Software Architecture and Design Specification

Project: Helping hands Software

Version: 1.0.0.0

Authors: Kushal Kumar G, Kirthan G, K Sreekar, K sriharsha

Date: 06-10-2025

Status: Draft

Revision History

Version: 1.0.0.0

Date: 06-10-2025

Author: Kushal Kumar G, Kirthan G, K Sreekar, K Sriharsha

Change Summary: To Be Filled in Future

Approvals

Role: Faculty

Name: Ananda M S

Signature/Date:

1. Introduction

1.1 Purpose

This system is intended to provide a structured platform for company employees, who are not regular mart workers, to volunteer during weekends and festivals when customer footfall is high. It enables staff and managers to book slots in nearby marts, choose preferred sections, and earn Helping Hands Points (HHP) for their contributions.

1.2 Scope

The scope of the **Helping Hands Software (HHS)** is to provide a centralized, web-based platform that connects company employees with marts requiring additional support during weekends and festivals. The system allows staff and managers to search nearby marts, select preferred sections, and book time slots based on availability. It manages waitlists, cancellations, and booking priorities using a First-Come-First-Serve (FCFS) approach. Store admins can define store details, set manpower requirements, and monitor attendance, while the main admin oversees system-wide management, including user and store

registration. The software also awards Helping Hands Points (HHP) to employees for completed slots, ensuring fairness, motivation, and performance tracking across all participants.

1.3 Audience
End Users (Staff, Branch Managers, and Company Employees), Store Admins, Main Admin

Secondary Stakeholders:

- **Developers and Testers**
- Management Team

- 1.4 Definitions
  Helping Hands (HH): An initiative where company employees volunteer to assist marts during weekends and festivals.
- **Helping Hands Software (HHS):** The system that manages slot booking, store requirements, and performance tracking under the Helping Hands initiative.
- **Helping Hands Points (HHP):** Reward points awarded to users based on the number of hours worked in their booked slots, used to monitor employee contribution.
- **Mart:** A store location (mall, supermarket, electronics store, or other retail outlet) where volunteers can book slots.
- **Slot:** A specific time duration (date and hours) reserved by a user to work in a mart.
- **Section:** A defined area within a mart (e.g., billing, groceries, electronics, customer service) where volunteers can choose to work.
- **Waitlist:** A queue where users are placed if their preferred slot/section is already full.
- **FCFS (First Come First Serve):** The scheduling principle used to allocate slots and waitlist positions.
- **End User:** A company employee (staff, branch manager, or other employee) volunteering through the system.
- **Store Admin:** A role responsible for managing mart details, manpower requirements, attendance, and bookings.
- **Main Admin:** The central authority responsible for registering/removing users and marts, viewing performance, and ensuring smooth system operation.

2. Document Overview

2.1 How to use this document
Provides architectural deliverables including UML diagrams, ADRs, threat models, and API design.

2.2 Related Documents
SRS, STP, RTM.

3. Architecture

3.1 Goals & Constraints

Goals

1. Provide a user-friendly platform for company employees to volunteer in marts during peak times.
2. Enable easy booking, cancellation, and waitlist management of slots.
3. Support role-based access for Users, Store Admins, and Main Admin.
4. Track and reward employee contributions using Helping Hands Points (HHP).
5. Ensure data privacy, security, and reliable notifications (SMS/Email).
6. Maintain scalability and performance during weekends and festivals.

