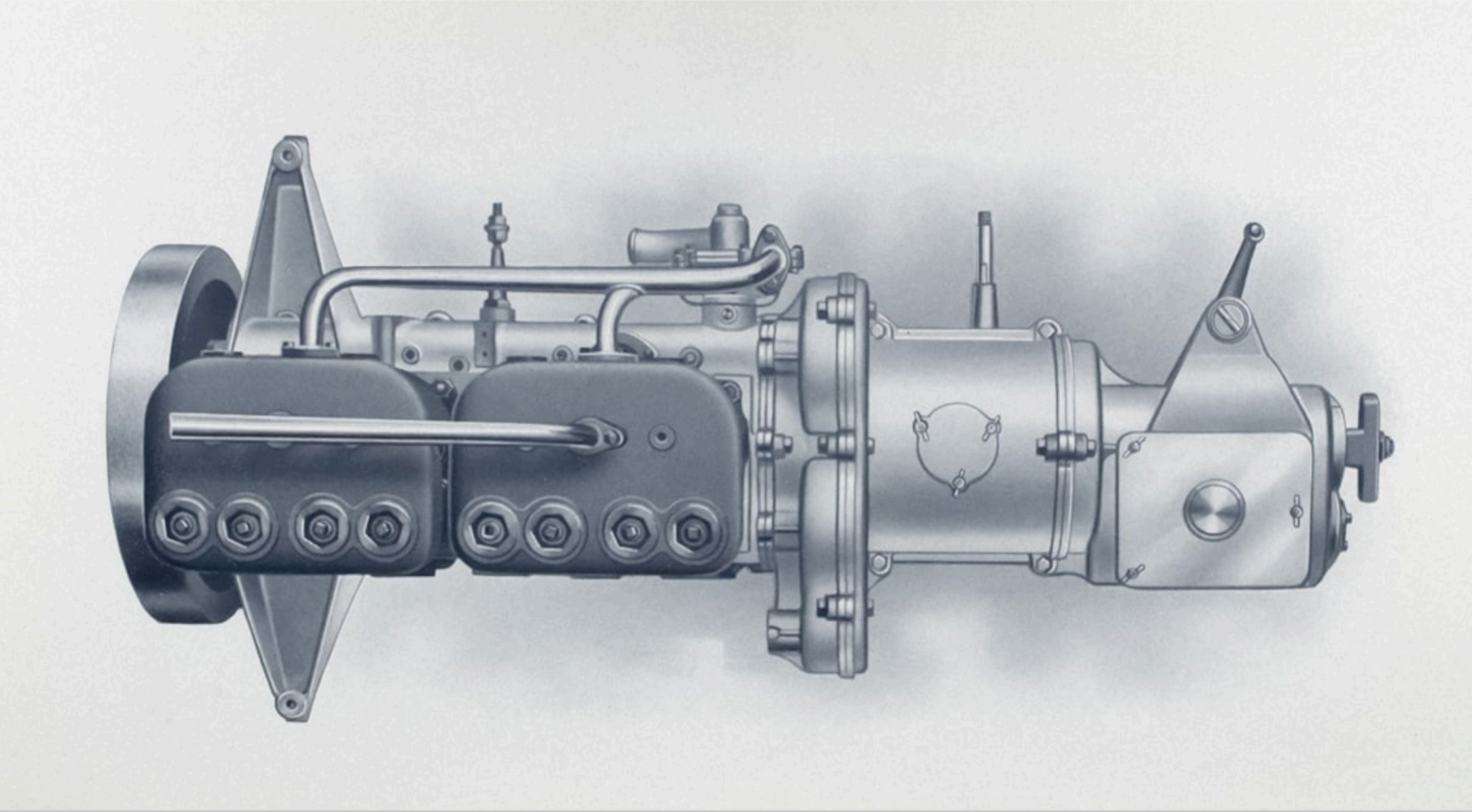


SMART MAINTENANCE

Salleha

Systems Analysis and Design
First Semester
2025/2026

University of Jordan

Systems Analysis and Design
Supervised by: Dr.Hamad Alsawalqah
First Semester 2025-10-15

Student Name	ID
Anas AL-Jallad	0225343
Orjoan Aldabaibah	0224933
Mosa Daradkah	0222634
Haneen Alajaleen	0226320
Shaima Hasan	0227646

Version Control

Version	Description
Version 1.0	Initial version for the software documentation. Added Project Initiation and Project Management Plan

Excutive Summary

Contents

Tables

Figures

1. Project initiation

1.1. Project Overview

SALLEHA is a platform designed for managing maintenance requests in facilities like offices or residential buildings, making maintenance and reporting more efficient and easier. Users can report issues, track progress, and get updates. Admins and technicians can assign, prioritize, and resolve tasks effectively.

1.2. Problem Definition

In many residential buildings, offices, and shared facilities people often face significant challenges in reaching authority of those In charge of maintenance managers and staff. In traditional methods such as : emails, paper forms, or phone calls, are typically inefficient, lack transparency and lead to delays. This often creates a communication gap between users and the authorities responsible, which results in frustration, unaddressed issues, and potential safety hazards.

1.2.1. Issues

Issue	weight
Users often struggle to reach the right maintenance personnel, resulting in delays or ignored requests. Without a centralized and accessible system, reporting issues becomes time-consuming and unreliable.	10
Maintenance teams often work without proper tools to prioritize, assign, and track tasks. This leads to missed or delayed repairs, no clear ownership of responsibilities, and no data to measure performance or improve operations.	9
Users rarely receive updates on the status of their maintenance requests. This lack of visibility creates frustration and reduces trust in the system, while maintenance teams struggle to keep everyone informed.	7
Users lack Privacy through and through with traditional reporting methods, this can lead to an upset and distrust to some people. Users care for their own privacy hence why some reports have never been sent before because of their own worry about the system.	6

1.2.2. Objectives

1. Simplify and centralize issue reporting through a user-friendly web/mobile interface that allows users to easily report maintenance problems and is available 24/7.
2. Enhance communication and transparency by providing real-time updates and notifications on request statuses
3. Create an analytics dashboard to provide administrators with insights and help them to identify trends and areas needing improvement.
4. Create a confidential system that ensures users anonymity and keeping their data secure and private from intruders.

1.2.3. Requirements

1. The system must ensure data security and protect the privacy of all users.
2. The system must be intuitive and user-friendly, allowing non-technical users to navigate and interact with it easily.
3. The analytics dashboard must be restricted to administrators only.

4. Maintenance reports must be submitted anonymously to ensure user comfort and honesty.

1.2.4. Constraints

1. Development costs must not exceed 45,000 JD
2. The project should be done by Sunday 4, Jan 2026

1.2.5. Vision Document

1.2.5.1. Problem Description

Ever since the digitalization of almost everything, people's expectations rose. People are in constant demand of systems that fulfills there needs. Current methods are almost obsolete they are ineffecient and insuffecient because the older methods have lack of transparency, increased delays, and unavaliable hence the overall user frustration, not to mention the diffuculty of managing the reports.

Without a system to hold everything together it require a lot of effort to pull through the maintenance tasks. To satisfy users, they need a system to adapt to their needs. Providing a smooth, painless experience through an easy to use interface. A system is needed such that it enables feedback submission, tracks administrative responses, and provides data-driven insights for continuous service improvement. Delaying this solution risks further dissatisfaction and missed opportunities for institutional growth.

1.2.5.2. System Capabilities

1. Ticket Submission Capabilities.
   - Users are able to submit maintenance tickets through a user-friendly interface.
   - Tickets include:
     - Text description of the issue.
     - Image attachments to provide visual context.
     - Location tagging to help technicians identify where the issue is.

2. Role-Based Dashboards.
   - The system provides separate dashboards based on user roles.
   - Roles include:
     - Users: Can submit and track tickets, view status updates, and provide feedback.
     - Technicians: Can view assigned tickets, update task statuses, and log maintenance work.
     - Administrators: Can assign tasks, monitor performance, and access analytics dashboards (restricted access).

3. Task Assignment and Scheduling.
   - Administrators can assign tasks to technicians based on priority and availability.
   - System supports:
     - Priority-based task distribution depending on if its urgent or normal.
     - Scheduling of tasks to optimize technician workload and response time.

4. Push and Email Notifications.
   - The system provides real time updates on ticket statuses.
   - Notification types include:

- Push notifications via web or mobile.
- Email alerts for important status changes.

5. Maintenance History and Analytics.
   - System keeps a log of all past maintenance activities.
   - Analytics dashboard features:
     - Insights into frequent issues by area or equipment.
     - Performance tracking for continuous improvement.
     - Access restricted to administrators only.

1.2.5.3. Business Benefits

- Providing a better quality of life.
- Overall increase of the user satisfaction.
- Increasing speed of maintenance tasks.
- Eliminating delays and lessening risks.
- Providing better communication channels.

1.3. Feasbility Studies

1.3.1. Techinical Feasibility

The technical feasibility assesses the technological components necessary to develop and operate the SALLEHA platform. This includes evaluating the required hardware, software tools, and the technical skills essential for building and maintaining the system.

Technology: The SALLEHA website is built using basic and easy-to-use Web tools like HTML, CSS, JavaScript, Bootstrap, and jQuery. These Tools help create a clean and responsive design that works well on Different devices. We also use Canva to design simple and clear images and graphics, making the website easy for Seniors users to understand and use .

Cloud Hosting: We are using GitHub to store and manage the project online. It helps us work together, keep track of changes, and easily share the project with others.

All the required resources including hardware, software tools, and hosting services are already available and accessible, ensuring smooth development and operation of the SALLEHA platform.

Since all these technologies are part of what we have learned at university and practiced in various projects, we are fully capable of developing and maintaining the SALLEHA platform using them. Our academic background and hands-on experience give us the technical foundation and confidence to build this system successfully.

1.3.2. Operational Feasibility

The proposed web and mobile application is operationally feasible. It is designed to receive maintenance requests in facilities such as universities, offices, or residential buildings, enabling users to report issues, track progress, and receive updates. Since it is both a web and mobile application, users can access it from anywhere. We expect that our system will gain wide acceptance from users, admins, and technicians because it addresses an essential need and saves time and effort. It will have clear privacy guidelines and mechanisms to ensure that our users' data will be secured, and it complies with the policies set by the country's laws and institutions. Additionally, the system is

well-suited to the local culture and environment. The end users are capable of using it smoothly and effectively without requiring extensive training, due to its simple and user-friendly design.

The proposed web and mobile application is operationally feasible. It is designed to receive maintenance requests in facilities such as universities, offices, or residential buildings, enabling users to report issues, track progress, and receive updates. Since it is both a web and mobile application, users can access it from anywhere. We expect that our system will gain wide acceptance from users, admins, and technicians because it addresses an essential need and saves time and effort. It will have clear privacy guidelines and mechanisms to ensure that our users' data will be secured, and it complies with the policies set by the country's laws and institutions. Additionally, the system is well-suited to the local culture and environment. The end users are capable of using it smoothly and effectively without requiring extensive training, due to its simple and user-friendly design.

1.3.3. Economic Feasibility

Development Costs:

Table 1: Devlopment Costs

Expense Category	Amount
Salaries	20,000 JD
Equipment and installations	8,000 JD
Training	1,500 JD
Facilities	2,000 JD
Utilities	1,000 JD
Travel\Miscellaneous	2,000 JD
Total	**39,500 JD**

Operational Costs:

Table 2: Operational Costs

Service	Annual Cost(Per year)
Operational maintenance	7,000 JD
Total Cost	**7,000 JD**

Table 3: Intangible Benefits

Intangible Benefits
Enhanced Institutional Trust and reputation
Increasing users satisfaction
Saving time and effort for both users and Institutions

Benefit and Payback Analysis:

Table 4: Benefit and Payback Analysis

Category	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
Value of benefits	0	16,000 JD	17,000 JD	18,000 JD	19,000 JD	20,000 JD
Development costs	−39,500 JD	0	0	0	0	0
Annual expenses	0	−7,000 JD	−7,000 JD	−7,000 JD	−7,000 JD	−7,000 JD
Net Benefit / Costs	−39,500 JD	9,000 JD	10,000 JD	11,000 JD	12,000 JD	13,000 JD
Discount Rate (7%)	1	0.934	0.873	0.813	0.763	0.713
Net Present Value (NPV)	−39,500 JD	8,406 JD	8,730 JD	8,943 JD	9,156 JD	9,269 JD
Cumulative NPV	−39,500 JD	−31,094 JD	−22,364 JD	−13,421 JD	−4,265 JD	5,004 JD
Payback Period	**4 years+**					**4.6 years**

Lifetime ROI = $\frac{90{,}000-74{,}500}{74{,}500} = 0.208 \vee 20.8\%$

Annual ROI = $\frac{20.8\%}{5} = 4.16\%$

1.3.4. Schedule Feasibility

Table 5: Project Development Schedule

Phase	Task	Estimated Time
Planning	Define Project Scope & Objectives	1 week
Analysis	Requirements Gathering, Process Analysis, and Document Delivery	2 weeks
Design	System Architecture and Interface Design	2 weeks
Implementation	Development of Core Features	4 weeks
Testing	System Testing and Quality Assurance	2 weeks
Deployment	System Deployment	1 week

1.3.5. Legal Feasibility

The proposed platform fully aligns with Jordanian laws, university policies, and institutional standards. All required approvals will be obtained from the University of Jordan's relevant departments before deployment. The system does not infringe upon any legal frameworks or intellectual property rights.

Licensing Compliance: All development tools, frameworks, and libraries used in the platform will be properly licensed. Open-source components will be used in accordance with their respective licenses, while any proprietary technologies will be incorporated only after acquiring valid usage rights.

Copyright and Intellectual Property Protection: The platform will comply with the Jordanian Copyright Law No. 22 of 1992 and its amendments. Any third-party content whether text, images, or software will be original, licensed, or used under fair use conditions with full attribution.

Data Privacy and Confidentiality: To comply with the Jordanian Personal Data Protection Law No. 24 of 2023, the system will:

1. Obtain explicit user consent prior to collecting or processing personal information.
2. Employ encryption and secure storage for sensitive data.
3. Ensure that personal data is used strictly for its intended purpose and accessed only by authorized personnel.

