GALAL MOHAMED | 1221307698 |
| BALFAQIH AHMED | 1221304386 |
|  |  |

## **1. Introduction**

Navigating a university campus can be challenging for individuals with mobility or accessibility needs, especially when faced with unexpected obstacles such as elevator outages, construction zones, or event-related disruptions. To address this issue, the **Campus Accessibility Navigation System with Facilities and Event Integration (CANS)** provides an intelligent, real-time navigation solution designed specifically to support accessibility.

This system integrates live data from the university’s Facilities Management System and Event Calendar to offer updated, accessible route guidance. Through a user-friendly web application, students, staff, and visitors can confidently plan routes while avoiding inaccessible paths and disruptions.

By prioritizing accessibility, real-time updates, and integration with existing campus infrastructure, this project promotes independence, inclusivity, and operational efficiency across the university environment. The system also supports non-functional goals such as performance, reliability, and data security, aligning with institutional IT standards and user privacy regulations.

### **1.1 Purpose**

The purpose of this Software Requirements Specification (SRS) is to define the **functional** and **non-functional** requirements for the Campus Accessibility Navigation System with Facilities and Event Integration (**CANS**). This document is intended for developers, university IT administrators, QA personnel, and stakeholders to ensure a shared understanding of system expectations.

### CANS aims to provide a centralized, intelligent, real-time navigation platform that integrates with existing university systems—including the Facilities Management System and Event Calendar. The system is designed to offer timely and accurate route planning, infrastructure status, and accessibility-aware event listings.

### **Primary Goals of the System:**

1. **For Students and Campus Visitors:**

* View real-time, accessible navigation routes across campus.
* Receive alerts for elevator outages, construction zones, and path changes.
* Report accessibility issues with photo uploads and geolocation tagging.
* Discover events with accessibility accommodations using filtering options.

1. **For Staff Members:**

* Update real-time facility statuses (e.g., elevator outages, pathway reopenings).
* Create and manage event details, including accessibility metadata.
* Address and resolve accessibility reports submitted by users.

1. **For Administrators:**

* Manage user roles, permissions, and system-wide access.
* Review and approve submitted event and facility data.
* Sync data with external university services.
* Generate usage analytics and accessibility reports for continuous improvement.

In addition to these goals, the system emphasizes non-functional qualities such as secure data handling, accessibility compliance (WCAG 2.1), and consistent availability.

### **1.2 Scope**

The Campus Accessibility Navigation System (**CANS**) is a web-based platform designed to enhance campus mobility and accessibility, particularly for individuals with disabilities or mobility challenges. It leverages real-time facility status updates, event metadata, and user-generated reports to improve navigation outcomes.

### CANS will be accessible on both desktop and mobile browsers, integrating with university databases to ensure up-to-date and accurate information. Features include accessible route generation, disruption alerts, event filtering by accessibility, issue reporting, and administrative tools.

### **1.3 Product Overview**

#### **1.3.1 Product Perspective**

CANS acts as a centralized, web-based platform connected to external systems via secure APIs. It bridges user input, facility data, and event scheduling through a unified, accessible interface.



*Figure 2: System Context Diagram*

#### **1.3.2 Product Functions**

Key functions of the system (categorized by user role) include:

* User authentication and dashboard access (all roles)
* Accessible route planning (Students)
* Event filtering by accessibility options (Students)
* Facility/event data management (Staff)
* Report resolution workflows (Staff, Admin)
* Role management, analytics, and system monitoring (Admins)



#### **1.3.3 User Characteristics**

|  |  |  |
| --- | --- | --- |
| User Role | Description | Expected Knowledge |
| Student | Uses system to plan routes, receive alerts, report issues, and find events. | Basic computer/mobile familiarity |
| Staff | Manages facility/event updates and issue resolution. | Moderate admin interface experience |
| Admin | Oversees system operations, roles, syncing, and reporting. | Advanced system and access management |

#### **1.3.4 Limitations**

* **Depends on external systems:** Real-time accuracy relies on Facilities/Event system updates.
* **Internet required:** CANS is cloud-based; offline use is not supported.
* **Campus-only scope:** Navigation limited to university premises.
* **Manual metadata updates:** Staff input determines data freshness.
* **Limited indoor precision:** Indoor pathfinding depends on future upgrades.
* **Device compatibility:** May not fully support outdated browsers/devices.

### **1.4 Definitions** A sample of defined terms (full list in Section 1.4 of original document):

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **CANS** | Campus Accessibility Navigation System |
| **WCAG 2.1** | Web Content Accessibility Guidelines for accessible UI design |
| **API** | Application Programming Interface for system integration |
| **Geolocation Data** | GPS/map-based coordinates for route and issue tracking |
| **RBAC** | Role-Based Access Control – limits access by user type |

### 

### **1.5 Apportioning of Requirements** **Phase 1: MVP (Core)** Includes real-time routing, alerts, login, RBAC, and basic reports.

### **Phase 2: Enhanced** Adds event filtering, photo-based reports, and mobile UI.

### **Phase 3: Advanced & Admin Tools** Supports full WCAG 2.1, integration APIs, analytics, and multilingual UI.

### *Future Work:* Indoor navigation, IoT integration, and AI-driven routing.

### **1.6 User Interface**

#### **1.6.1 Overview**

CANS features role-specific dashboards and an interactive map. Users receive disruption alerts, submit reports, and manage events through a modern, WCAG-compliant interface.

#### **1.6.2 UI Guidelines**

* **Accessibility:** WCAG 2.1 AA compliant, screen reader support, high contrast.
* **Consistency:** Fixed top navigation, branded theme, common icon sets.
* **Feedback:** Clear messages for errors, status changes, and actions.
* **Mobile-Ready:** Fully responsive layout and touch-friendly design.