Constraints

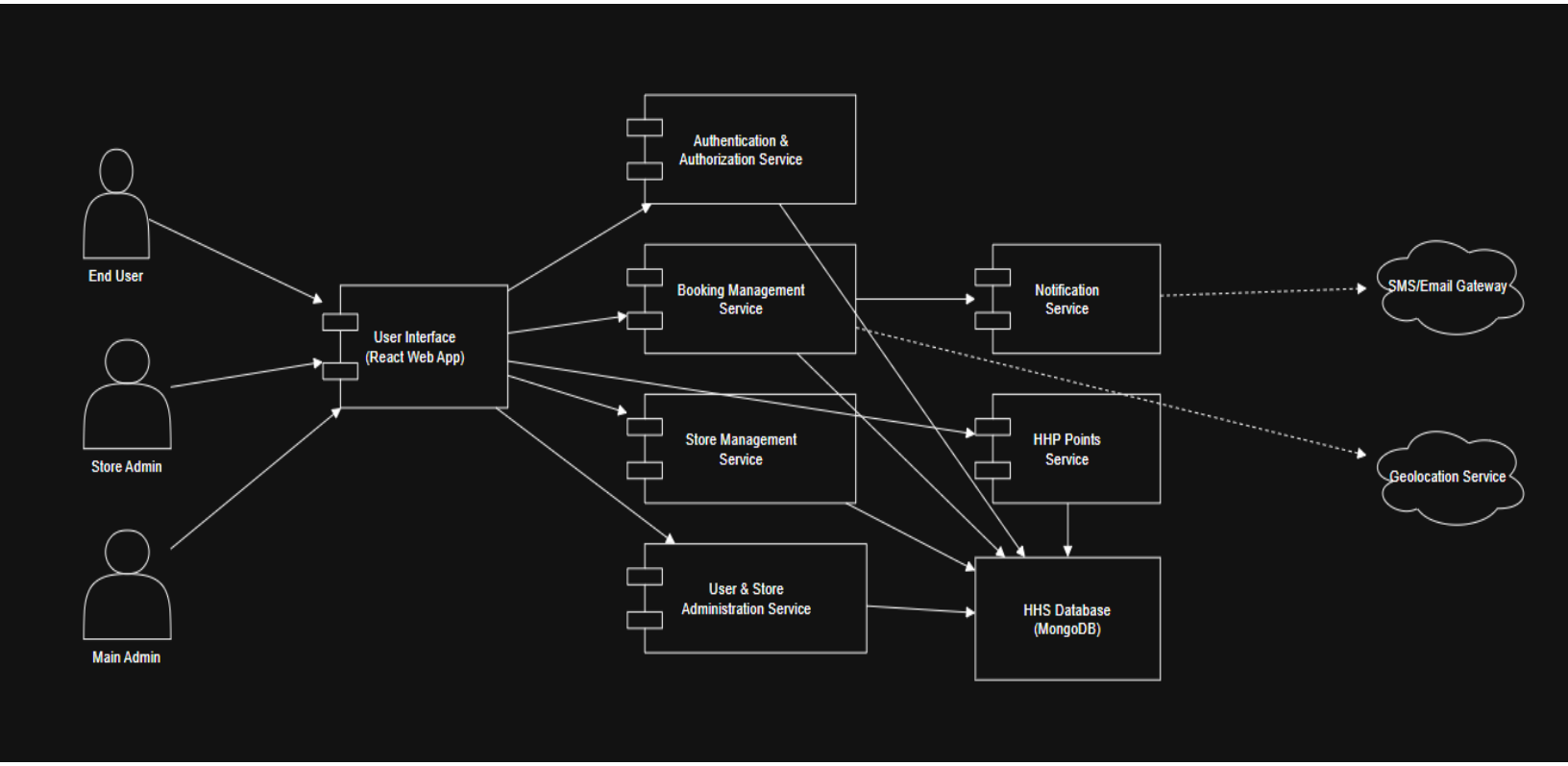
1. Users can cancel a maximum of **two confirmed slots per day**.
2. Slot allocation follows **First Come, First Serve (FCFS)** priority.
3. Location services are **only for mart search** and require user consent.
4. System must remain available with **≥99.5% uptime** on weekends/festivals.
5. Database must support **regular backups and recovery**.
6. Interfaces must be **web-based and mobile responsive**.

3.2 Stakeholders & Concerns

Customers: End Users (Staff, Branch Managers), Store Admin, Main Admin, Developers, Testers, management

Bank Ops:
Regulators:
Developers:

3.3 Component (UML) Diagram



3.4 Component Descriptions

AD.1. User Interface Component

- Description: Provides the front-end interface for Users, Store Admins, and Main Admins.

AD.2. Authentication & Authorization Component

- Description: Ensures secure login and role-based access.

AD.3. Booking Management Component

- Description: Manages all slot booking operations.

AD.4. Store Management Component

- Description: Used by Store Admins to manage store-related details.

AD.5. User & Store Administration Component

- Description: Centralized control for the Main Admin.

AD.6. Notification Component

- Description: Handles user alerts and reminders.

AD.7. HHP (Helping Hands Points) Component

- Description: Calculates and updates HHP points based on attendance.

AD.8. Database Component

- Description: Central storage system for all data.