Electronic Communication and Records Compliance: Under the Electronic Transactions Law No. 15 of 2015, all digital communications and transactions carried out through the platform will be handled as legally recognized records, protected through appropriate technical and procedural safeguards.

Terms of Service and Legal Disclosures: Users will be provided with clear Terms of Service and Privacy Policy agreements outlining:
- Data collection and usage practices
- User rights and responsibilities
- Risk disclosures and security provisions

These documents will comply with both university IT regulations and national legal requirements.

1.4. Recommended Solution and Expected Project Deliverables

To manage the maintenance requests of issues as they arise, we can use a great solution: a Maintenance Request software system that allows requesters to report maintenance issues directly to the maintenance team using a web-based form or mobile app. It helps streamline communication, submission, and tracking without wasting time gathering complete and accurate information or delaying repairs. The people who the maintenance teams usually rely on, such as employees and visitors, will be able to submit detailed maintenance forms that include descriptions, images, and location information. Other processes such as workflows for reviewing and approving requests will be managed through a dashboard that allows the team to assign, prioritize, and monitor tasks. Technicians can update the status of each request in real time, and notifications will be sent to users to inform them about progress and completion.

Expected Deliverables: A Maintenance Request Software that will include:

- **Request Submission:**

Maintenance teams will accept requests through the system to ensure they collect all necessary information to proceed with other maintenance processes.
Users will submit it like a post; It'll have a title, location, photo and description.

- **Review and Approve Requests:**

A dashboard and analysis tool will help the organization review all forms and decide which requests will be approved. Each request will be evaluated to ensure that it is valid, not redundant, and not already being addressed.

- **Status Updates to Deliver Better Customer Service:**

To build trust between managers and customers, there will be a communication tool that responds back to requesters to inform them that their requests are accepted and to update them about the status of their issues until completion. These updates will be automated through real-time notifications.

- **Database Design and Documentation:**

For storing and tracking requests, there will be request records that provide a clear view of all issues, and a database that documents what issues were reported, how they were resolved, and when. This will help with future planning and decision-making, as well as provide tools for summarizing the analysis, design, and implementation process.

- **Performance Tracking:**

Updates on team work status for measuring team performance will help ensure that the original problem has been addressed and will identify bottlenecks in the request process. This will lead to providing excellent customer service and demonstrate that the maintenance team is responsive, works well, and continues improving the organization's overall reputation. This will be done by tracking average response time and turnaround time.

1.5. Local and Global Impact of the Proposed Solution

Locally: The maintenance request system will ensure accuracy and enable the maintenance team to begin planning and scheduling maintenance work more quickly through automatically generated work orders from approved requests. Following best practices will also provide better customer service in this area. Automation means better experiences! It will reduce delays in handling requests, minimize manual paperwork, and ensure maintenance work is managed from start to finish through automated tools. This can lead to better resource management, higher efficiency, and greater satisfaction not just among staff but also among customers by keeping them updated about the status of their requests without delaying feedback.

Globally: It contributes to digital transformation and sustainability efforts. It is essential for businesses of all sizes to rely on such systems in their operations to efficiently allocate resources and maximize the performance of their assets. A well-designed maintenance request system demonstrates how technology enables organizations to make data-driven decisions to achieve better operational efficiency by centralizing all maintenance requests in one platform.

2. Project Management plan

2.1. Project Organization

The project organization shows how the team is structured and who is responsible for each part of the project. It helps make sure everyone knows their role and who to report to, keeping the work organized and efficient. The structure for our team also supports good communication between team members during development, as shown in Figure 1.

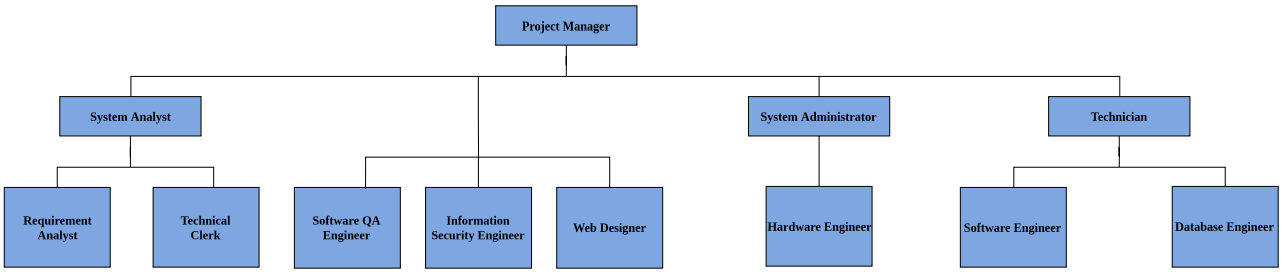


Figure 1: Project Organizational Structure

Table 6: Team Roles Assignments and Responsibilities

Assigned Membe	Roles
Anas	Project Manager, System Analyst, Requirement Analyst
Orjoan	Technical Clerk, Technician
Mosa	System Administrator, Hardware Engineer,Database Engineer
Haneen	Software Engineer, Web Designer,
Shaima	Software Engineer, Web Designer, Technical Clerk
External	Information Security Engineer, Software QA Engineer

2.2. Roles and Responsibilities

In this section, we will explain each role with its responsibilities in perspective. see Table 7.

Table 7: Roles and Responsibilities

Role	Responsibility
Project Manager	Responsible for planning, organizing, and overseeing projects to ensure they are completed on time, within budget, and meet quality standards. They coordinate team efforts, manage resources, and communicate with stakeholders throughout the project lifecycle.
System Analyst	Systematically assesses how businesses function by examining the inputting and processing of data and the outputting of information with the intent of improving organizational processes.
Requirement Analyst	Gathers and documents user requirements, ensuring alignment between business needs and system design.
Technical Clerk	Assists in maintaining documentation, schedules, and techinical records. Supports techinical team with admin tasks.
Software QA Engineer	Ensuring that software products meet the highest standards of quality and functionality.
Information Security Engineer	Rresponsible for designing, implementing, and maintaining security systems to protect an organization's data and networks from cyber threats. They also monitor security measures, respond to incidents, and ensure compliance with security policies and regulations.

Web Designer	Responsible for creating the visual layout and user experience of websites, ensuring they are both attractive and functional. Their tasks include designing page layouts, coding navigation, and collaborating with clients to meet their needs.
System Administrator	Responsible for managing and maintaining an organization's computer systems and networks, ensuring they operate efficiently and securely. This role often involves troubleshooting issues, installing software, and managing user accounts.
Hardware Engineer	Responsible for analyzing blueprints and technical drawings, reviewing system tests and performing updates as needed, implementing the latest systems and processes and ensuring everyone follows them, monitoring the manufacturing and assembly of hardware equipment, acting as the technical leader in product development
Technician	Responsible for installing, maintaining, and repairing equipment and systems across various industries. Their key duties include troubleshooting issues, performing routine maintenance, and ensuring compliance with safety standards
Software Engineer	Responsible for designing, developing, testing, and maintaining software applications and systems to meet user needs. Their responsibilities include analyzing user requirements, writing and testing code, and collaborating with other team members to ensure software functionality and performance.
Database Engineer	Rponsible for designing, implementing, and maintaining database systems to ensure efficient data storage and retrieval. Their key responsibilities include database design, data security, performance optimization, backup and recovery, and data migration.

2.3. Software Process Model

We are going to use the Waterfall Software Process Model. This model is suitable because our project goals and requirements are clearly defined from the beginning, and we have a strict timeline to follow. Since this model is based on a highly structured approach, it will help us maintain organization and ensure that all deliverables are completed on time. See Figure 2.

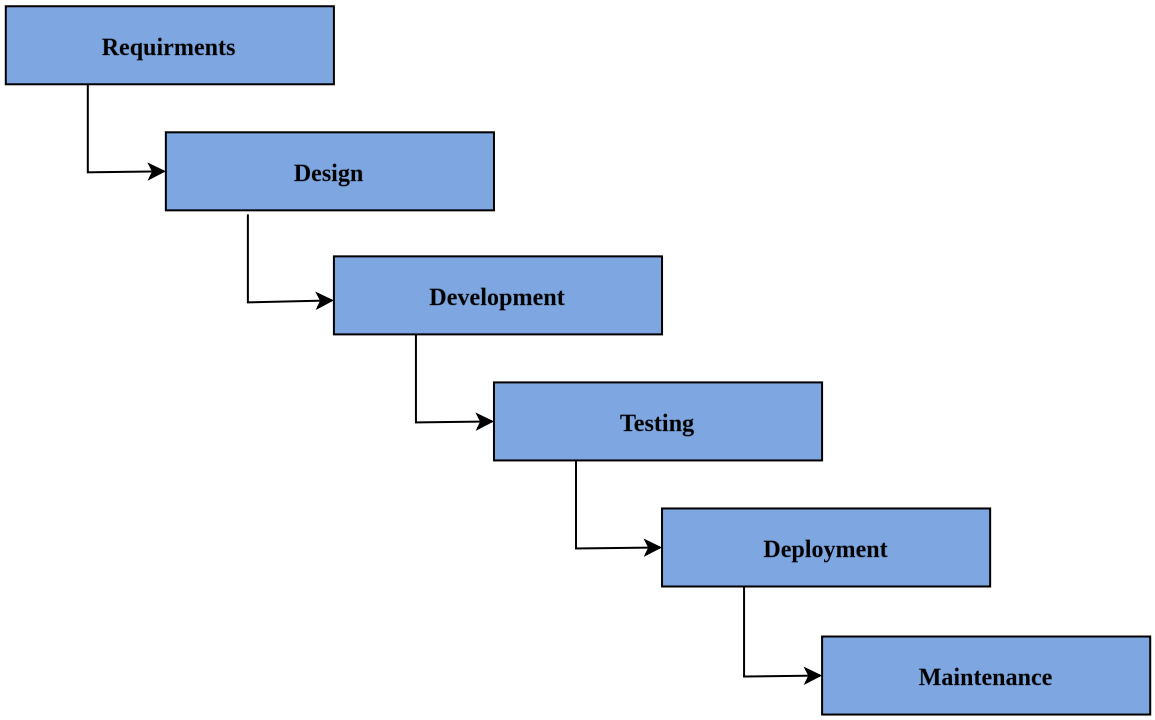


Figure 2: Waterfall Software Process Model

2.3.1. Main Phases :

1. **Requirements Analysis and Specification:**
   - Requirements Analysis: Gathering and understanding all the requirements of the client, then documenting and analyzing them.
   - Requirement Specification: Documenting the analyzed requirements in a software requirement specification document that serves as a reference for the next phases.
2. **System Design:**
   - Translating the requirements from the requirement specification document into a detailed system design as well as creating the overall architecture.
3. **Development:**
   - Developing the web and mobile applications according to the designs created earlier, using a suitable programming languages and frameworks.
4. **Testing and Deployment:**
   - Testing the whole software and verifying that all components work correctly and satisfy user expectations. After testing, the software is ready and available for use.
5. **Maintenance:**
   - The final, ongoing phase. It ensures that the software remains functional, secure, and up-to-date throughout its operational life.

2.4. Project Environment

2.4.1. Procedures

- **Initiation:**
  - Establish the project team, define the requirements goals, and potential risks.
- **Planning:**
  - Create a detailed project plan using the Waterfall methodology, this includes outlining all necessary steps, allocating resources, and developing cost, schedule, and communication plans to achieve our outcomes.
- **Execution and Testing:**
  - Implement the planned activities and test them carefully to ensure quality and functionality.
- **Monitoring:**
  - Track progress and compare it with planned goals to ensure timely delivery and quality control.
- **Documentation:**
  - All design diagrams, reports, and testing results are documented using Typst and stored in a shared repository to ensure collaboration among team members.

2.4.2. Tools

Table 8: Tools

Tool	Purpose
Development Tools	Visual Studio Code for developing the web application, and Flutter for building a responsive mobile application using one shared codebase.
Documentation	Typst for creating our PDF documentation.
Version Control	GitHub for sharing and tracking project progress and managing team collaboration.
Design & Prototyping	Figma for creqting UI/UX design, and Draw.io for diagrams.
Database Management	MySQL, chosen for its reliability, speed, and support for relational data.
Testing	JUnit, an open-source testing framework, to verify our code's correctness and performance.
Communication Tools	Google Meet, WhatsApp, and GitHub for coordination.

2.4.3. Hardware (HW) Resources:

Table 9: Hardware (HW) Resources

Category	Description
Developer Devices	Personal laptops and PCs for developing both the web and mobile applications.
Database Server	A server for hosting MySQL database, optimized for security and data backup.
Testing Devices	PCs for testing the website, and Android smartphones for mobile testing

2.4.4. Software (SW) Resources:

Table 10: Software (SW) Resources

Category	Description
Frontend Technologies	HTML, Tailwind CSS, and JavaScript for dynamic and responsive designs.
Mobile Development	Flutter for building the mobile app.
Frameworks and Libraries	React for web development, and Flutter for mobile.
Backend Technologies	Java and Node.js for handling server-side operations and building a secure and scalable backend.
Database	MySQL for storing and managing maintenance request data efficiently.
Operating System	Windows

2.5. Project tasks

Table 11: Project Tasks

Phase	Task	Detailed task	EstimatedTime
• Resource & Schedule Planning (T1) • Requirements Gathering (T2)	• Develop Work Breakdown Structure for web and mobile app development • Create risk register and define quality standards for the application	• Define project phases and deliverables • Assign WBS elements to team members • Identify risks and quality objectives • Define acceptance criteria and mitigation strategies	1 weeks
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**
	• N/A	• **Dependencies** ▸ Approved project charter ▸ Availability of key stakeholders for interviews such as the requesters and the facility managers • **Constraints** ▸ Budget constraints	• **Deliverables** ▸ Work breakdown structure diagram ▸ Table with risk, impact, probability, priority, and mitigation ▸ Quality standards document (acceptance criteria checklist) • **Milestones** ▸ M1.1: Charter Approved and communicated to the team. ▸ M1.2: Work Breakdown Structure completed and reviewed by all members. ▸ M1.3: Resource plan and schedule baseline approved.