### **1.7 External Interfaces**

* **Facilities API:** Provides real-time status updates on elevators, construction, etc. (JSON + OAuth 2.0).
* **Event Calendar API:** Imports events and accessibility metadata nightly; supports manual overrides.

## **2. References**

# **Referenced Standards, Guidelines, Tools & Technologies**

## **Standards & Guidelines**

* WCAG 2.1 (Web Content Accessibility Guidelines)

Description: International standard for web accessibility.

URL: <https://www.w3.org/TR/WCAG21/>

* GDPR (General Data Protection Regulation)

Description: EU regulation for data privacy and protection.

URL: <https://gdpr-info.eu/>

* CCPA (California Consumer Privacy Act)

Description: US regulation for consumer data rights.

URL: <https://oag.ca.gov/privacy/ccpa>

* AES Encryption Standard

Description: Specification for secure data encryption.

URL: <https://csrc.nist.gov/publications/detail/fips/197/final>

## **Tools & Technologies**

* Figma

Description: UI/UX design tool for prototyping the system interface.

URL: <https://www.figma.com/>

* MySQL

Description: Relational database management system (RDBMS) for storing user, facility, and event data.

URL: <https://www.mysql.com/>

* React.js

Description: Frontend framework for building the web application.

URL: <https://reactjs.org/>

* Node.js

Description: Backend runtime environment for server-side logic.

URL: <https://nodejs.org/>

## **3. Requirements**

### **3.1.1 Log in to the System**

|  |  |
| --- | --- |
| Use Case ID/Name | Log in to the System |
| Precondition | The user is registered in the system and has valid credentials. |
| Postcondition | The user is authenticated and redirected to their role-specific dashboard. |
| Main Flow | * The user opens the login page. * Enter username and password. * The system verifies credentials. * System redirects to user-specific interface (Student/Staff/Admin). |
| Alternate Flow | If users forget their password, they click "Forgot Password" and follow the recovery process. |
| Exception Flow | If the staff lacks permission or the record is locked, the system blocks the update and shows an appropriate message. |

### **3.1.2 Plan Accessible Route**

|  |  |
| --- | --- |
| Use Case ID/Name | Plan Accessible Routet |
| Precondition | The user is logged into the system. |
| Postcondition | A recommended accessible route is displayed on the map |
| Main Flow | * Student inputs current location and destination. * The system fetches real-time facility data (e.g., elevator status). * The system considers user preferences (e.g., avoid stairs). * The system displays the optimised accessible route on the map. |
| Alternate Flow | If real-time data is unavailable, the system uses static map data to generate a basic route. |
| Exception Flow | If route generation fails (e.g., no accessible path found), the system notifies the user with alternative suggestions (e.g., nearest help desk). |

### **3.1.3 Accessibility Issue Reporting**

|  |  |
| --- | --- |
| Use Case ID/Name | Manage Facility Data |
| Precondition | Student is logged in and at a valid reporting location. |
| Postcondition | The issue is submitted and appears in the staff dashboard for review. |
| Main Flow | * Student clicks “Report Issue.” * Uploads a photo, selects location (via GPS or map), and enters a description. * Submits the report. * The system logs the issue and notifies the responsible staff. |
| Alternate Flow | If GPS is unavailable, the student selects a location manually. |
| Exception Flow | If required fields are empty, the system prompts the user to complete them before submission. |

### **3.1.4 Facility and Event Management**

|  |  |
| --- | --- |
| Use Case ID/Name | Facility and Event Management |
| Precondition | The staff member is logged in with valid permissions. |
| Postcondition | Facility or event data is updated in the system. |
| Main Flow | * Staff navigates to the management dashboard. * Selects a facility (e.g., elevator) or event to update. * Enters new status or adds event metadata (e.g., ramps, interpreters). * Saves changes. * System updates the backend and refreshes user-facing information. |
| Alternate Flow | Staff uploads event details in bulk using a form or template. |
| Exception Flow | If submission fails (e.g., due to missing fields or network error), the system displays an error and logs the failure. |

### **3.1.5 Report Resolution Workflow**

|  |  |
| --- | --- |
| Use Case ID/Name | Report Resolution Workflow |
| Precondition | A reported issue exists in the system; a staff member or admin is logged in. |
| Postcondition | The issue is marked resolved or otherwise updated. |
| Main Flow | * Staff reviews new issue reports. * Verifies validity (onsite or via user input). * Marks the issue as “Resolved,” “In Progress,” or “Dismissed.” * The reporter receives a status update. |
| Alternate Flow | Admin oversees and audits the resolution history and status changes. |
| Exception Flow | If the staff lacks permission or the record is locked, the system blocks the update and shows an appropriate message. |

### **3.1.6 Event Status Alerting**

|  |  |
| --- | --- |
| Use Case ID/Name | Event Status Alerting |
| Precondition | Student is logged in; real-time event and facility data are available. |
| Postcondition | Alerts are displayed in the interface or sent as notifications. |
| Main Flow | * The system continuously monitors facility and event changes. * If a relevant event or obstruction occurs (e.g., elevator outage), it triggers an alert. * Student receives notification on dashboard or mobile push.. |
| Alternate Flow | The student views the alert history for previously triggered events. |
| Exception Flow | If alert delivery fails, the system logs the error and retries or prompts the user to refresh. |