3.5 Chosen Architecture Pattern and Rationale

Architecture Pattern: Three-Tier Architecture (Presentation, Application, Data Layers)

The Helping Hands Software (HHS) will adopt a **three-tier architecture pattern** consisting of:

1. **Presentation Layer (Client UI):**
    - Provides the web-based interface for Users, Store Admins, and Main Admins.

- Implements responsive design for desktop and mobile devices.

2. **Application Layer (Backend Services):**
    - Handles business logic such as slot booking, waitlist management, HHP calculation, user/store administration, and notifications.
    - Implements secure authentication and authorization using role-based access control.
3. **Data Layer (Database):**
    - Stores persistent data such as user profiles, mart details, slots, attendance, and HHP points.
    - Ensures secure and reliable storage with backup and recovery mechanisms.

3.6 Technology Stack & Data Stores

Frontend: React.js, HTML5, CSS3, JavaScript
Backend: Node.js, Express.js, SMS/Email API, Geolocation API
Database: MongoDB
Tools & Environment: Git, GitHub

3.7 Risks & Mitigations

1. Risk: Unauthorized access to sensitive data.
    - Mitigation: Implement RBAC, secure authentication (JWT), and encryption.
2. Risk: Data loss due to system or hardware failure.
    - Mitigation: Regular database backups and disaster recovery plan.
3. Risk: Server overload during peak times (weekends/festivals).
    - Mitigation: Scalable backend architecture and load balancing.
4. Risk: Location privacy concerns.
    - Mitigation: Make location use opt-in, anonymize data, and store only necessary details.
5. Risk: SMS/Email notification failure.
    - Mitigation: Use reliable third-party APIs with retry logic and fallback to email if SMS fails.

6. **Risk: Malicious admin actions (misuse of privileges).**
    - **Mitigation: Maintain audit logs and restrict critical operations with dual authorization where needed.**
7. **Risk: Bugs or usability issues affecting adoption.**
    - **Mitigation: Conduct thorough testing (unit, integration, UAT) and gather user feedback.**

3.8 Traceability to Requirements

Requirement ID	Requirement Description	System Feature	Acceptance Test(s)
FR-1	Search nearby marts using GPS/manual location input	Location Search	AT-1
FR-2	View mart details (type, size, sections)	Mart Info Display	AT-1
FR-3	Select preferred section in a mart	Section Selection	AT-2, AT-3
FR-4	Book available slots	Slot Booking	AT-2
FR-5	Maintain waitlist when slots/sections full	Waitlist Management	AT-3
FR-6	Restrict cancellations to max 2/day	Cancellation Rule	AT-4
FR-7	View booking history & upcoming bookings	Booking History	AT-2
FR-8	Display Helping Hands Points	HHP Tracking	AT-6
FR-9	Send SMS/email alerts before slot	Notifications	AT-5
FR-10	Modify store details	Store Setup & Management	**AT-7**
FR-11	Manage sections & manpower requirements	Section Management	AT-2, AT-3
FR-12	View/manage user bookings	Booking Oversight	AT-2, AT-3
FR-13	Mark attendance	Attendance Management	AT-6
FR-14	Enforce intake limits	Intake Control	AT-3
FR-15	View user booking information	User Booking Records	AT-2
FR-16	Register/remove users	Admin Management	**AT-8**
FR-17	Register/remove marts	Admin Management	**AT-9**
FR-18	Access user details & store data	Data Access Panel	**AT-10**
FR-19	Monitor HHP performance	Performance Monitoring	AT-6
FR-20	Oversee system operations	Admin	**AT-11**
NFR-1	Handle 1000 concurrent users	Performance	**AT-12**

Requirement ID	Requirement Description	System Feature	Acceptance Test(s)
NFR-2	99.5% uptime	Reliability	**AT-13**
NFR-3	Responsive & user-friendly UI	Usability	AT-2
NFR-4	Slot booking must finish within 2 sec	System Speed	AT-2
NFR-5	Modular and maintainable	Maintainability	**AT-14**
SO-1	Protect confidentiality & integrity of user data	Security Framework	**AT-15**
SO-2	Enforce authorized access only	RBAC	**AT-16**
SR-1	Secure login/logout	Authentication	**AT-17**
SR-2	Role-based access control	Authorization	**AT-16**
SR-3	Encrypt sensitive data	Encryption Layer	**AT-15**
SR-4	Location accessed only with consent	Privacy Control	AT-1
SR-5	Regular backups & recovery	Backup Management	**AT-18**

3.9 Security Architecture

- Client Layer (Browser/Mobile): Handles input validation before sending requests.