			▸ M1.4: Risk register and quality standards finalized.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Analysis	• Requirements Elicitation (T3) • Requirements Modeling & Documentation (T4) • Requirements Validation (T5)	• Conduct stakeholder interviews • Distribute surveys to end-users • Observe existing maintenance request processes • Write Software Requirements Specification (SRS) • Create use-case diagrams for maintenance workflows • Develop entity-relationship diagrams (ERDs) for database • Document non-functional requirements (performance, security, compatibility) • Facilitate requirements review sessions • Build wireframes for React web and Flutter mobile interfaces • Resolve conflicts and ambiguities • Obtain formal sign-off on SRS	2 weeks
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**
	• N/A	• **Dependencies**	• **Deliverables**

		▸ Access to collaboration tools so the informations can be shared and reviewed. • **Constraints** ▸ Stakeholder time; limited availability may restrict depth of interviews ▸ Data privacy when handaling the facility data	▸ Software Requirements Specification ▸ Use-case diagrams for the main maintenance workflows ▸ Entity-Relationship Diagram ▸ Wireframes ▸ Survey results and interview notes ▸ Non-functional requirements list ▸ Requirements validation report ▸ Requirements Traceability Matrix • **Milestones** ▸ M2.1: Software Requirements Specification (SRS) reviewed and approved by stakeholders. ▸ M2.2: Requirements validation and traceability matrix completed.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Design	• Architectural Design (T6) • High-Level (Logical) Design (T7) • Detailed (Physical) Design (T8) • Design Review & Approval (T9)	• Choose overall system architecture for web and mobile • Define network topology and cloud infrastructure • Break system into modules (frontend, backend, database)	2 weeks

		• Define REST API contracts and module interfaces • Draft high-level sequence diagrams for maintenance request workflows • Create class diagrams for React and Flutter components • Design database schema with tables, indices, and constraints • Specify UI layouts and navigation flows for web and mobile • Define error-handling and logging approaches • Organize design walkthroughs with team and stakeholders	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• Information Security Engineer	• **Dependencies**
		  ▸ Completion and approval of Software Requirements Specification
		  ▸ UI design depends on confirmed user workflows and functional requirements because they show what the user needs to do and how the
			• **Deliverables**
			  ▸ Web and Mobile System Architecture Document
			  ▸ High-Level Design including system modules and interaction diagrams
			  ▸ Detailed Design including Class diagrams, Sequence diagrams, Database schema

		system should respond • **Constraints** ▸ Technology constraints ▸ Design must comply with quality standards and security requirements	and API documentation ▸ UI/UX prototypes for web and mobile • **Milestones** ▸ M3.1: System architecture approved. ▸ M3.2: High-level and detailed design documents completed. ▸ M3.3: UI/UX prototypes finalized and validated with stakeholders. ▸ M3.4: Design reviewed and approved by all stakeholders.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Development	• Development Setup (T10) • Front-end Code (T11) • Back-end Code (T12) • Database Physical Design (T13)	• Configure Git repository and version control • Set up web development environment • Set up Flutter development environment with Android/iOS SDKs • Initialize backend framework and dependencies • Configure linters, formatters, and testing frameworks • Implement web frontend components for maintenance	4 weeks

		request management • Build Flutter screens for mobile app navigation • Develop responsive UI for web and mobile platforms • Create RESTful API endpoints for maintenance operations • Implement authentication and authorization logic • Build business logic for request processing and notifications • Create database tables, indices, and relationships • Implement data access layer with ORM • Set up database migrations and seeders	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• N/A	• **Dependencies**
		  ▸ Depends on approved design phase deliverables auch as architecture, and database schema
		• **Constraints**
		  ▸ Follow the coding standards, security rules, and framework versions that were set in the earlier phases
			• **Deliverables**
			  ▸ Git repository
			  ▸ Working web and mobile front-end modules (React and Flutter)
			  ▸ Implemented database schema
			  ▸ Unit testing and integration tests
			• **Milestones**
			  ▸ M4.1: Development environment configured and

		▸ Internet or cloud service interruptions may slow down integration and testing activities	repository initialized. ▸ M4.2: Initial Functional prototype completed. ▸ M4.3: Unit and integration testing completed with 80% code coverage. ▸ M4.4: Quality assurance (QA) verification passed and build approved for testing. ▸ M4.5: Production environment prepared and ready for deployment.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Testing	• Test Planning (T14) • Integration Testing (T15) • System & Acceptance Testing (T16) • Regression & Release Testing (T17)	• Develop comprehensive test plan for web and mobile platforms • Define test environments (development, staging, production) • Prepare test data sets for maintenance request scenarios • Test integration between web frontend and backend API • Test integration between Flutter mobile app and backend API • Verify database operations and data integrity	2 weeks

		• Conduct end-to-end system testing across all platforms • Perform user acceptance testing with stakeholders • Test cross-platform compatibility (iOS, Android, Web browsers) • Execute regression tests after bug fixes • Perform security and performance testing • Conduct final release testing and quality checks	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• External QA engineers	• **Dependencies**
		  ▸ Test data and environment setup depends on finalized database
		  ▸ Test plan and test cases finalized
		• **Constraints**
		  ▸ Availability of performance-test and security-test tools
		  ▸ Limited time for testing may constrain full regression coverage
		  ▸ Must follow project's quality assurance and version control procedures
			• **Deliverables**
			  ▸ Completed integration and system test reports
			  ▸ Signed-off User Acceptance Test report from stakeholders to be ready for deployment
			• **Milestones**
			  ▸ M5.1: Test plan and test cases developed and approved.
			  ▸ M5.2: System and integration tests executed successfully.
			  ▸ M5.3: User Acceptance Testing (UAT) completed and approved.

			▸ M5.4: Exit criteria met and testing phase formally closed.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Deployment	• Release Planning (T18) • Environment Provisioning (T19) • Go-Live Execution (T20) • Transition & Support (T21) • Post-Deployment Review (T22)	• Create release plan with rollback strategy • Prepare deployment documentation and checklists • Schedule go-live date with stakeholders • Provision cloud servers and configure network • Set up production database with security settings • Configure CI/CD pipelines for automated deployment • Deploy web application to hosting platform • Publish Flutter mobile app to App Store and Google Play • Configure production environment variables and API keys • Conduct user training sessions for web and mobile platforms • Provide technical documentation and user guides • Establish helpdesk and support channels	1 weeks

		• Monitor system performance and user feedback • Review deployment metrics and KPIs • Document lessons learned and improvement areas	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• N/A	• **Dependencies**
		  ▸ Dependent on successful completion of testing
		• **Constraints**
		  ▸ It must be scheduled
		  ▸ Risk of configuration errors
		  ▸ Security and data compliance must be checked and confirmed before the system goes live
			• **Deliverables**
			  ▸ Successfully deployed web and mobile applications
			  ▸ User training and support materials completed
			  ▸ Documented post-deployment review and lessons learned
			• **Milestones**
			  ▸ M6.1: Final stakeholder approval for production release obtained.
			  ▸ M6.2: Web and mobile applications deployed to production environment.
			  ▸ M6.3: User training sessions completed.
			  ▸ M6.4: Post-deployment review completed.
			  ▸ M6.5: Project officially closed.

2.6. Project Schedule

2.6.1. Activity Network

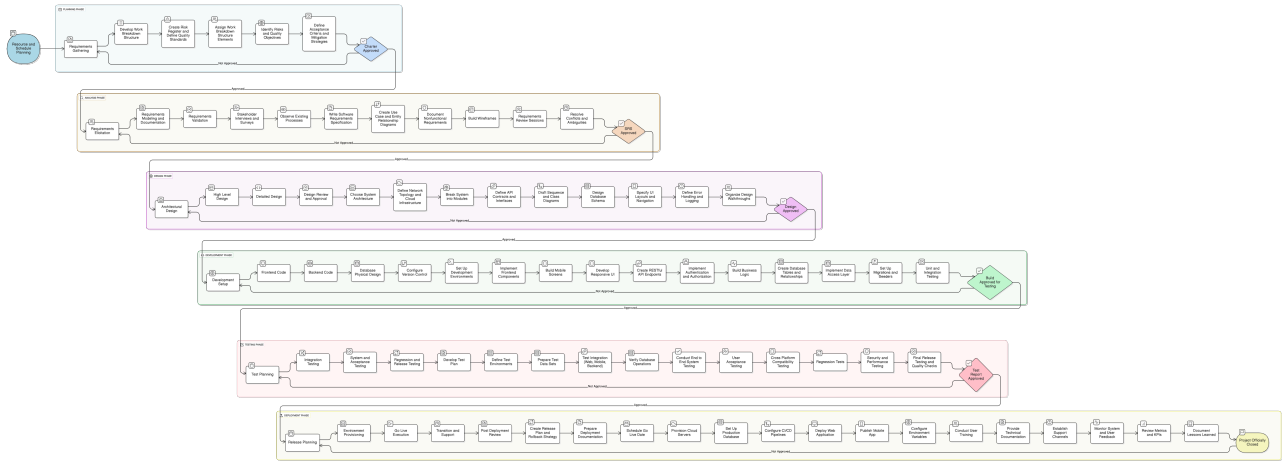


Figure 3: Activity Network Diagram

2.6.2. Gantt Chart

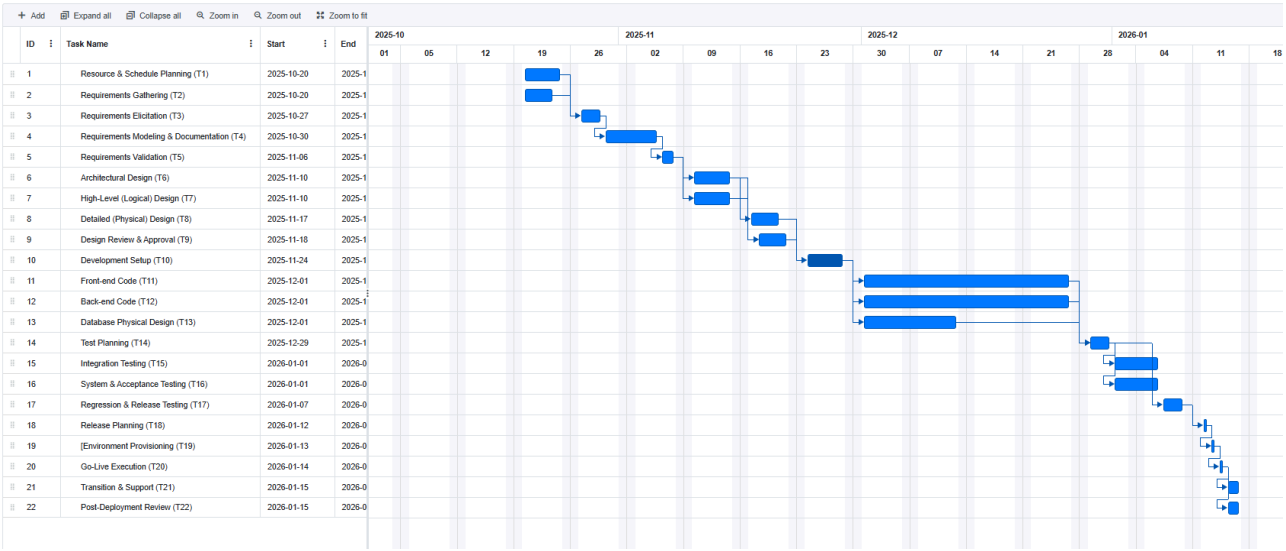


Figure 4: Gantt Chart

2.7. Assigning Team Members to Tasks

Table 12: Task-to-Team Member Assignment

Task ID	Assigned to
T1	Project Manager (Anas), Technical Clerk (Orjoan)
T2	Project Manager (Anas)
T3	System Analyst (Anas), Requirement Analyst (Anas), Technical Clerk (Orjoan)
T4	System Analyst (Anas), Requirement Analyst (Anas)
T5	System Analyst (Anas), Technical Clerk (Orjoan)
T6	Software Engineer (Haneen & Shaima), Hardware Engineer (Mosa), Web Designer (Haneen & Shaima)
T7	Software Engineer (Haneen & Shaima), Database Engineer (Mosa)
T8	Software Engineer (Haneen & Shaima), Database Engineer (Mosa), Web Designer (Haneen & Shaima)
T9	System Analyst (Anas), Software Engineer (Haneen & Shaima), InfoSec Engineer (Ext.)
T10	Software Engineer (Haneen & Shaima)
T11	Software Engineer (Haneen & Shaima)
T12	Software Engineer (Haneen & Shaima)
T13	Database Engineer (Mosa)
T14	Software Engineer (Haneen & Shaima), Software QA Engineer (External)
T15	Software Engineer (Haneen & Shaima)
T16	Software QA Engineer (External), System Analyst (Anas)
T17	Software QA Engineer (External), Software Engineer (Haneen & Shaima)
T18	Software QA Engineer (External), System Analyst (Anas)
T19	Software QA Engineer (External)
T20	System Administrator (Mosa), Project Manager (Anas), Software Engineer (Haneen & Shaima)
T21	System Administrator (Mosa), Database Engineer (Mosa)
T22	System Administrator (Mosa), Software Engineer (Haneen & Shaima), InfoSec Engineer (Ext.)

2.8. Monitoring and Controlling Mechanisms

Earned value management

Table 13: Earned Value Management Progress Tracking

Phase	Estimated cost	Cumulative estimate	Estimated duration	Stage completed	Actual cost of Phase to date	Actual cost of project to date
Planning	4,000 JD	4,000 JD	2 weeks	80%	1,000 JD	1,000 JD
Analysis	6,500 JD	10,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Design	8,000 JD	18,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Implem entation	16,000 JD	34,500 JD	4 weeks	0%	Not yet begun	Not yet begun
Testing	8,000 JD	42,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Deployment	5,000 JD	47,500 JD	1 week	0%	Not yet begun	Not yet begun

- $P = 80\%$
- **Planned Value (PV) = 4,000 JD**
- **Actual Cost (AC) = 1,000 JD**
- **Earned Value (EV) = 3,200 JD**
- **Cost Variance (CV) = 2,200 JD**
  - We have saved 2,200 JD on the planning phase
- **Schedule Variance (SV) = (-800) JD**
  - This indicates delayed progress in the planning phase.
- **Cost Performance Index (CPI)= (3.2)**
  - We are under budget. The team is spending less than planned to accomplish the work.
- **Schedule Performance Index (SPI) = (0.8)**
  - We are behind schedule. The progress is slower than expected.
- **Estimate to Complete (ETC) = (13,593.75) JD**
- **Estimate at Completion (EAC) = 14843.75 JD**

- Based on information we gained from EVM analysis, we have to expedite our schedule.