### **3.1.7 Track Report Status**

|  |  |
| --- | --- |
| Use Case ID/Name | Track Report Status |
| Precondition | The student or staff has previously submitted or received an issue report. |
| Postcondition | The user can view the current status (e.g., pending, resolved). |
| Main Flow | * The user opens the “My Reports” section. * The system displays a list of submitted or assigned reports. * Each report shows the current status and update history. |
| Alternate Flow | User filters reports by status or type. |
| Exception Flow | If the report data fails to load, the system shows an error message and a retry option. |

### **3.1.8 Manage Event Information**

|  |  |
| --- | --- |
| Use Case ID/Name | Sync with External Systems |
| Precondition | Staff are authenticated and have permission to manage events. |
| Postcondition | Accessibility information for an event is updated or added. |
| Main Flow | * Staff access the event management interface. * Selects or creates an event. * Adds accessibility metadata (e.g., wheelchair access, sign language). * Submits changes. |
| Alternate Flow | Staff imports event data from the external calendar via sync. |
| Exception Flow | If metadata is missing or invalid, the system shows validation errors. |

### **3.1.9 Resolve Reported Issues**

|  |  |
| --- | --- |
| Use Case ID/Name | Resolve Reported Issues |
| Precondition | An issue has been submitted by a user and is pending. |
| Postcondition | The issue is resolved, and the status is updated. |
| Main Flow | * Staff review incoming reports in the dashboard. * Investigate the issue onsite or via details. * Takes action to resolve (e.g., repair, remove blockage). * Updates the report status. |
| Alternate Flow | Staff escalates the issue to the administrator or the maintenance unit. |
| Exception Flow | If staff lack the authority or resources to resolve, the system flags for escalation. |

### **3.1.10 Data Synchronization**

|  |  |
| --- | --- |
| Use Case ID/Name | Data Synchronization |
| Precondition | Admin is logged in, and the sync connection with external systems is available. |
| Postcondition | External facility or event data is updated in the system. |
| Main Flow | * Admin opens the sync interface. * Initiates manual or automatic sync. * System fetches and updates data from external APIs (e.g., facilities DB). * Status is logged. |
| Alternate Flow | The system runs a scheduled auto-sync and logs sync success/failure. |
| Exception Flow | If the API connection fails, the system retries or alerts the admin of the failed sync. |

### **3.1.11 User Role & Permission Management**

|  |  |
| --- | --- |
| Use Case ID/Name | User Role & Permission Management |
| Precondition | Admin is authenticated with configuration rights. |
| Postcondition | User roles and access levels are updated. |
| Main Flow | * Admin opens the user management panel. * View current users and roles. * Adds new users, assigns or changes roles (Student, Staff, Admin). * Saves configuration. |
| Alternate Flow | Admin imports the user list from a CSV or external system. |
| Exception Flow | If an invalid role is assigned, the system blocks and shows an error. |

### **3.1.12 System Monitoring & Reporting**

|  |  |
| --- | --- |
| Use Case ID/Name | System Monitoring & Reporting |
| Precondition | Admin is logged in. |
| Postcondition | Monitoring logs or analytical reports is generated. |
| Main Flow | * Admin opens the system analytics dashboard. * Selects report type (e.g., usage logs, issue stats). * Generates and exports a report. |
| Alternate Flow | Admin schedules recurring reports via system tools. |
| Exception Flow | If a data source is unavailable, the system shows “report generation failed” and logs an error. |

### **3.1.13 Multilingual and Accessible UI**

|  |  |
| --- | --- |
| Use Case ID/Name | Multilingual and Accessible UI |
| Precondition | The system is running, and the user is interacting with an interface. |
| Postcondition | Interface adjusts according to the user’s language and accessibility settings. |
| Main Flow | * The user selects the preferred language or accessibility mode. * System updates UI with selected settings (e.g., screen reader mode, high contrast). |
| Alternate Flow | Admin sets default accessibility settings per role or user group. |
| Exception Flow | If the selected language is unsupported, the system defaults to English and notifies the user. |

### **3.2 Performance Requirements** The system must work smoothly and efficiently for all users. Below are the key performance standards it must meet:

1. Speed of the System

Finding Routes: When a user asks for a route, the system should show it within 2 seconds most of the time.

Alerts: If an elevator breaks down or a path is blocked, users should get a notification within 10 seconds.

Reporting Issues: When a user submits a problem (like a blocked ramp), the system should save it within 5 seconds, even with photos.

1. Handling Many Users at Once

The system should work without slowing down even when 1,000+ people use it at the same time (e.g., during busy hours like mornings).

1. Always Available

The system should be up and running 99.5% of the time (24/7), except during planned maintenance.

1. Managing Updates

The system should handle 100+ updates per second (e.g., elevator status changes) without delays.

1. Easy to Use on All Devices

The website should load quickly on phones, tablets, and computers (within 3 seconds).

1. Accuracy

The map should show the user’s location within 5 meters outdoors to help them find the best route.

1. Reports for Admins

When administrators generate reports (e.g., monthly usage stats), it should take no longer than 30 seconds.

1. Recovery from Problems

If the system crashes, it should restart and work normally within 5 minutes.

1. No Overloads

The servers should never use more than 80% of their power, even during busy times, to avoid crashes.

### **3.3 Usability Requirements**

The system will attempt to meet the following usability criteria to ensure efficiency, learnability, and satisfaction for all user roles:

* **Learnability:**
  + New users should be able to perform core tasks (e.g., plan a route, report an issue) within 5 minutes without prior training.
  + Contextual tooltips and an optional guided tutorial may be provided for first-time users.
* **Efficiency:**
  + Frequently carried out tasks (e.g., route planning) should require less than 3 clicks from the homepage.
  + Autocomplete for location inputs (e.g., "Lib" → "Library Building").
* **Error Handling:**
  + Form validation with inline feedback to prevent submission errors (e.g., "Photo required for issue reports").
  + Undo options should be provided for critical actions to prevent unintentional accidents (e.g., accidental report deletion).
* **Satisfaction:**
  + Post-release surveys should achieve roughly ≥ 90% positive feedback on ease of use.