- Application Layer (Backend): Implements RBAC, JWT authentication, and API security.

- Data Layer (Database): Stores encrypted data, enforces restricted access, and ensures backup/recovery.

- External Services: Integrated with secure APIs for notifications (SMS/Email) and location services.

4. Design

4.1 Design Overview

1. Presentation Layer

- **Provides role-based interfaces for End Users, Store Admins, and the Main Admin.**
- **Built using React.js for a responsive and interactive user experience.**
- **Supports desktop and mobile browsers.**

2. Application Layer

- **Backend implemented in Node.js with Express.js.**
- **Manages core business logic including:**
    - **Slot booking, waitlists, and cancellations.**

    - **Store and section management.**
    - **Attendance tracking and HHP point calculation.**
    - **Notification services (SMS/Email).**
- **Implements authentication and authorization via JWT and RBAC.**

3. Data Layer

- **MongoDB database to store users, stores, bookings, attendance, and HHP points.**
- **Supports backups, recovery, and secure access.**
- **Designed to scale with increasing data during peak events.**

4. External Services

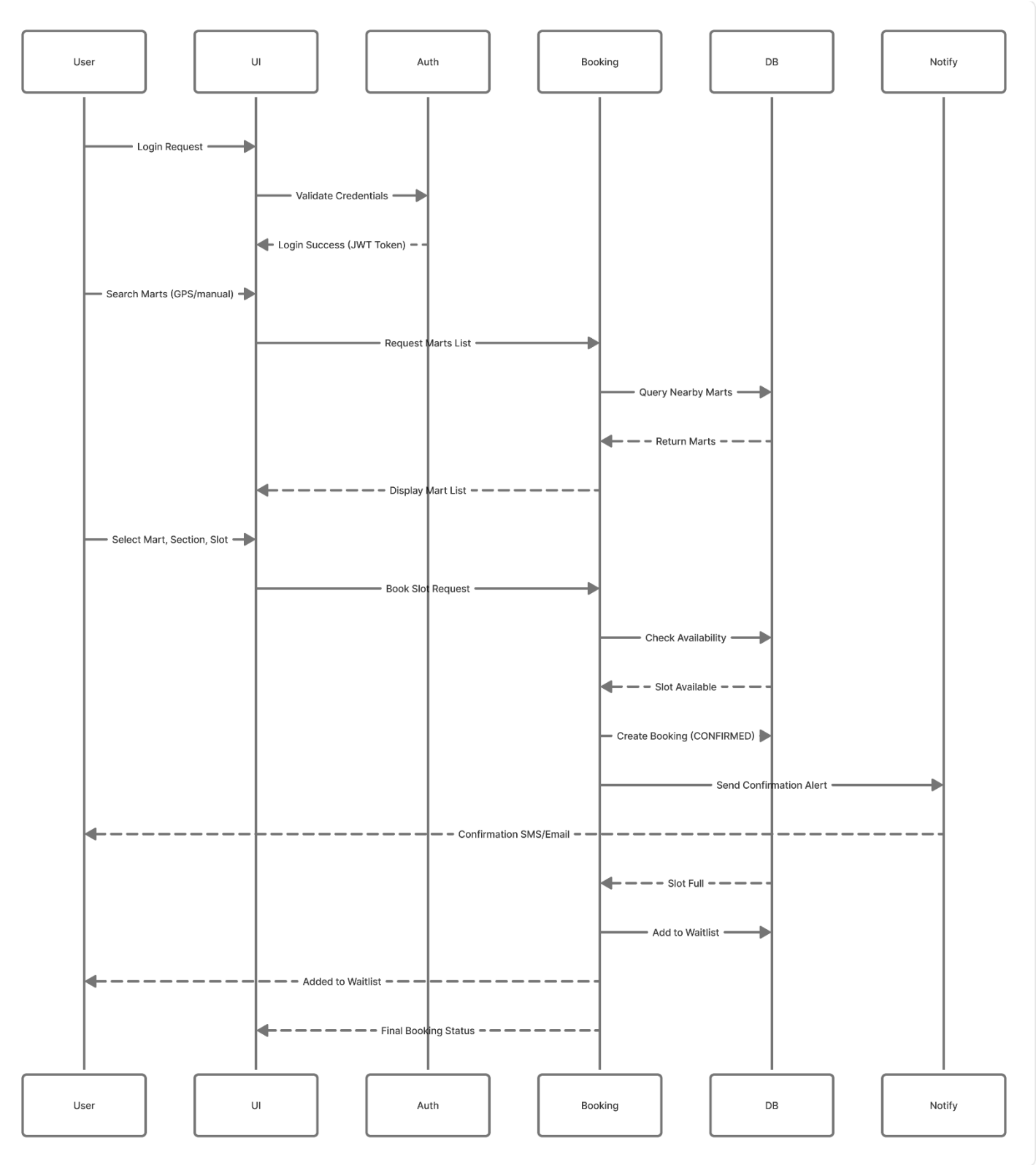
- **SMS/Email APIs (e.g., Twilio, SendGrid) for notifications.**
- **Geolocation API (HTML5/Google Maps) for mart search functionality.**