Table 14: Time and Cost Options

Activity	Estimated duration (days)	Crash time (days)	Cost/day (JD)
T1	5	3	200
T2	4	4	200
T3	3	3	300
T4	5	4	450
T5	2	2	300
T6	5	5	300
T7	5	5	300
T8	4	4	300
T9	4	4	300
T10	5	4	250
T11	20	17	350
T12	20	19	400
T13	10	18	450
T14	3	3	250
T15	4	4	250
T16	4	4	300
T17	3	2	300
T18	1	1	300
T19	1	1	250
T20	1	1	250
T21	2	1	150
T22	2	2	300

Table 15: Schedule Expediting Table

Eligible activities	Activity chosen	Time for each path (Days)			Cost (JD/day)	Cumulative cost (JD)
		83	83	73		
T1,T4, T10,T11,T12, T17,T21	T21	82	82	72	150	150
T1,T4, T10,T11,T12, T17	T1	81	81	71	200	350
T1,T4, T10,T11,T12, T17	T1	80	80	70	200	550
T4, T10,T11,T12, T17	T10	79	79	69	250	800
T4, ,T11,T12, T17	T17	78	78	68	300	1100
T4, ,T11,T12, T17	T17	77	77	67	300	1400
T4, ,T11,T12,	T11	76	77	67	350	1750
T4, ,T11,T12,	T11	75	77	67	350	2100
T4, ,T11,T12,	T11	74	77	67	350	2450
T4, ,T12,	T12	74	76	67	400	2850
T4	T4	73	75	66	450	3300

2.9. Risk Analysis

2.9.1. Effects and Causes

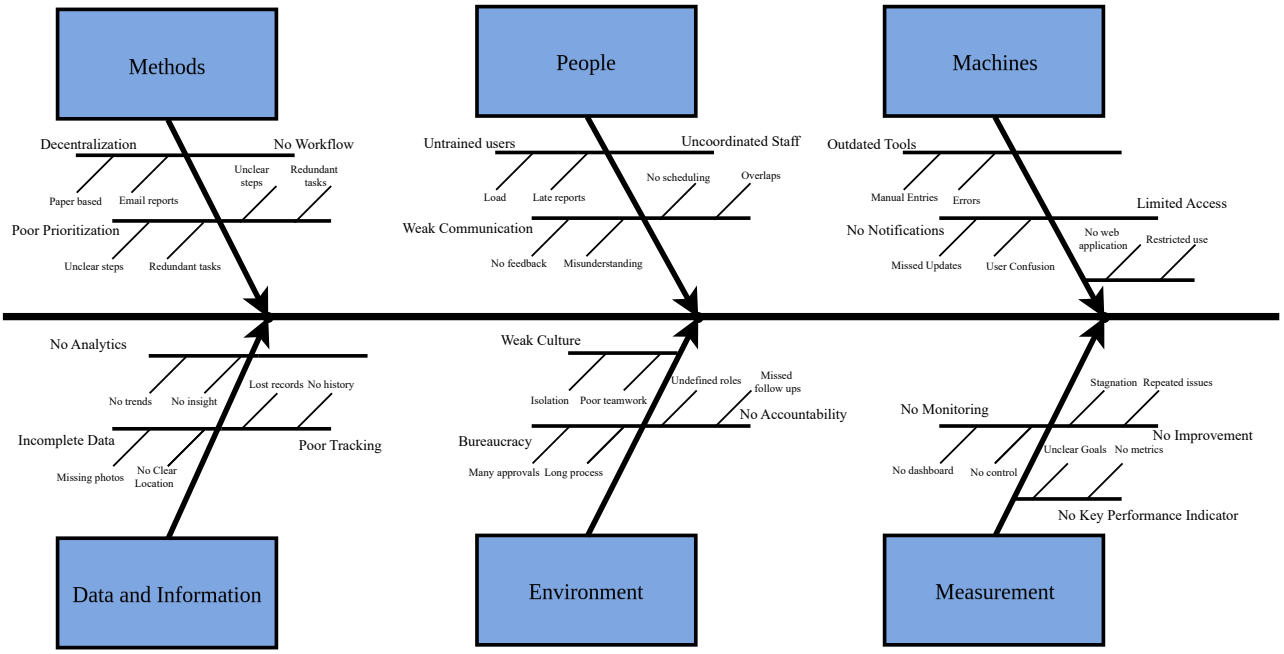


Figure 5: Fishbone Diagaram

2.10. Communication Plan

Our team will maintain communication through online meetings using Google Meet, and instant messaging using a WhatsApp group to assign tasks and ensure that everyone is updated on progress and deadlines. **Communication Methods:**

- In-Class Meetings:
  - Held every Sunday to discuss progress, issues, and next steps.
- Out-of-Class Meetings:
  - Through Google Meet whenever there is a need for rapid discussion.
- Messaging:
  - A WhatsApp group for updates and daily coordination.
- Version Control:
  - GitHub is used for tracking code updates and version history.
  - We also use GitHub to store and organize our documentation created using Typst, ensuring all of us can access the latest versions easily.

3. Software Requirements Specifications

3.1. System Stakeholders and Requirement Sources

3.1.1. System Stakeholders

1. **Operational Stakeholders**

- End Users
  - Residents, Students, Employees.
  - Invdiviuals who report maintenance issues through the application.
  - Their main concerns are ease of use, privacy and clear updates with minimum delay.

- Techinicians
  - Invdiviuals who resolve maintenance requests assigned by the admin.
  - Their main concerns are clear task assignments, prioritization, smooth workflow and centralization.

2. **Internal Stakeholders**

- Maintenance Administrators
  - Invdiviuals who oversee the whole maintenance workflow.
  - Their main concerns are efficiency, resource optimization and centralization.

- Project Manager
  - Resposible for overall planning and coordination and project execution.
  - Main concerns are great communication, maintaining quality standards and ensuring project completion.

- System Administrators
  - Invdiviuals who are responsible for the systems infrastructure, backups, user access controls and system performance.
  - Their main concerns are to ensure data integrity and smooth operation through out all the application processes.

3. **Excutive Stakeholders**
   - Executive Management / Facility Management Leadership
     - Use system reports and analytics for decision-making.
     - Their main concerns are efficiency, cost control, and performance monitoring.

4. **External Stakeholders**
   - Regulatory Authorities
     - Ensuring that the system complies with national laws.
   - Institution Responsible for Maintenance Services
     - Funds and authorizes the development of the system.
     - Main concerns are return on investment, system reliability, and long-term sustainability.

3.1.2. Information Gathering

To make sure the project's vision is well understoond and defined, while also having the culture in mind, we will preform a set of techniques in our search for information.

3.1.2.1. Interviews

Table 16: Operational Stakeholders Interviews

End Users (Residents, Students, Employees)	
Question	**Question Type**
How do you currently report maintenance issues?	open-ended
Do you find the current maintenance reporting process easy to use?	close-ended: yes or no
What difficulties do you face when submitting a maintenance request?	open-ended
How important is privacy when reporting maintenance issues?	scale
Do you receive timely updates about the status of your requests?	close-ended: yes or no
What type of notifications or updates would you prefer to receive?	open-ended
How satisfied are you with the response time for maintenance issues?	scale
Would you use a centralized application for all maintenance-related issues?	close-ended: yes or no
Technicians	
Question	**Question Type**
How are maintenance tasks currently assigned to you?	open-ended
Are task priorities clearly defined when you receive assignments?	close-ended: yes or no
What information do you need most to complete a maintenance task efficiently?	open-ended
How easy is it to track your assigned tasks and their status?	scale
Do you face delays due to unclear or incomplete requests?	close-ended: yes or no
What features would improve your daily maintenance workflow?	open-ended
Would a centralized system improve communication between you and administrators?	close-ended: yes or no

Table 17: Internal Stakeholders Interviews

Maintenance Administrators	
How do you currently manage and track maintenance requests?	open-ended
Do you find it difficult to prioritize maintenance tasks?	close-ended: yes or no
How effective is the current system in allocating technicians and resources?	scale
What challenges do you face in overseeing the entire maintenance workflow?	open-ended
Do you require reporting or analytics to support decision-making?	close-ended: yes or no
What type of reports would be most useful for you?	open-ended
Would automation help reduce your workload?	close-ended: yes or no
Project Manager	
Question	**Question Type**
What are the key objectives you expect this system to achieve?	open-ended
Do you believe the current maintenance process meets project goals?	close-ended: yes or no
How important is cross-team communication in this project?	scale

What risks do you foresee in implementing this system?	open-ended
Do you require milestone tracking and progress reports?	close-ended: yes or no
What indicators would you use to measure project success?	open-ended

Table 18: Executive Stakeholder Interviews

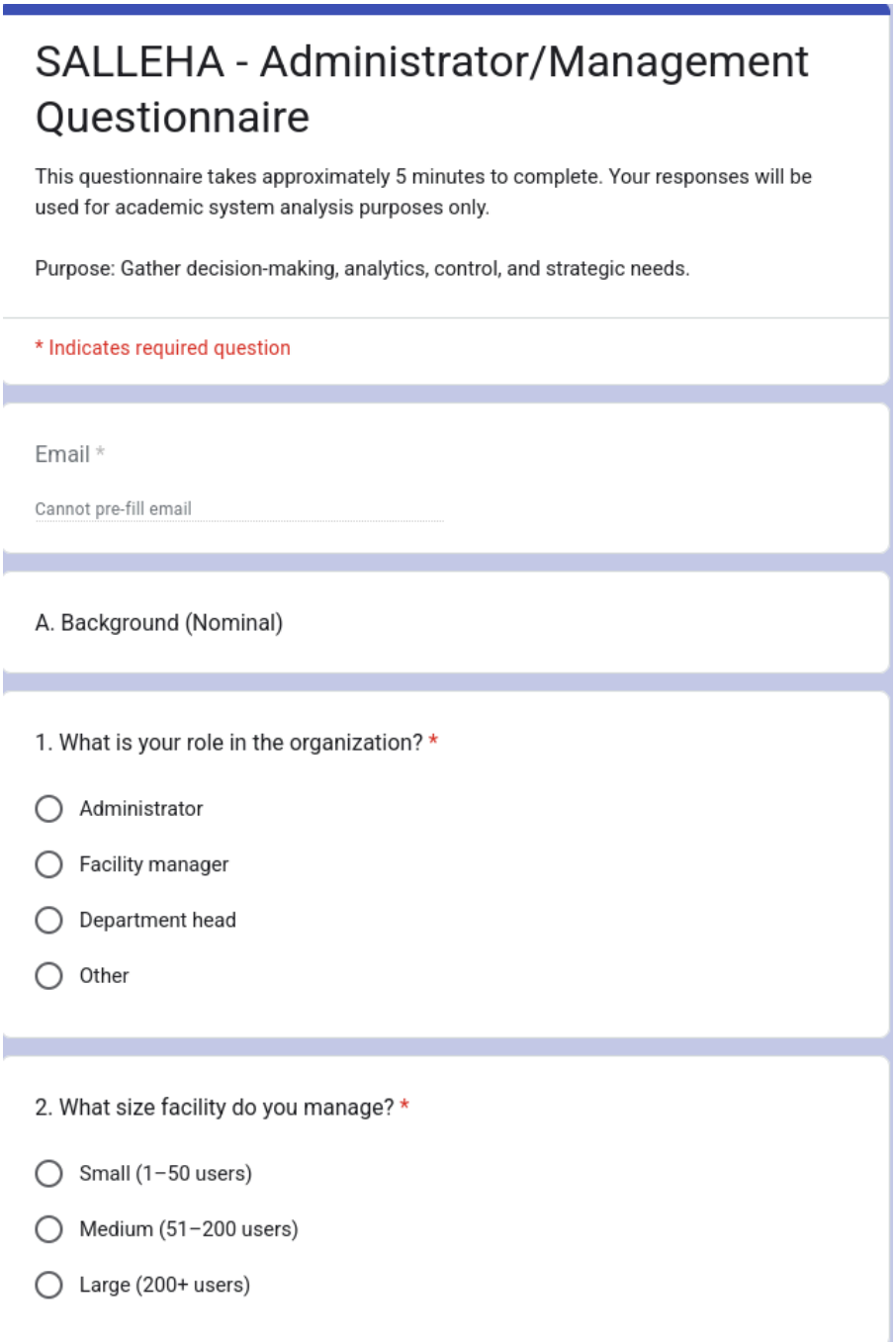
Executive Management / Facility Management Leadership	
Question	**Question Type**
How do you currently evaluate maintenance performance?	open-ended
Do you rely on reports and analytics for decision-making?	close-ended: yes or no
How important is cost efficiency in maintenance operations?	scale
What key performance indicators would you like to monitor?	open-ended
Would real-time dashboards improve strategic oversight?	close-ended: yes or no
System Administrators	
Question	**Question Type**
What systems or tools are currently used to manage maintenance data?	open-ended
Is data security a major concern for this system?	close-ended: yes or no
How critical is system availability and uptime?	scale
What access control mechanisms are required for different users?	open-ended
Do you require regular backups and recovery mechanisms?	close-ended: yes or no
What performance issues do you anticipate as the system scales?	open-ended

Table 19: External Stakeholder Interview

Regulatory Authorities	
Question	**Question Type**
What regulations must this system comply with?	open-ended
Is data protection compliance mandatory for this system?	close-ended: yes or no
How strict are reporting and audit requirements?	scale
What documentation or logs are required for compliance checks?	open-ended

3.1.2.2. Questionnaires

- **Administrator Questionnaire**



SALLEHA - Administrator/Management Questionnaire

This questionnaire takes approximately 5 minutes to complete. Your responses will be used for academic system analysis purposes only.