### **3.4 Interface Requirements**

* **Hardware Interfaces:**
  + Compatible with touchscreens (mobile/tablet) and mouse/keyboard (desktop) input.
  + GPS support for outdoor location tracking (estimate of around 5m accuracy).
* **Software Interfaces:**
  + Frontend: React.js (desktop/mobile browsers).
  + Backend: Node.js APIs for data processing and business logic.
  + Database: MySQL with read replicas for scalability.
* **Communication Interfaces:**
  + HTTPS for all client-server communication, encrypted via SSL/TLS.
  + WebSocket for real-time alerts (e.g., push notifications).

# 3.5 Logical Database Requirements

3.5.1 Entity-Relationship Diagram

### The Entity-Relationship Diagram (ERD) for the Campus Accessibility Navigation System (CANS) provides a structured database model to manage accessibility-related data efficiently. The design includes core entities such as Users, Facilities, Events, Reports, Notifications, and Permissions, linked through defined relationships to ensure data integrity and streamlined operations. The User entity is categorized into Students, Staff, and Admins, enabling role-based access control and functionality. Users interact with Reports to document accessibility issues and with Events to track participation in campus activities. Facilities are managed by staff, with updates triggering Notifications to inform users of relevant changes. The Permissions entity enforces security by regulating access based on user roles. Primary and foreign keys maintain relational integrity, supporting efficient data retrieval and consistency. This ERD framework facilitates real-time navigation, event management, facility oversight, and user engagement, ensuring a cohesive and secure system for accessibility management.

|  |
| --- |
|  |
| **ERD** |

3.5.2 Relationship

The Entity-Relationship Diagram (ERD) establishes the database structure for efficient accessibility data management, with the User entity serving as the central component connected to Student, Staff, and Admin roles through one-to-one (1:1) relationships for authentication and role differentiation. Users engage with the system by submitting Reports through a one-to-many (1:M) relationship and managing Events via another 1:M relationship, while the Event Participation entity tracks attendee data in a separate 1:M relationship. The Facility entity maintains a many-to-one (M:1) relationship with Staff for management purposes and generates Notifications through a 1:M relationship to alert users, who receive these alerts in another 1:M configuration. The Permissions entity enforces system security by regulating access through a 1:M relationship. Primary and foreign keys ensure data integrity, enabling seamless functionality across facility management, event coordination, report handling, and user authentication within the accessibility framework.

## 3.5.3 Data Dictionary

3.5.3.1 User Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| user\_id | INTEGER | Unique identifier for each user | PRIMARY KEY, NOT NULL, UNIQUE |
| email | VARCHAR(225) | User email for authentication | UNIQUE, NOT NULL |
| password | VARCHAR(225) | Encrypted password for authentication | NOT NULL |
| role | ENUM | Defines user type | NOT NULL |

3.5.3.2 Student Table

| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| --- | --- | --- | --- |
| student\_id | INTEGER | Unique identifier for each student | PRIMARY KEY, NOT NULL, UNIQUE |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), UNIQUE |
| enrolment\_number | VARCHAR(50) | Student’s registration number | UNIQUE, NOT NULL |

3.5.3.2 Staff Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| stuff\_id | INTEGER | Unique identifier for each staff membe | PRIMARY KEY, NOT NULL, UNIQUE |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), UNIQUE |
| department | VARCHAR(100) | Staff department or assigned unit | NOT NULL |

3.5.3.3 Admin Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| admin\_id | INTEGER | Unique identifier for each admin | PRIMARY KEY, NOT NULL, UNIQUE |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), UNIQUE |
| permissions | TEXT | Admin-level access rights | NOT NULL |

3.5.3.4 Report Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| report\_id | INTEGER | Unique identifier for each report | PRIMARY KEY, NOT NULL, UNIQUE |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), NOT NULL |
| description | TEXT | Details of accessibility issue | NOT NULL |
| status | ENUM | Current report status | ENUM('Pending', 'Resolved', 'In-Progress') NOT NULL |
| created\_at | TIMESTAMP | Timestamp when the report was submitted | DEFAULT CURRENT\_TIMESTAMP |

3.5.3.5 Event Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| event\_id | INTEGER | Unique identifier for each event | PRIMARY KEY, NOT NULL, UNIQUE |
| organizer\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), NOT NULL |
| facility\_id | INTEGER | Foreign key linking to Facility table | FOREIGN KEY REFERENCES Facility(facility\_id), NOT NULL |
| event\_name | VARCHAR(255) | Title or name of the event | NOT NULL |
| event\_date | DATE | Scheduled date for the event | NOT NULL |

3.5.3.6 Event Participation Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| participation\_id | INTEGER | Unique identifier for event attendance | PRIMARY KEY, NOT NULL, UNIQUE |
| event\_id | INTEGER | Foreign key linking to Event table | FOREIGN KEY REFERENCES Event(event\_id), NOT NULL |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), NOT NULL |
| registration\_status | ENUM | Attendee status | ENUM('Confirmed', 'Waitlisted') NOT NULL |

3.5.3.7 Facility Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| facility\_id | INTEGER | Unique identifier for each facility | PRIMARY KEY, NOT NULL, UNIQUE |
| facility\_name | VARCHAR(255) | Name of the facility | NOT NULL |
| type | ENUM | Facility type (Elevator, Ramp, Room, Pathway) | ENUM('Elevator', 'Ramp', 'Room', 'Pathway') NOT NULL |
| status | ENUM | Current facility operational status | ENUM('Operational', 'Out-of-Service') NOT NULL |
| last\_updated | TIMESTAMP | Timestamp for last update | DEFAULT CURRENT\_TIMESTAMP |
| managed\_by\_staff\_id | INTEGER | Foreign key linking to Staff table | FOREIGN KEY REFERENCES Staff(staff\_id), NOT NULL |