5. Key Design Principles

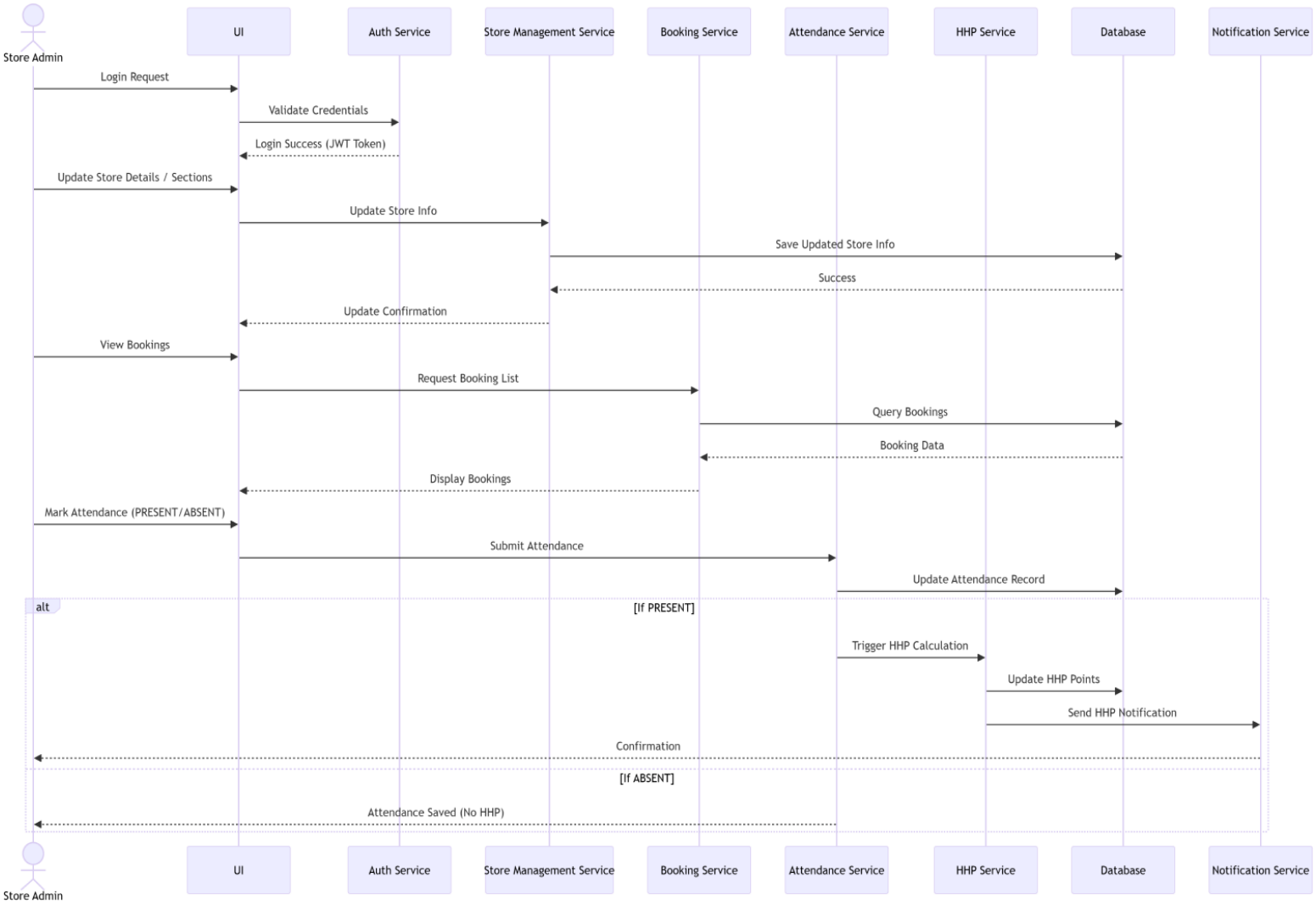
- **Modularity: Each feature (booking, store management, notifications) is a separate component.**
- **Security: Data encryption, RBAC, secure APIs, and location privacy controls.**
- **Scalability: Can handle 1000+ concurrent users during peak times.**
- **Maintainability: Clear separation of concerns allows independent updates to UI, backend, or database.**