Purpose: Gather decision-making, analytics, control, and strategic needs.

* Indicates required question

Email *

Cannot pre-fill email

A. Background (Nominal)

1. What is your role in the organization? *

- Administrator
- Facility manager
- Department head
- Other

2. What size facility do you manage? *

- Small (1–50 users)
- Medium (51–200 users)
- Large (200+ users)

Figure 6: Admin Questionnaire (Nominal)

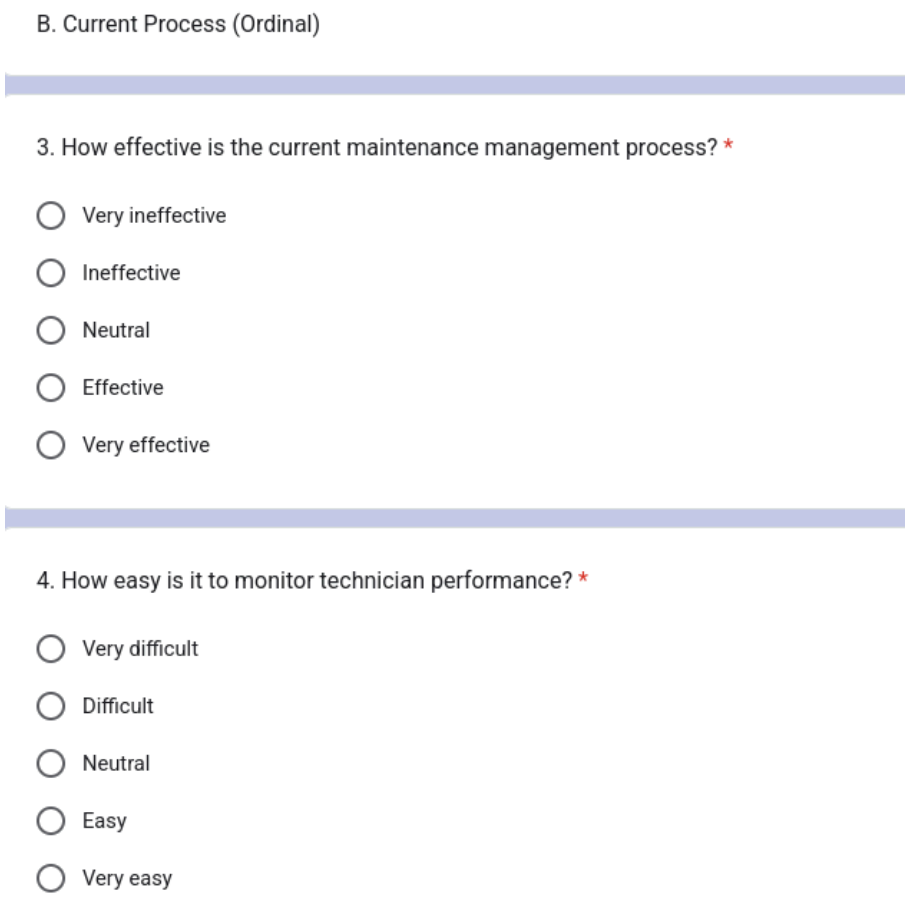


Figure 7: Admin Questionnaire (Ordinal)

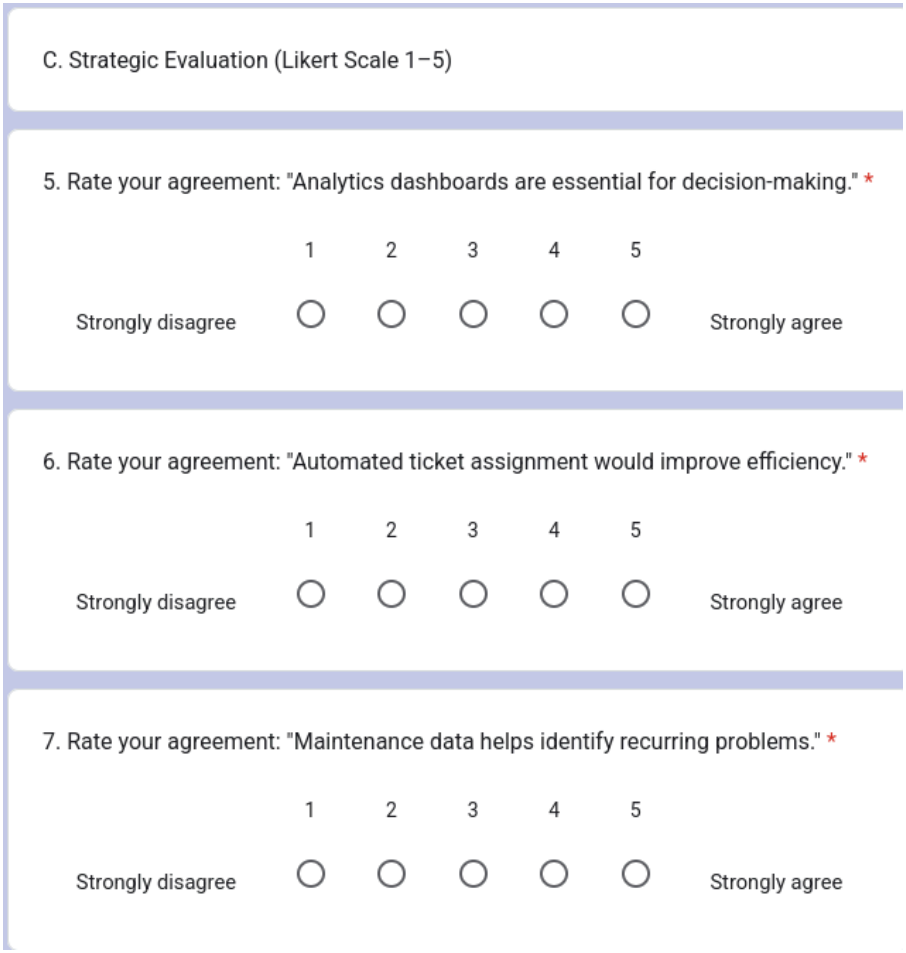


Figure 8: Admin Questionnaire (Interval)

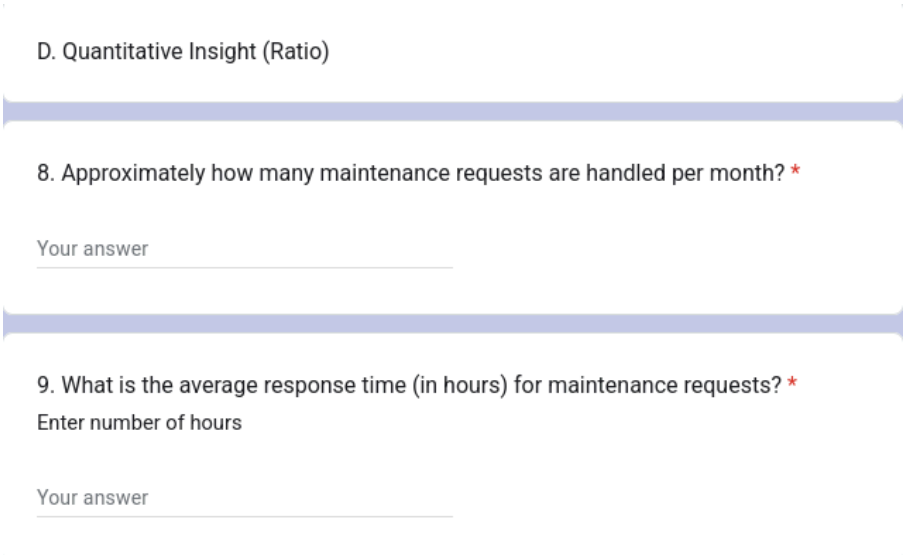


Figure 9: Admin Questionnaire (Ratio)

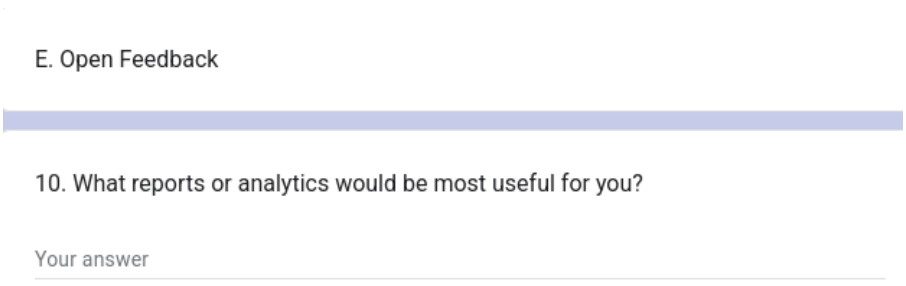
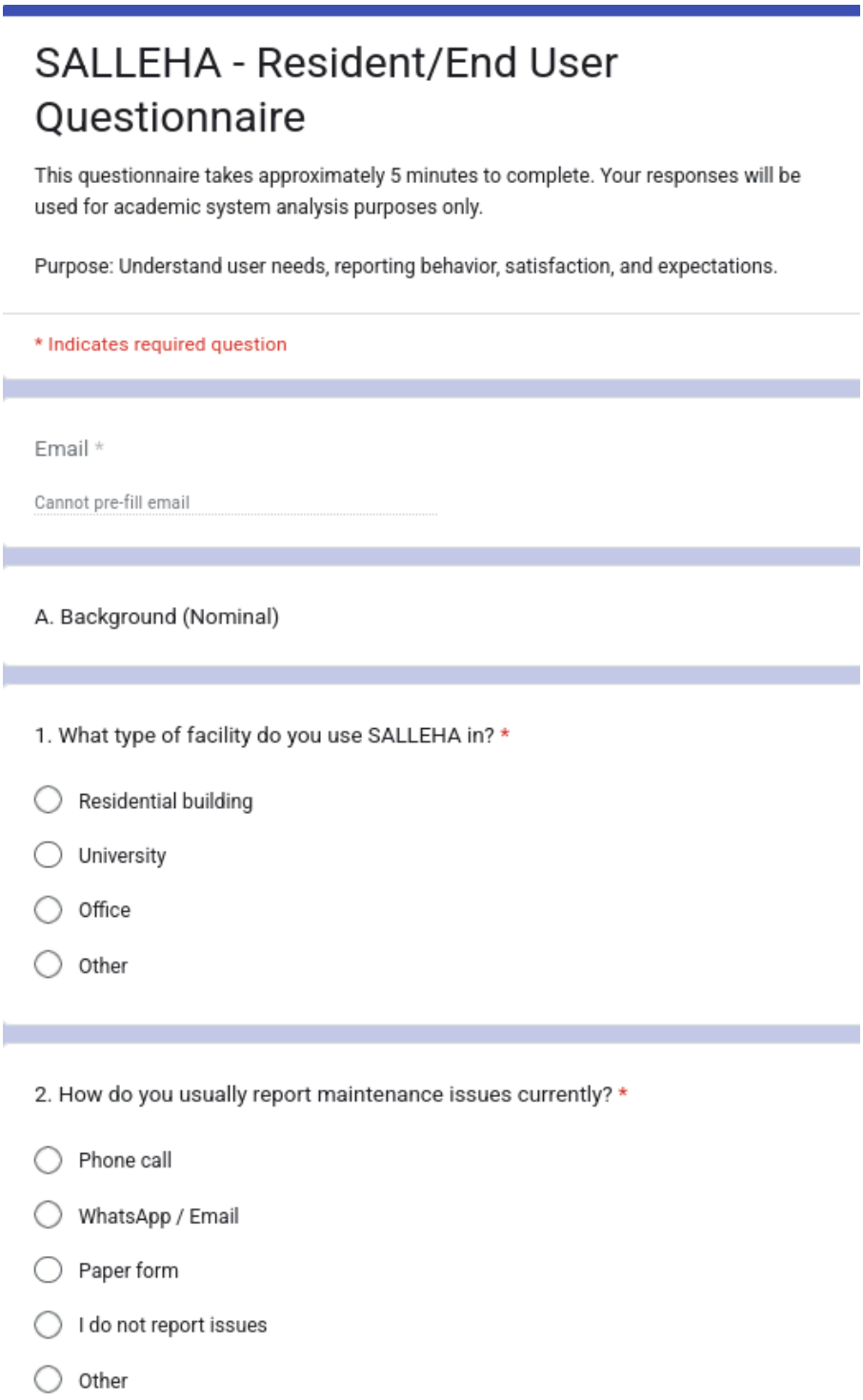


Figure 10: Admin Questionnaire (Open-ended)

- **Resident Questionnaire**



SALLEHA - Resident/End User Questionnaire

This questionnaire takes approximately 5 minutes to complete. Your responses will be used for academic system analysis purposes only.

Purpose: Understand user needs, reporting behavior, satisfaction, and expectations.