3.5.3.8 Notification Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| notification\_id | INTEGER | Unique identifier for each notification | PRIMARY KEY, NOT NULL, UNIQUE |
| facility\_id | INTEGER | Foreign key linking to Facility table | FOREIGN KEY REFERENCES Facility(facility\_id), NOT NULL |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), NOT NULL |
| message | TEXT | Notification content | NOT NULL |
| timestamp | TIMESTAMP | Timestamp when notification was sent | DEFAULT CURRENT\_TIMESTAMP |

3.5.3.9 Permission Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** | ***Constraints*** |
| permission\_id | INTEGER | Unique identifier for access permissions | PRIMARY KEY, NOT NULL, UNIQUE |
| user\_id | INTEGER | Foreign key linking to User table | FOREIGN KEY REFERENCES User(user\_id), NOT NULL |
| access\_level | ENUM | Defines access rights | ENUM('View', 'Edit', 'Manage') NOT NULL |

## 3.6 Design Constraints

The Design Constraints Table outlines the key limitations and requirements for the development of the Campus Accessibility Navigation System (CANS). The constraints are organized into technical, environmental, and regulatory categories to guide system development.

Technical constraints include database scalability, real-time notifications, role-based access control, and security encryption. Environmental constraints cover device compatibility, network limitations, and accessibility compliance. Regulatory constraints involve data privacy laws, facility accessibility standards, and legal record-keeping requirements.

Each constraint is assigned a priority level to guide development focus. These constraints ensure the system meets performance, usability, and compliance standards while remaining scalable and secure.

|  |  |  |  |
| --- | --- | --- | --- |
| ***ID*** | ***Constraint*** | ***Description*** | ***Priority*** |
| ***REQ\_DC001*** | Database Scalability | The system must efficiently handle large volumes of users, reports, events, and notifications while ensuring fast data retrieval and query execution. Indexing and optimized relational mapping should be implemented to maintain performance. | High |
| ***REQ\_DC002*** | Real-time Notifications | Facility updates must trigger immediate notifications to affected users, ensuring timely awareness of accessibility changes. Asynchronous processing and message queuing may be required for scalability. | High |
| ***REQ\_DC003*** | Role-Based Access Control (RBAC) | Role-based permissions must be enforced to control data access for Students, Staff, and Admins, preventing unauthorized actions while ensuring a seamless user experience. | High |
| ***REQ\_DC004*** | Security & Data Protection | Sensitive user data, including authentication credentials and reports, must be encrypted and stored securely. Secure protocols such as **AES encryption**, **SSL/TLS** for data transmission, and regular security audits should be enforced. | High |
| ***REQ\_DC005*** | Multi-Platform Compatibility | The system must be accessible across different platforms, including desktops, mobile devices, and tablets, with responsive UI design and adaptive components for various screen resolutions. | Medium |
| ***REQ\_DC006*** | Accessibility Compliance | The user interface must comply with **WCAG 2.1 standards**, ensuring usability for individuals with disabilities through features like screen reader compatibility, keyboard navigation, and adjustable contrast settings. | High |
| ***REQ\_DC007*** | Device & Network Limitations | The system should function on low-end devices and operate efficiently in low-bandwidth environments by optimizing resource usage and minimizing heavy processing requirements. | Medium |
| ***REQ\_DC008*** | Localization & Language Support | Multi-language functionality should be supported based on user demographics, ensuring accessibility for a diverse audience. | Low |
| ***REQ\_DC009*** | Server Infrastructure | The hosting environment must be optimized for **scalability**, **load balancing**, and **failover mechanisms** to maintain system stability during peak usage periods. Cloud-based solutions should be considered for reliability. | High |
| ***REQ\_DC010*** | Data Privacy & Compliance | The system must comply with global and local data protection laws, including **GDPR**, **CCPA**, and institutional guidelines to protect user privacy and prevent unauthorized data exposure. | High |
| ***REQ\_DC011*** | Public Facility Regulations | The system must align with government and institutional accessibility policies to ensure proper tracking of infrastructure updates. | High |
| ***REQ\_DC012*** | Legal Record-Keeping | Reports and facility updates must be stored securely with **audit logs**, ensuring traceability and preventing data manipulation to maintain accountability. | High |
| ***REQ\_DC013*** | Copyright & Intellectual Property | The system must enforce copyright compliance, ensuring that all stored and shared content adheres to legal regulations and ethical usage policies. | Medium |

### **3.7 Software System Attributes** These attributes define the overall quality and operational characteristics of the Campus Accessibility Navigation System (CANS):

1. Reliability

The system shall operate consistently under normal use, with ≤ 1 critical failure per month.

Real-time updates (e.g., elevator statuses, route changes) must be accurate and reflect the latest facility data.

User-submitted reports shall be stored without loss of data (e.g., photos, geolocation).

2. Availability

The system shall be accessible 24/7, with scheduled maintenance limited to ≤ 2 hours per month.

During peak hours (8 AM–5 PM), uptime shall remain ≥ 99.5%.

3. Security

All user data (passwords, reports, preferences) shall be encrypted using AES-256 encryption.

Role-based access control (RBAC) shall prevent unauthorized users from accessing sensitive features (e.g., staff dashboards).

4. Maintainability

The codebase shall be modular, with clear documentation for future updates

Database backups shall be performed daily to enable quick recovery from failures.