4.2 UML Sequence Diagrams

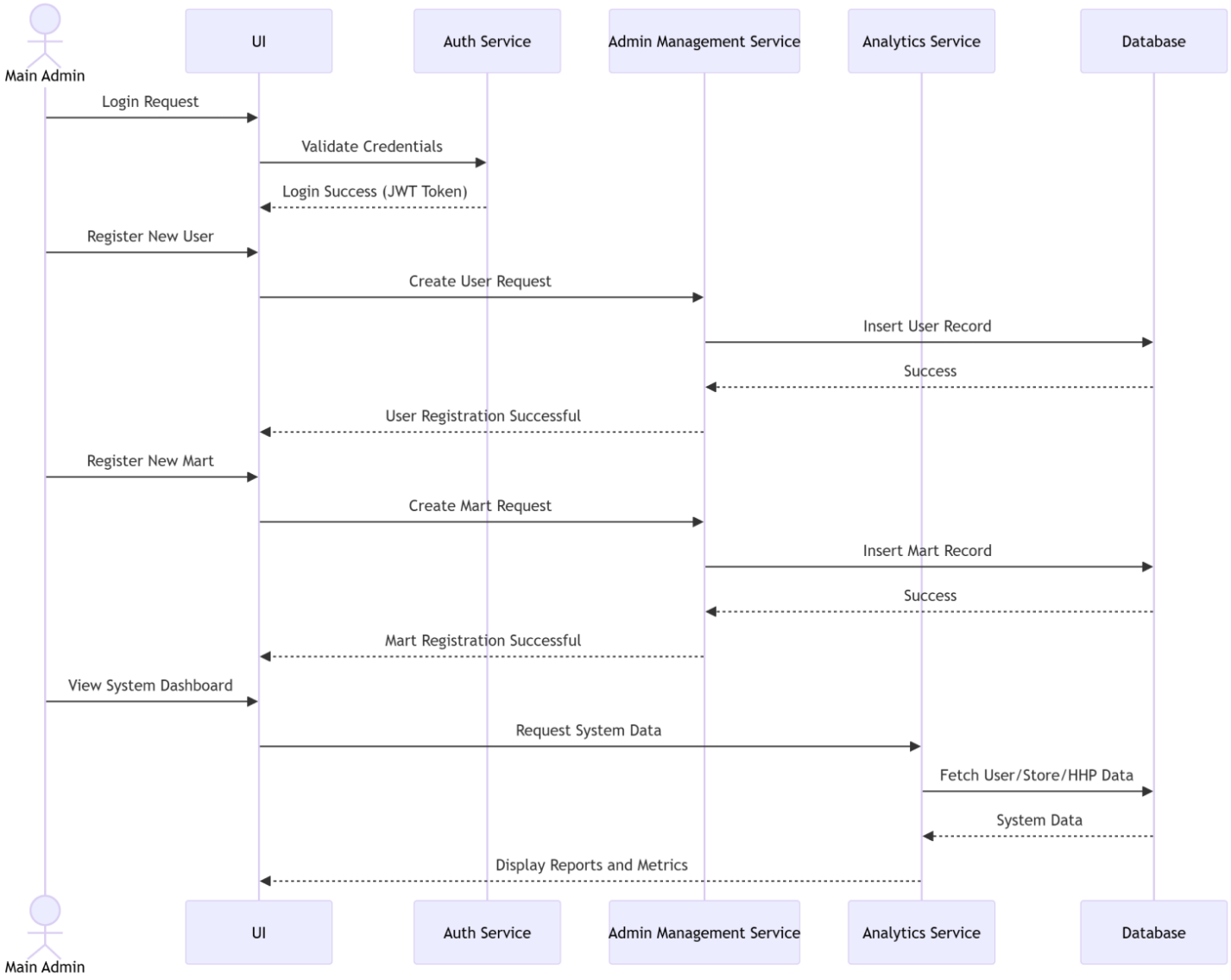
End User


User
UI
Auth
Booking
DB
Notify
Login Request
Validate Credentials
Login Success (JWT Token)
Search Marts (GPS/manual)
Request Marts List
Query Nearby Marts
Return Marts
Display Mart List
Select Mart, Section, Slot
Book Slot Request
Check Availability
Slot Available
Create Booking (CONFIRMED)
Send Confirmation Alert
Confirmation SMS/Email
Slot Full
Add to Waitlist
Added to Waitlist
Final Booking Status
User
UI
Auth
Booking
DB
Notify

Store Admin


Store Admin
UI
Auth Service
Store Management Service
Booking Service
Attendance Service
HHP Service
Database
Notification Service
Login Request
Validate Credentials
Login Success (JWT Token)
Update Store Details / Sections
Update Store Info
Save Updated Store Info
Success
Update Confirmation
View Bookings
Request Booking List
Query Bookings
Booking Data
Display Bookings
Mark Attendance (PRESENT/ABSENT)
Submit Attendance
Update Attendance Record
alt
[If PRESENT]
Trigger HHP Calculation
Update HHP Points
Send HHP Notification
Confirmation
[If ABSENT]
Attendance Saved (No HHP)
Store Admin
UI
Auth Service
Store Management Service
Booking Service
Attendance Service
HHP Service
Database
Notification Service

Main Admin


Main Admin
UI
Auth Service
Admin Management Service
Analytics Service
Database
Login Request
Validate Credentials
Login Success (JWT Token)
Register New User
Create User Request
Insert User Record
Success
User Registration Successful
Register New Mart
Create Mart Request
Insert Mart Record
Success
Mart Registration Successful
View System Dashboard
Request System Data
Fetch User/Store/HHP Data
System Data
Display Reports and Metrics
Main Admin
UI
Auth Service
Admin Management Service