* Indicates required question

Email *

Cannot pre-fill email

A. Background (Nominal)

1. What type of facility do you use SALLEHA in? *

- Residential building
- University
- Office
- Other

2. How do you usually report maintenance issues currently? *

- Phone call
- WhatsApp / Email
- Paper form
- I do not report issues
- Other

Figure 11: Resident Questionnaire (Nominal)

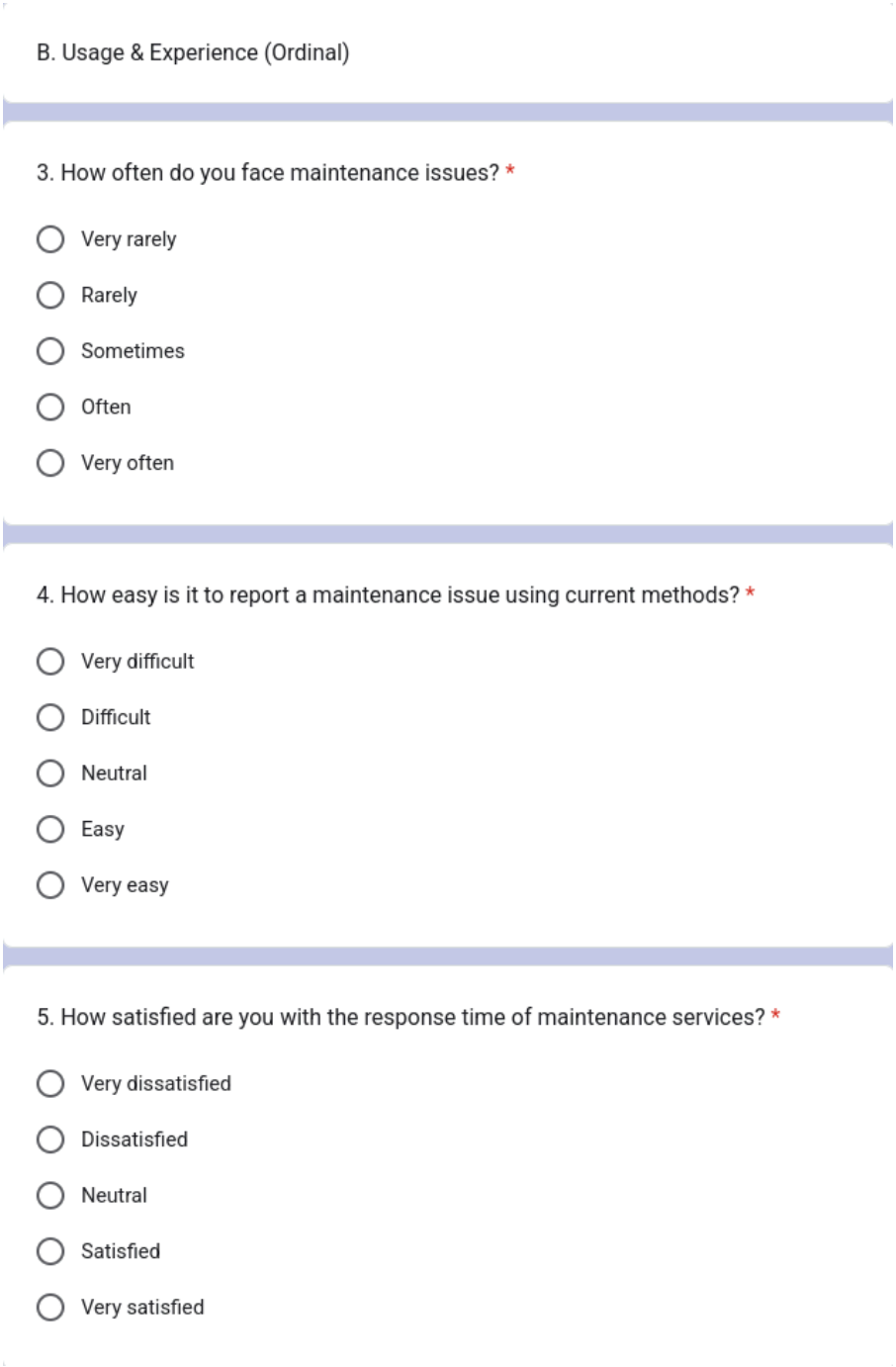


Figure 12: Resident Questionnaire (Ordinal)

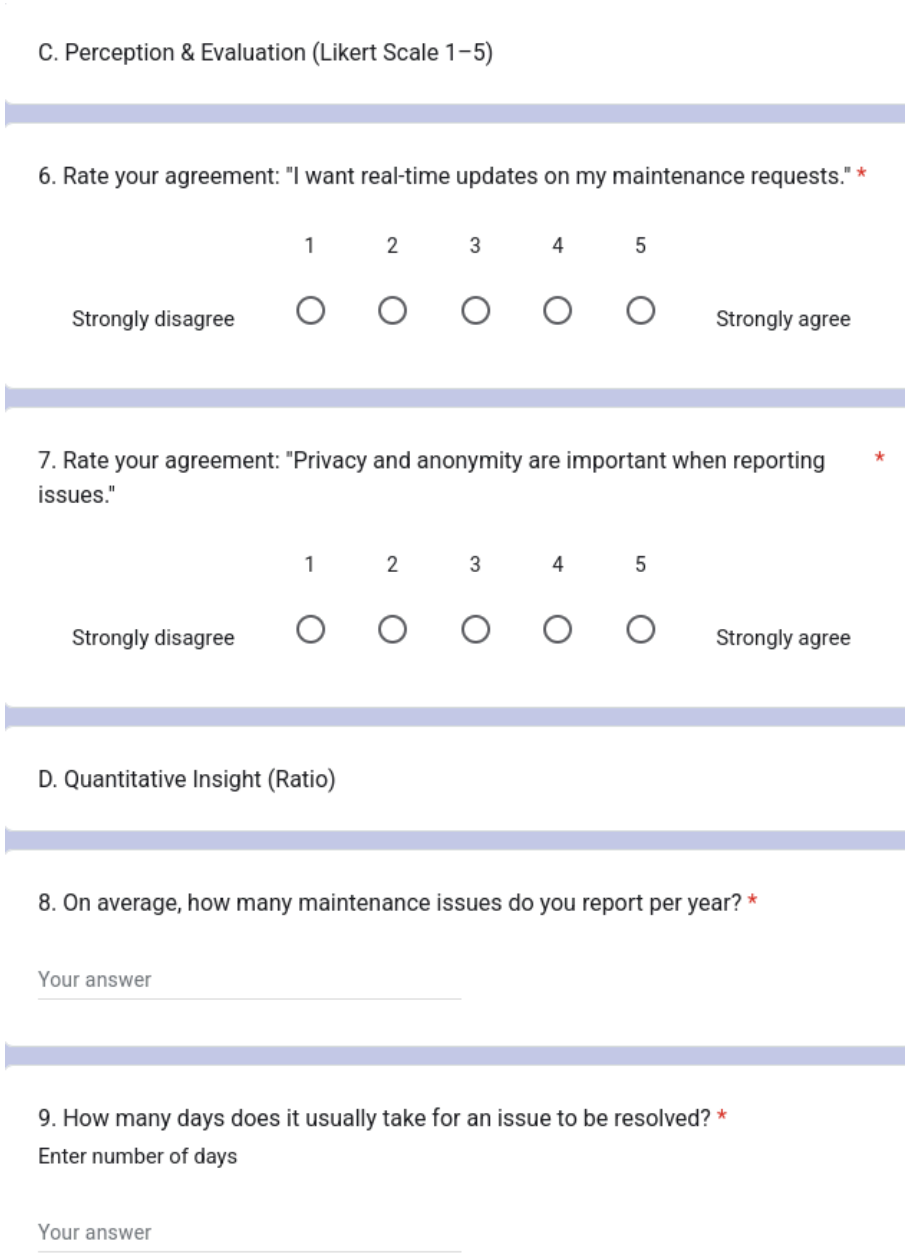


Figure 13: Resident Questionnaire (Interval and Ratio)

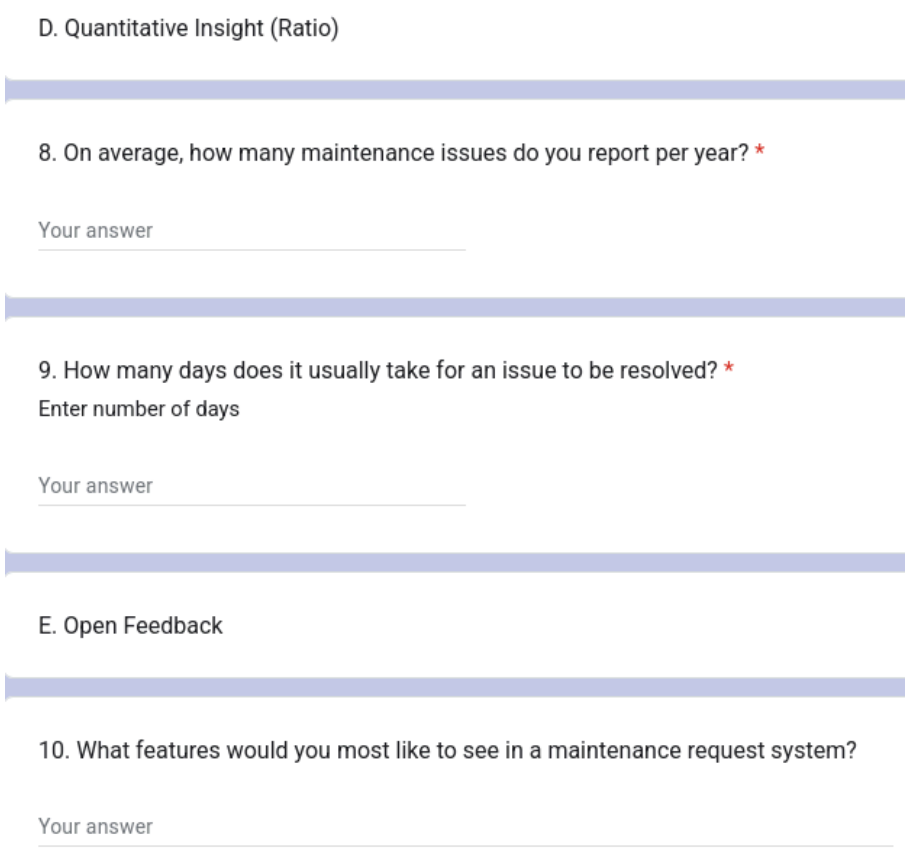
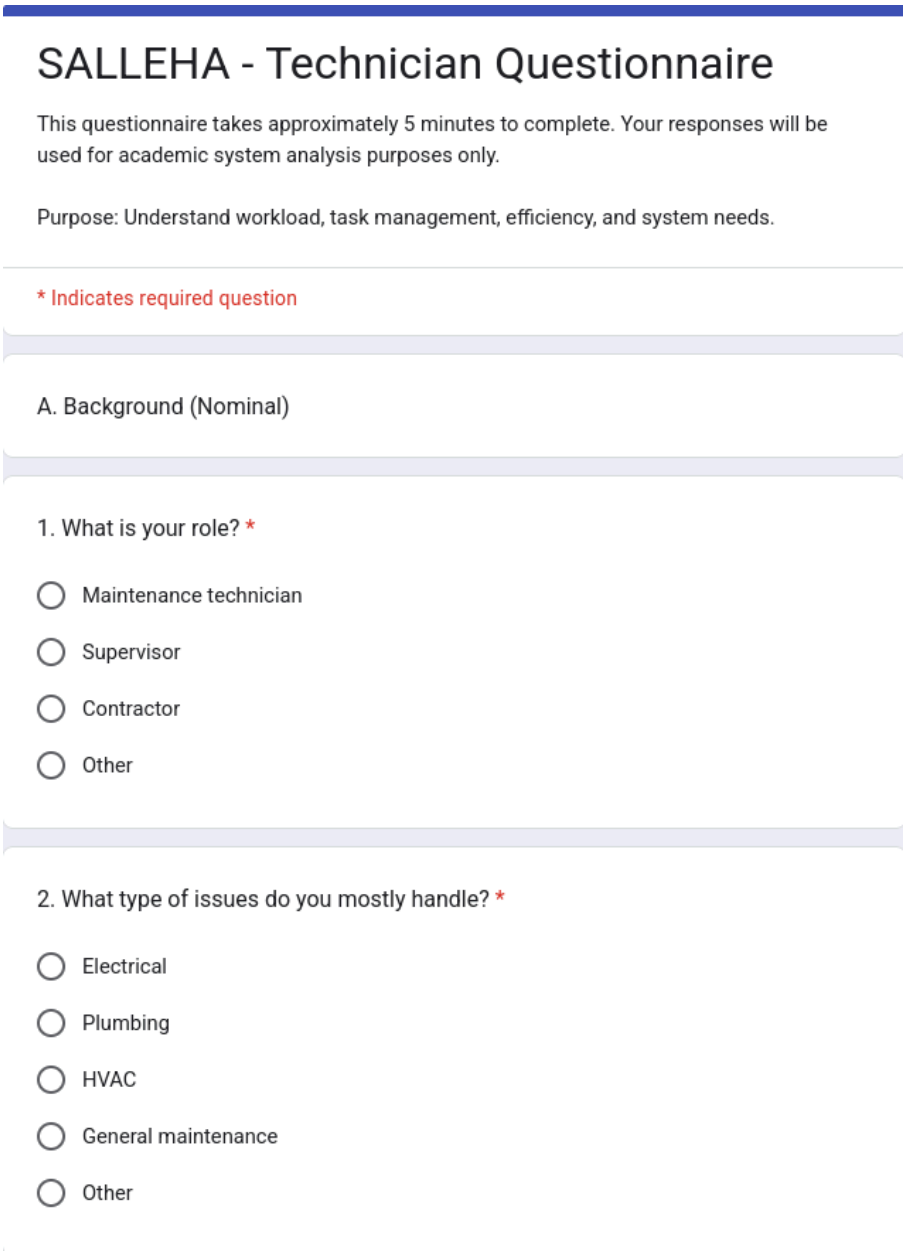


Figure 14: Resident Questionnaire (Open-ended)

- **Technician Questionnaire**



SALLEHA - Technician Questionnaire

This questionnaire takes approximately 5 minutes to complete. Your responses will be used for academic system analysis purposes only.

Purpose: Understand workload, task management, efficiency, and system needs.