The project shall use Git-based version control to track changes, facilitate team collaboration, and ensure rollback capability if needed.

Continuous Integration (CI) tools (e.g., GitHub Actions) shall be used to automate testing and deployment pipelines, ensuring that new code can be integrated and deployed efficiently with minimal disruption.5. Portability

The web application shall function on Chrome, Firefox, Safari, and Edge browsers (latest stable versions).

Mobile responsiveness shall ensure usability on devices with screen sizes ≥ 5 inches.

6. Compliance

The UI shall fully comply with WCAG 2.1 AA standards for accessibility.

Data handling shall adhere to GDPR and CCPA regulations for user privacy.

7. Scalability

The system shall support a 20% annual increase in users without requiring architectural changes.

### 

### **3.8 Supporting Information**

Balancing "Must-Have" Minorities: This section provides context for the system’s requirements based on the Kano Model survey conducted during the requirements elicitation phase. The results explain how user feedback shaped the prioritization and categorization of features, even though most responses fell into the "Indifferent" category.

## **1. Elicitation Methodology**

* Technique:

A Kano Model survey was distributed via Google Forms to gather user preferences for accessibility and navigation features.

* Execution:

Survey Link: Shared via class WhatsApp groups and social media.

Duration: Open for 2 weeks (May 7–21, 2025).

* Participants:

Total submissions: 45

Valid responses (complete Kano pairs): 30

* Analysis:

A Python script categorized responses into Kano categories (Must-Have, Attractive, Indifferent, Reverse, etc.).

## **2. Survey Results and Requirement Mapping**

The table below summarizes the Kano categorization of features and their linkage to system requirements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Kano Category | Counts | Linked Requirements | Interpretation |
| Accessible routes across campus | Indifferent | I:25, A:15, E:1, M:3 | RQ-01, FN-02 | Neutral reception; prioritized due to institutional accessibility mandates. |
| Real-time elevator outage alerts | Indifferent | I:23, A:14, R:2, M:5 | RQ-02, FN-03 | Some users expect alerts (M), but most indifferent. Critical for compliance. |
| Live construction/path information | Indifferent | I:21, A:17, R:2, M:4 | RQ-02, FN-03 | Helpful for safety; retained despite neutral feedback. |
| Event-based rerouting | Indifferent | I:21, A:14, M:7, R:1 | RQ-04, RQ-09 | Notable "Must-Have" (M) subgroup justified inclusion. |
| Speech/screen reader support | Indifferent | I:27, A:14, M:3 | RQ-14 | Basic accessibility expectation; required for WCAG 2.1 compliance. |

Key: I: Indifferent, A: Attractive, M: Must-Have, R: Reverse, E: Excitement

## **3. Implications for System Design**

* Indifferent Features as Requirements:

Despite neutral user responses, features like accessible routes (RQ-01) and screen reader support (RQ-14) were prioritized to comply with university accessibility policies and WCAG 2.1 standards.

* Although staff/admin feedback was limited, we included features based on institutional accessibility standards and WCAG 2.1 compliance expectations.

Features with notable "Must-Have" votes (e.g., event-based rerouting) were prioritized to serve niche but critical user needs.

## **4. Limitations of Elicitation**

• Sample Bias: Most respondents were tech-savvy students; staff and visitors were underrepresented.

• Scope: The survey focused on usability, not technical feasibility (e.g., API integration challenges in RQ-11).

## **5. Supporting Artifacts**

• Raw Data: Survey responses stored in a CSV file (see repository).

• Python Script: Automated Kano categorization tool (shared with the development team).

• Survey Design: can be seen from the link <https://docs.google.com/forms/d/e/1FAIpQLSc9RD4jTCL5mbspPtAizovIrXu8ecASnML_RfMZHT-r71SQIw/viewform?usp=header>

### 

### 

## 

## **4. Verification**

### **4.1 Functions Verification**

The **Campus Accessibility Navigation System (CANS)** includes a range of user-facing and administrative functions that must be verified to ensure they meet the defined requirements and support real-world usage by students, staff, and administrators. This section outlines how each major function will be verified through interaction testing, role-based checks, and validation of expected behavior.

Each function corresponds to one or more use cases and requirement IDs defined earlier. Verification focuses on ensuring the correctness, completeness, and usability of system features as experienced by all types of users.

### **4.2 Performance Verification**

This section outlines the procedures to verify that the Campus Accessibility Navigation System (CANS) meets its performance requirements. Each requirement from Section 3.2 is mapped to a verification method, success criteria, and tools used for testing.

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Verification Method | Tools/Techniques | Success Criteria |
| Route generation ≤ 2 seconds | Simulate 500 concurrent users requesting routes; measure average response time. | JMeter, Postman | 95% of requests complete within 2 seconds. |
| Alerts ≤ 10 seconds | Trigger an elevator outage and measure time until the alert appears on user devices. | Selenium, Manual Testing | Alert delivered within 10 seconds to all affected users. |
| Report submission ≤ 5 seconds | Submit 100 reports with photos simultaneously; track processing time. | JMeter, AWS LoadRunner | All reports saved within 5 seconds. |
| Handle 1,000+ concurrent users | Simulate 1,200 users accessing the system during peak hours (8 AM–5 PM). | LoadNinja, BlazeMeter | No errors or crashes; response times remain within acceptable thresholds. |
| 99.5% uptime | Monitor system availability over 30 days using uptime tracking tools. | Nagios, UptimeRobot | System available ≥ 99.5% of the time, excluding scheduled maintenance. |
| 100+ updates/second | Send 150 updates/second to the Facilities API and monitor processing latency. | Postman, AWS Lambda | API processes ≥ 100 updates/second with ≤ 500ms latency. |
| Load time ≤ 3 seconds | Test page load times on mobile (Chrome, Safari) and desktop (Firefox, Edge) browsers. | Google Lighthouse, WebPageTest | All pages load within 3 seconds on devices with ≥ 5-inch screens. |
| Geolocation ≤ 5m accuracy | Compare system-reported user location with GPS benchmarks in outdoor campus areas. | GPS Visualizer, Manual Testing | Location accuracy within 5 meters for 95% of tests. |
| Admin reports ≤ 30 seconds | Generate a usage report covering 6 months of data; measure processing time. | MySQL Workbench, Custom Scripts | Report generated within 30 seconds. |
| Recovery ≤ 5 minutes | Simulate a server crash and measure time to restore functionality. | AWS CloudWatch, Manual Testing | System fully operational within 5 minutes. |
| Server capacity ≤ 80% | Monitor CPU/memory usage during peak load (1,200 users). | New Relic, Datadog | Server resource usage never exceeds 80%. |