Analytics Service
Database

4.3 API Design

1.Booking Management API

Endpoint: /api/bookings
Method: POST
Request: { userId, martId, section, slot }
Response: { status, bookingId, message }
Errors: 401 Unauthorized, 409 Slot already booked, 400 Invalid data.

Endpoint: /api/bookings/cancel
Method: POST
Request: { userId, bookingId }
Response: { status, message, remainingCancellations }
Errors: 401 Unauthorized, 403 Cancellation limit exceeded, 404 Booking not found.

Endpoint: /api/bookings/search
Method: GET
Request: { location, section }
Response: { status, marts: [{martId, martName, availableSlots}] }
Errors: 404 No marts found, 400 Invalid location input.

2.Attendance & HHP Management API

Endpoint: /api/attendance/mark
Method: POST
Request: { bookingId, userId, status }
Response: { status, message }
Errors: 401 Unauthorized, 404 Booking not found, 400 Invalid attendance status.

Endpoint: /api/hhp/update
Method: POST
Request: { userId, hoursWorked }
Response: { status, newHHP, message }
Errors: 401 Unauthorized, 404 User not found, 500 Calculation error.

Endpoint: /api/hhp/view
Method: GET
Request: { userId }
Response: { status, totalHHP }
Errors: 404 User not found, 401 Unauthorized.

4.4 Detailed Business Logic (Booking, Cancellation, Waitlist & HHP Rules)

This section defines the internal business rules that govern user bookings, cancellations, FCFS allocation, waitlist handling, attendance validation, and HHP point calculations. These rules ensure consistency across all modules, align with the SRS requirements, and are enforced through both backend logic and database constraints.

4.4.1 Slot Booking Logic (FCFS)

Objective: Allocate booking slots strictly on a First-Come-First-Serve (FCFS) basis.