* Indicates required question

A. Background (Nominal)

1. What is your role? *

- Maintenance technician
- Supervisor
- Contractor
- Other

2. What type of issues do you mostly handle? *

- Electrical
- Plumbing
- HVAC
- General maintenance
- Other

Figure 15: Technician Questionnaire (Nominal)

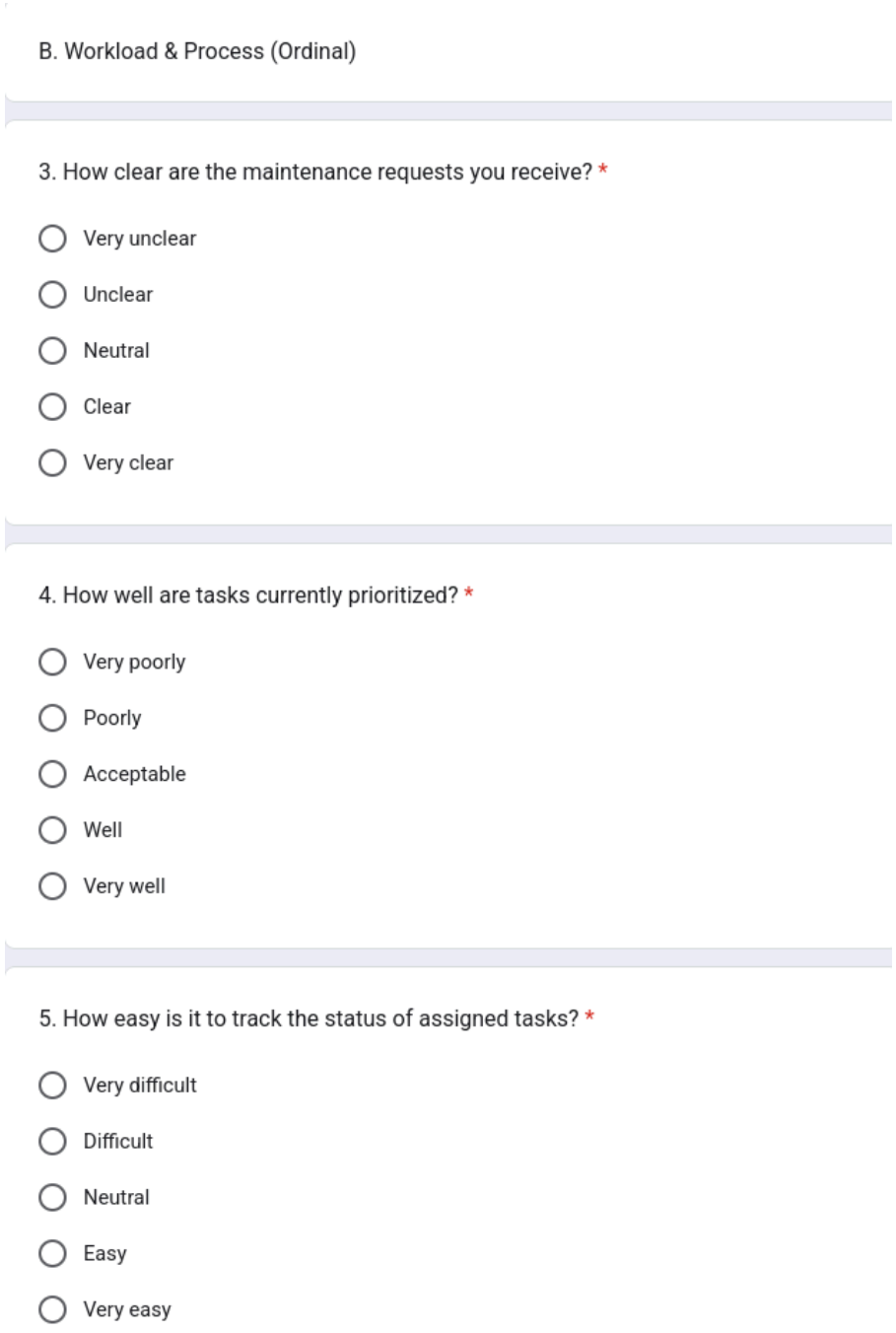


Figure 16: Technician Questionnaire (Ordinal)

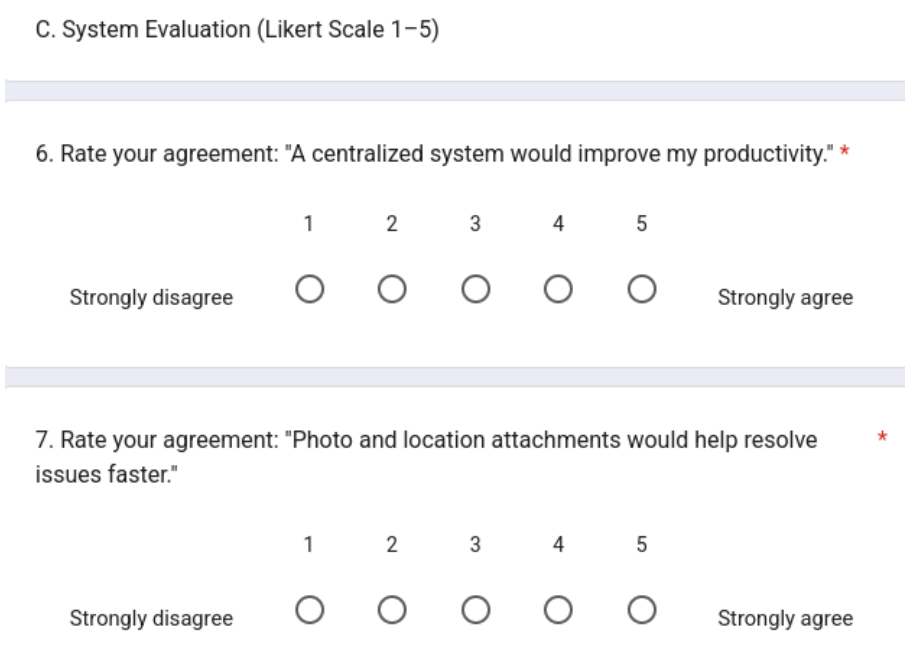


Figure 17: Technician Questionnaire (Interval)

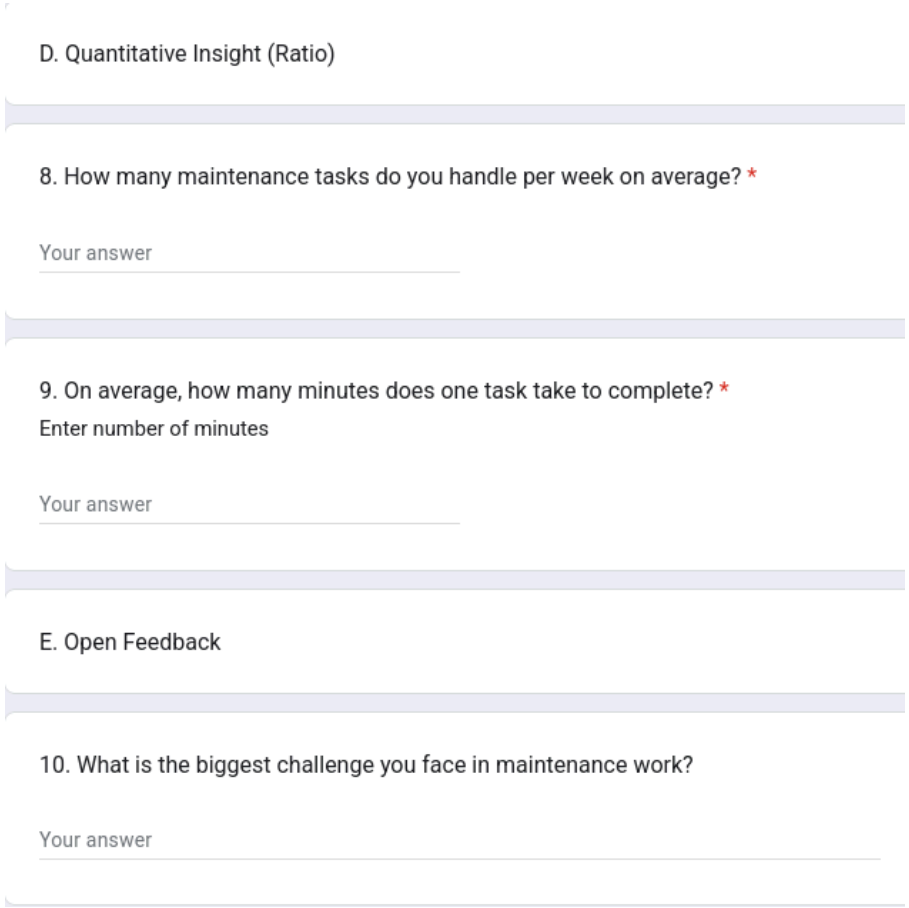


Figure 18: Technician Questionnaire (Ratio and Open-ended)

3.1.2.3. Document Analysis

In this project, we systematically reviewed a range of institutional maintenance-related documents to gain a deeper understanding of stakeholder needs. These included historical maintenance service

records, internal workflow reports, facility management evaluations, and previous user feedback from residents, students, and staff. This analysis helped identify inefficiencies, recurring issues, and key areas for improvement within the existing maintenance process. Additionally, it enabled us to uncover implicit requirements not directly expressed by stakeholders, ensuring that the proposed application addresses operational, usability, and performance needs effectively.

3.1.2.4. Observation

Based on the analysis of the questionnaire responses, it was observed that a significant number of participants expressed dissatisfaction with the current organization of maintenance services. Respondents frequently reported unclear reporting procedures, lack of prioritization, and delays caused by poor coordination between involved parties. Many users indicated that they often feel stressed or uncertain due to the absence of timely updates regarding the status of their requests. The findings suggest that implementing a centralized maintenance management system would improve organization, enhance transparency, and ensure continuous status updates. Such a system would reduce user stress, improve communication, and increase overall satisfaction by providing a more structured and reliable maintenance process.