## **Verification Process**

Test Environment:  
  
 • Use cloud servers (AWS EC2) to replicate production conditions.  
 • Simulate campus Wi-Fi speeds (50 Mbps download/20 Mbps upload).

Data Collection:  
  
 • Log response times, errors, and resource usage during tests.  
 • Validate geolocation accuracy using campus landmarks (e.g., library, cafeteria).

Reporting:  
  
 • Document results in a verification report, highlighting any deviations from success criteria.  
 • Retest failed scenarios after fixes.

## **Pass/Fail Criteria**

Pass: All success criteria are met for ≥ 95% of test cases.  
  
Fail: Any critical requirement (e.g., uptime, route generation) fails more than 5% of tests.

### 

### **4.3 Usability Verification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Verification Method** | **Success Criteria** | **Possible Tools** |
| Learnability | Observe 10 new users performing core tasks. | 90% complete tasks within 5 minutes. | UserTesting.com |
| Efficiency | Log clicks for 50 route-planning sessions. | ≤ 3 clicks to generate a route. | Hotjar, Google Analytics |
| Error Recovery | Simulate 20 erroneous form submissions. | 100% receive actionable error messages. | Jest, Selenium |
| Satisfaction | Post-release survey (Likert scale 1–5). | ≥ 4.0 average satisfaction score. | SurveyMonkey |

### 

### **4.4 Interface Verification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Test** | **Success Criteria** | **Possible Tools** |
| Facilities API | Mock outage data → validate alert triggers. | Alerts appear within 10 seconds. | Postman, Jest |
| Event Calendar Sync | Compare iCal feed with CANS event listings. | 100% data parity after sync. | Python scripts |
| GPS Accuracy | Test at 10 campus landmarks. | Location accuracy ≤ 5m. | Google Maps API |
| Cross-Browser Support | Render UI on Chrome, Safari, Firefox, Edge. | Consistent functionality | BrowserStack |

**Verification Process:**

* Test Environment: AWS cloud servers replicating production.
* Pass/Fail: ≥ 95% success rate for all criteria.

### 

### **4.5 Logical Database Verification**

The logical database verification process involves systematic validation activities to ensure the database design meets specified system requirements and stakeholder expectations. These verification procedures focus on confirming proper database normalization to eliminate redundancy and maintain structural efficiency. Indexing strategies are evaluated to optimize query performance and data retrieval operations. Relationship integrity checks validate the correct implementation of primary and foreign key constraints across database tables. Security protocols are reviewed to confirm appropriate data protection measures are in place. These verification activities collectively ensure the database maintains data integrity, supports consistent operations, and delivers optimal performance. The process contributes to the system's overall reliability, scalability, and ability to effectively manage user interactions, content organization, and real-time data processing requirements.

4.5.1 Verification Procedures

|  |  |
| --- | --- |
| ***Verification Step*** | ***Description*** |
| ***Schema Validation*** | Ensures all tables, primary keys, and foreign keys are properly defined according to relational database principles |
| ***Referential Integrity*** | Confirms that all foreign key constraints correctly enforce relationships and prevent orphaned records. |
| ***Normalization Check*** | Validates that the schema follows **normalization rules (up to 3NF)** to minimize data redundancy and ensure efficient data storage. |
| ***Data Type Accuracy*** | Verifies that each column has the appropriate data type to maintain data consistency and prevent processing errors. |
| ***Unique & Not Null Constraints*** | Ensures primary keys are **unique**, and mandatory fields are enforced using **NOT NULL constraints**. |
| ***Index Optimization*** | Confirms that indexes are correctly applied to frequently queried fields, improving database query efficiency |
| ***Relationship Consistency*** | Verifies that **one-to-many (1:M), many-to-one (M:1), and one-to-one (1:1) relationships** between entities align with the expected system behavior. |
| ***Security & Access Controls*** | Validates that **role-based permissions** are correctly implemented to prevent unauthorized access and protect sensitive data. |

### 

### **4.6 Design Constraints Verification**

The Design Constraints Verification process outlines the methods for validating each constraint identified in [**section 3.6**](#_o1hjz2g7vfww) throughout system development. This verification ensures compliance with technical specifications, environmental requirements, and regulatory standards while confirming the system's functionality and performance capabilities. For technical constraints, verification will involve testing database scalability, access control mechanisms, and security protocols. Environmental constraints will be validated through compatibility testing across devices and network conditions, along with accessibility compliance checks. Regulatory requirements will be verified through documentation reviews and compliance audits. These verification activities serve to objectively demonstrate that all design constraints have been properly implemented and that the final system meets both technical specifications and stakeholder expectations across all constraint categories.