Logic Steps:

1. User selects mart → section → date → slot duration.
2. System checks availability using:
3. available = capacity – confirmedBookingsCount
4. **If available > 0 → booking is confirmed.**
5. **If available = 0 → user is placed in the waitlist queue.**
6. Booking record is saved with status:
    - CONFIRMED
    - WAITLISTED
7. API enforces FCFS by timestamp sorting.

Related Requirements: FR-4, FR-5
Related Components: Booking Component, API Layer

4.4.2 Waitlist Management Logic

Objective: Ensure fair, automatic, FCFS-based movement from waitlist to confirmed bookings.

Logic Steps:

1. When a cancellation occurs, system checks waitlist queue.
2. First user in the queue is promoted to CONFIRMED.
3. User receives notification (Email/SMS).
4. Database operations use atomic transactions to prevent conflicts.

Queue Behavior:

- FIFO (First In → First Out)
- Maintains timestamp priority

Related Requirements: FR-5
Related Components: Booking Component, Notification Component

---

4.4.3 Cancellation Rules (Max 2 Per Day)

Objective: Prevent misuse by limiting daily cancellations.

Logic Steps:

1. When user requests cancellation:
    - System checks: cancellationCount for today.
2. **If count < 2 → cancellation allowed.**
3. **If count ≥ 2 → system returns error "Cancellation Limit Exceeded".**
4. Cancelled slot triggers:
    - Waitlist check
    - Promotion of next user

System Flags:

- Tracks cancellations by:
  userId + date

Related Requirements: FR-6
Related API: /api/cancellations

4.4.4 Attendance & Check-In Rules

Objective: Validate whether the user actually attended the booked slot.

Logic Steps:

1. Store admin opens attendance dashboard.
2. For each slot:
    - Admin marks PRESENT or ABSENT.
3. Presence triggers HHP calculation (Section 4.4.5).
4. Absence may reduce ranking or score in performance dashboard.

Related Requirements: FR-13
Related Components: Attendance Component

4.4.5 HHP (Helping Hands Points) Calculation Logic

Objective: Award points for each completed slot based on duration and store's multiplier.

Formula:

HHP = durationHours × pointsPerHour

Rules:

- PointsPerHour is set by store admin.
- If user completes more than one slot, points are cumulative.
- If marked ABSENT → HHP = 0
- Monthly summaries generated in Reporting module.

Example:
Slot = 3 hours, Rate = 10 HHP/hour
→ **User earns 30 HHP**

Related Requirements: FR-8, FR-19
Related Components: HHP Engine, Reporting Module

4.4.6 Priority Rules & Conflict Handling

To ensure fairness:

1. FCFS determines both confirmation and waitlist order.
2. No user can hold overlapping slots.
3. System prevents:
    - Double booking
    - Slot overlaps
    - Exceeding store capacity

On conflict, API returns:
409 — Slot conflict detected

Related Requirements: FR-4, FR-5

4.4.7 Notifications Logic

Triggered on:

- Booking confirmation
- Waitlist promotion
- Slot cancellation
- Attendance updates

Delivery channels:

- SMS
- Email

Notification metadata includes:

- date, time, store, section, slot duration

Related Requirements: FR-9

4.5 Error Handling, Logging & Monitoring

The system implements centralized error handling to capture and display user-friendly messages for invalid actions or failed operations. All critical events and errors are logged for audit and debugging, while real-time monitoring tools track API performance and system health.

4.6 UX Design

The UX design of the Helping Hands Software focuses on simplicity, clarity, and role-based accessibility. It provides intuitive navigation with separate dashboards for Users, Store Admins, and Main Admins. Users can easily search nearby marts, view available slots, and make bookings, while Store Admins and Main Admins have dedicated interfaces for management and monitoring. The design ensures responsive layouts, quick feedback for actions, and minimal steps for each task, enhancing the overall user experience.

4.7 Open Issues & Next Steps

Some features such as automated notification scheduling, advanced analytics for HHP tracking, and enhanced security auditing are pending implementation. Future steps include performance optimization, mobile app integration, and deployment of the system on a scalable cloud environment.

5. Appendices

5.1 Glossary: HHS, HHP, FCFS, UI, API
5.2 References: IEEE SRS Documentation Standard, Draw.io and StarUML for UML Diagram Design
5.3 Tools: React.js, HTML, CSS, JavaScript, Node.js, Express.js, MongoDB, StarUML, Draw.io, Git & GitHub