|  |  |  |
| --- | --- | --- |
| ***ID*** | ***Constaint*** | ***Verification method*** |
| ***REQ\_DC001*** | Database Scalability | Load testing and stress tests will be conducted to measure the system's ability to handle increasing data volumes efficiently |
| ***REQ\_DC002*** | Real-Time Notifications | System stress testing and event simulations will confirm timely notification delivery without delays. |
| ***REQ\_DC003*** | Role-Based Access Control (RBAC) | Unit testing will validate user permissions by ensuring restricted access for different roles |
| ***REQ\_DC004*** | Security & Data Protection | Security audits, penetration testing, and encryption validation will ensure data is protected against unauthorized access. |
| ***REQ\_DC005*** | Multi-Platform Compatibility | Cross-device testing across desktop, mobile, and tablet interfaces will verify UI adaptability and performance. |
| ***REQ\_DC006*** | Accessibility Compliance | WCAG 2.1 compliance testing will ensure usability for individuals with disabilities, including screen reader and keyboard navigation tests. |
| ***REQ\_DC007*** | Device & Network Limitations | Performance testing on low-end devices and simulations in low-bandwidth environments will verify system efficiency |
| ***REQ\_DC008*** | Localization & Language Support | Language selection testing and UI validation will ensure accurate translations and proper layout adjustments. |
| ***REQ\_DC009*** | Server Infrastructure | Load balancing and failover testing will validate system reliability during peak traffic and hardware failures. |
| ***REQ\_DC010*** | Data Privacy & Compliance | GDPR and data protection compliance audits will be performed to ensure secure user data handling. |
| ***REQ\_DC011*** | Public Facility Regulations | Institutional review and accessibility policy validation will confirm adherence to facility management standards. |
| ***REQ\_DC011*** | Legal Record-Keeping | Automated logging and integrity tests will verify accurate storage of reports and facility updates for auditing purposes. |
| ***REQ\_DC012*** | Copyright & Intellectual Property | Legal review of digital content storage and distribution policies will ensure compliance with copyright regulations. |

### **4.7 Software System Attributes Verification**

This section describes how the **quality attributes** of the Campus Accessibility Navigation System (CANS) will be verified to ensure that the system meets operational, performance, and compliance expectations.

|  |  |  |
| --- | --- | --- |
| Attribute | Verification Method | Success Criteria |
| Reliability | Conduct continuous usage simulations under normal and peak load conditions. | ≤ 1 critical failure per month; data (e.g., reports, routes) saved without loss. |
| Availability | Monitor system uptime via tools (e.g., UptimeRobot, Nagios) over a 30-day window. | ≥ 99.5% uptime (excluding ≤ 2 hours/month for scheduled maintenance). |
| Security | Perform penetration testing, encryption validation, and access control testing. | AES-256 encryption, secure password storage, no unauthorised access to any feature. |
| Maintainability | Review source code modularity and documentation; evaluate backup systems. | Clear module boundaries; daily backups confirmed; future updates supported easily. |
| Portability | Conduct compatibility testing across latest versions of Chrome, Safari, Edge, and Firefox on various devices. | The system functions properly on all tested devices and browsers. |
| Compliance | Validate system against WCAG 2.1 AA accessibility standards and GDPR/CCPA data handling guidelines | Full WCAG 2.1 AA compliance; no unencrypted user data; user privacy settings active. |
| Scalability | Simulate 20% annual user growth and measure system performance and resource usage. | System performs smoothly without architectural changes or performance degradation. |

### **4.8 Supporting Information Verification**

This section outlines how the requirements elicited using the **Kano Model** were validated and mapped to system functions, ensuring that stakeholder feedback shaped the final product design-even when responses indicated mixed priorities.

**4.8.1 Verification of Elicitation Validity**

* **Technique**: Kano Model-based survey
* **Execution**: Google Forms
* **Cross-validation**: Mapped Kano categories to corresponding requirements (e.g., RQ-01, FN-02).

**4.8.2 Mapping Survey to System Requirements**

|  |  |  |
| --- | --- | --- |
| Feature | Kano Category | Design Justification |
| Accessible routes across campus | Mostly Indifferent | Prioritized for compliance with accessibility standards. |
| Elevator outage alerts | Indifferent / Must-Have | Included due to compliance needs and safety concerns. |
| Construction/path disruptions | Indifferent / Attractive | Important for accurate real-time routing. |
| Event accessibility filtering | Some Must-Have responses | Retained to support inclusive campus event participation. |
| Screen reader/speech support | Indifferent / Must-Have | Required for WCAG 2.1 compliance and inclusive design. |

**4.8.3 Elicitation Gaps & Adjustments**

* **Underrepresented Roles: Staff and visitors were underrepresented in the survey.**
* **Design Adjustment: Despite neutral responses, features were retained to meet university policy and legal requirements (e.g., WCAG, GDPR).**

## **5. Appendices**

### **5.1 Assumptions and Dependencies**

- All users have access to internet-enabled devices (smartphones, tablets, laptops).

- The university facilities and event databases provide accurate and timely data via APIs.

- Users will report accessibility issues honestly and with relevant evidence (e.g., photos).

- The university administration supports and enforces the use of the system among departments.

- Integration with external systems: University Facilities Management System and Event Calendar API.

- Real-time data availability from third-party systems (e.g., elevator status).

- Availability of accurate campus map data including accessibility metadata.

- Continuous collaboration from IT and Accessibility departments during development and deployment.

### **5.2 Acronyms and Abbreviations**

Refer to [Section 1.4](#_wo04108p9uxg) – Definitions – for the full list of acronyms and abbreviations used in this document.