3.1.2.5. Prototype

- **Authentication Screens**

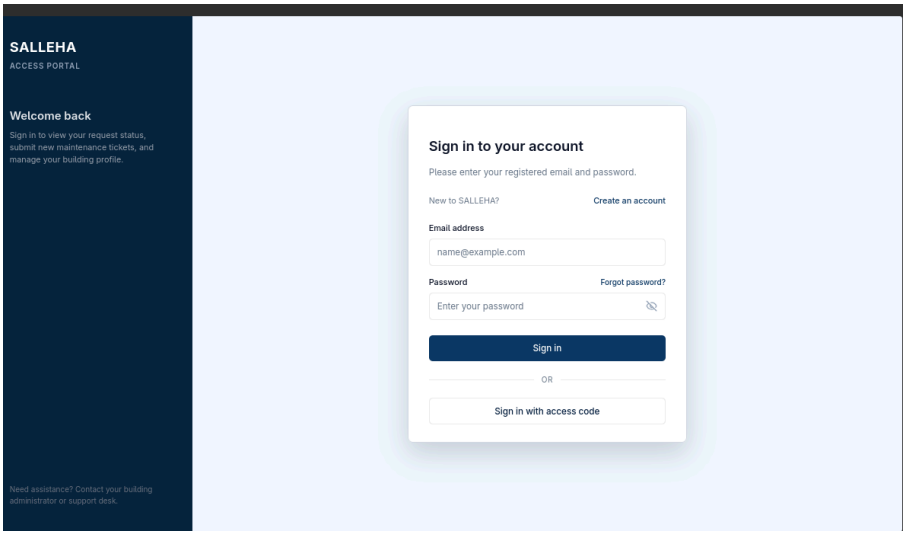


Figure 19: Login Screen

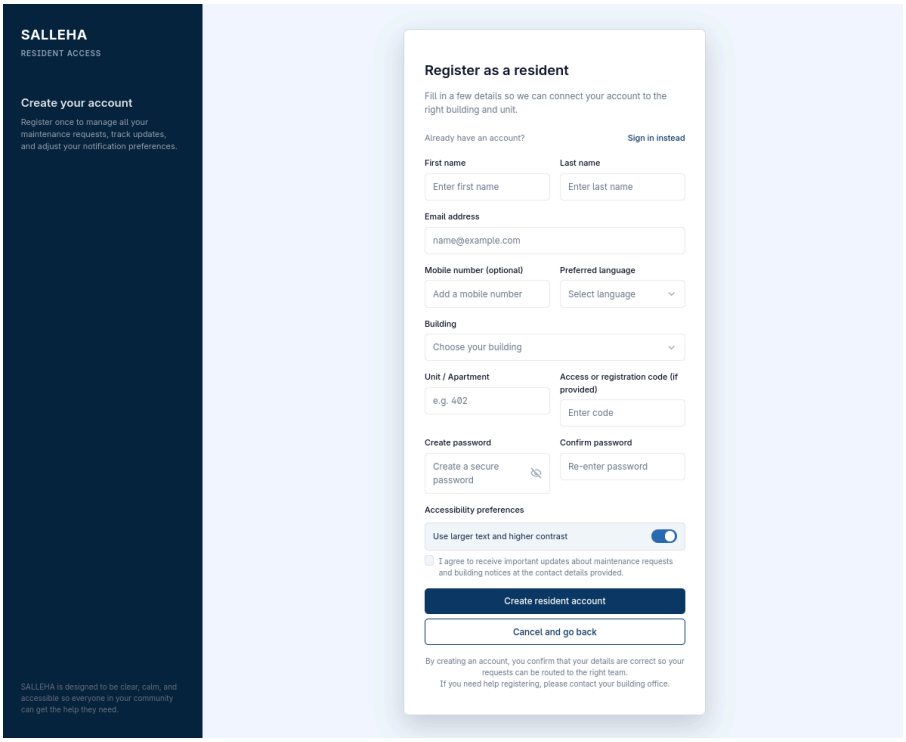


Figure 20: Registration Screen

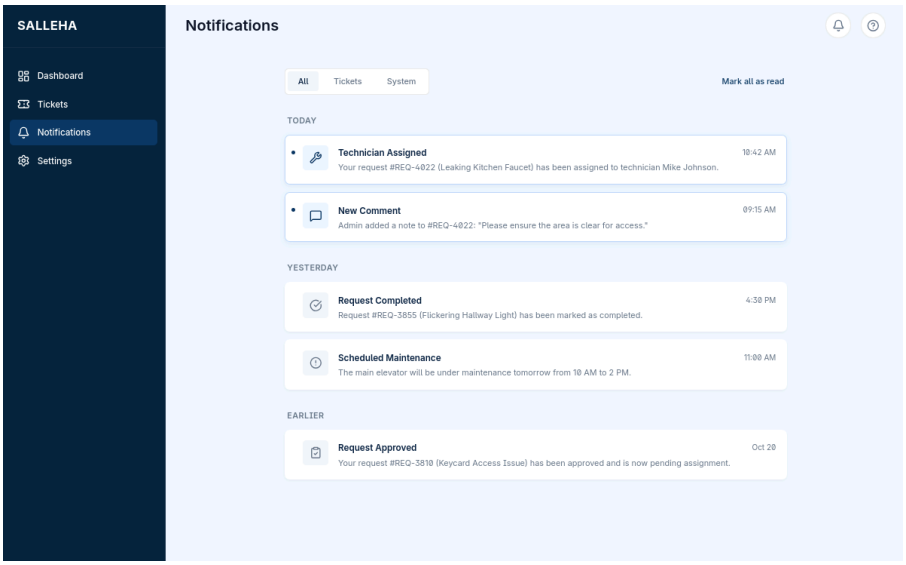


Figure 21: Notifications Screen

- **Resident Prototype Screens**

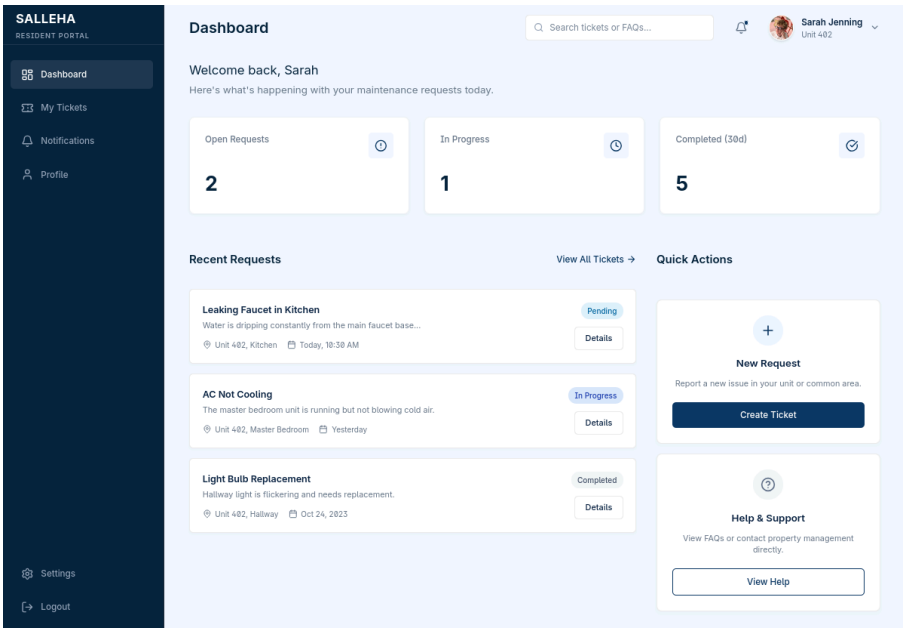


Figure 22: Resident – Dashboard

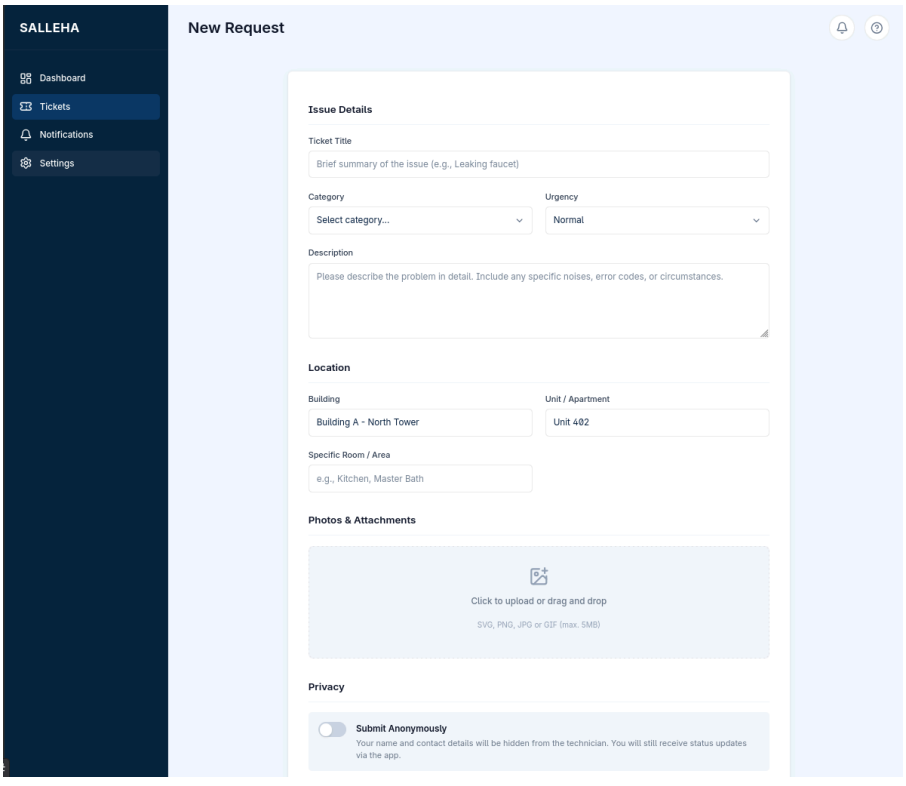


Figure 23: Resident – Ticket Request

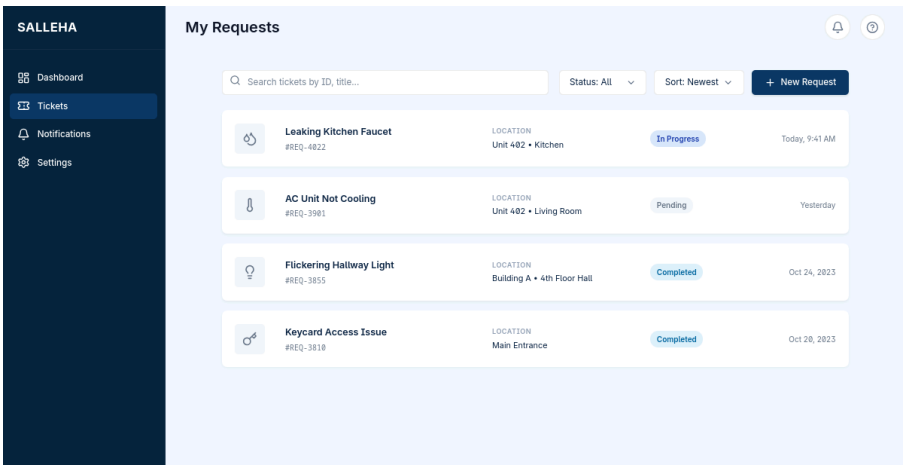


Figure 24: Resident – Tickets Screen

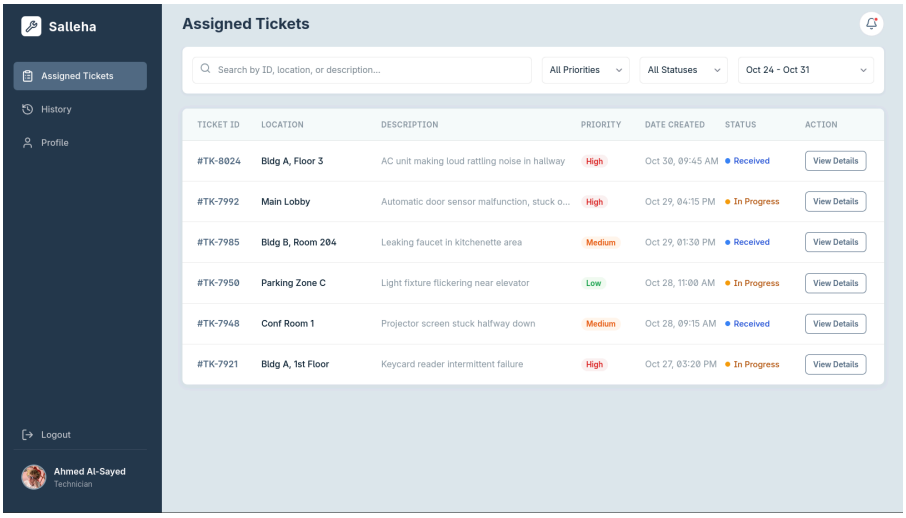


Figure 25: Technician – Dashboard

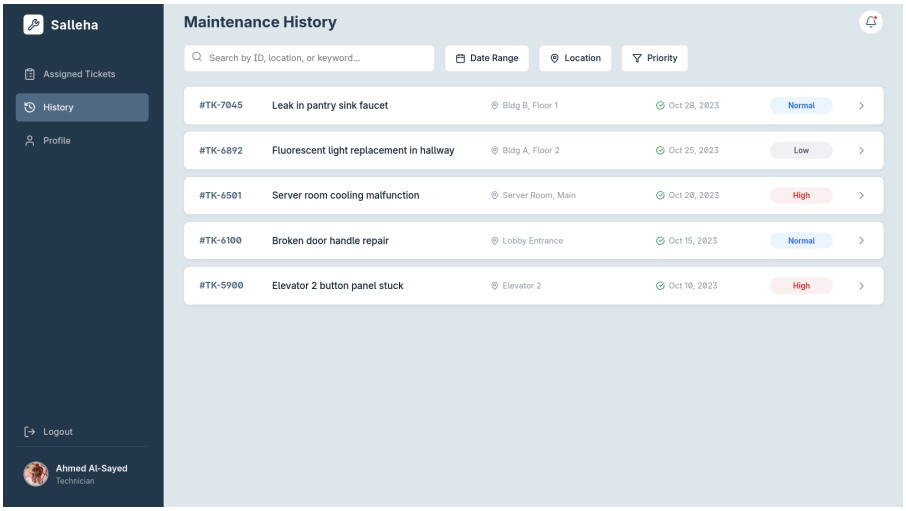


Figure 26: Technician – Maintenance History

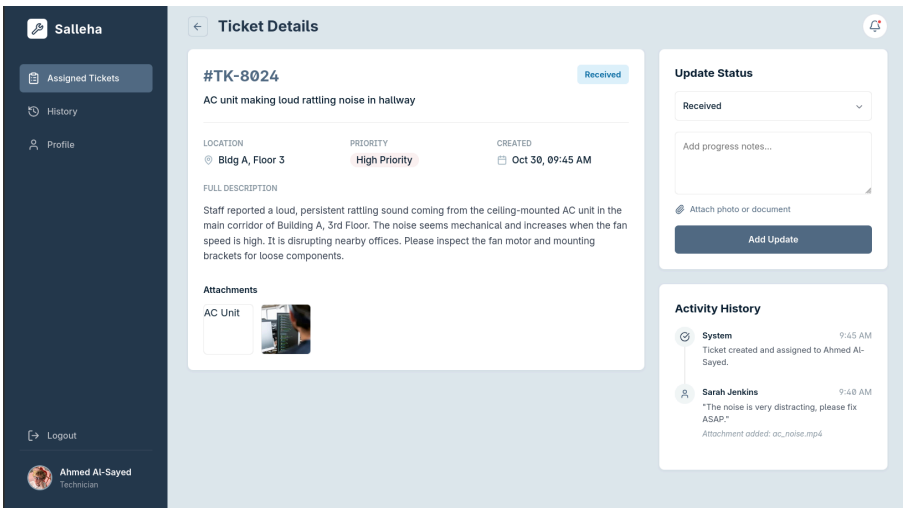


Figure 27: Technician – Ticket Details

- **Administrator Prototype Screens**

Figure 28: Administrator – Dashboard

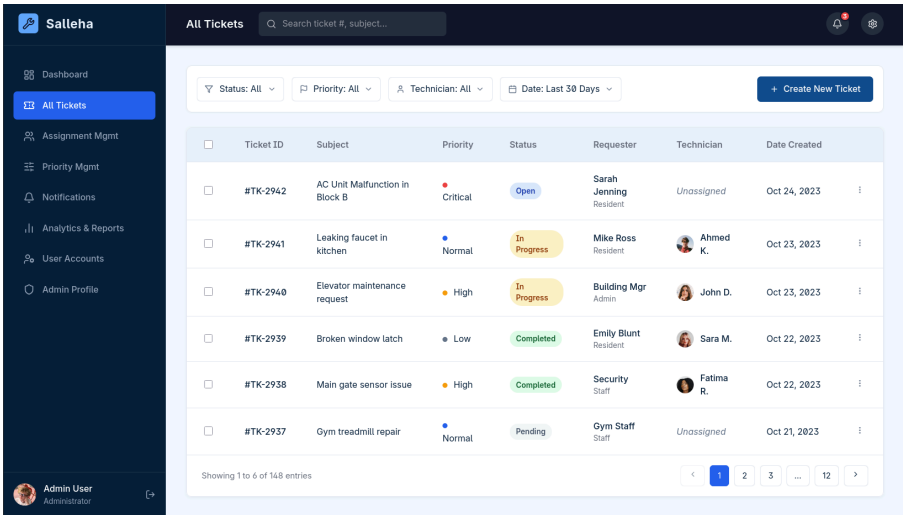


Figure 29: Administrator – All Tickets

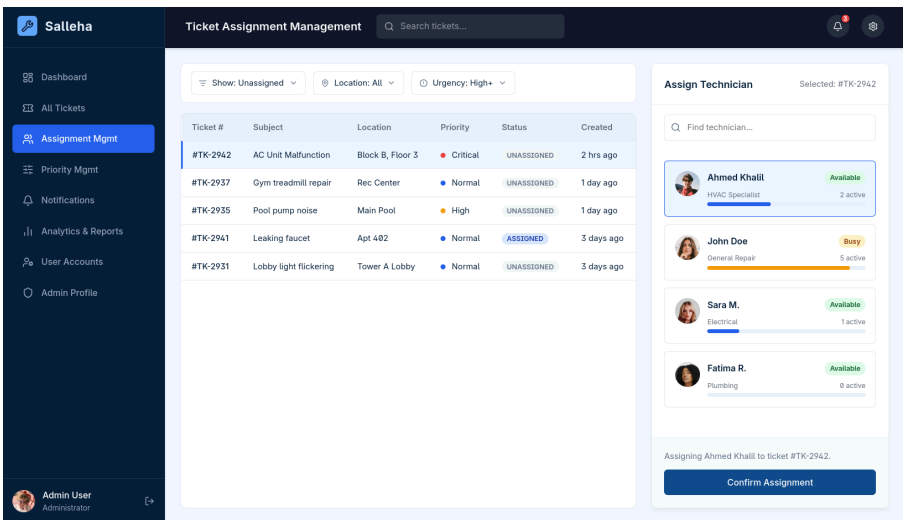


Figure 30: Administrator – Assignment Management

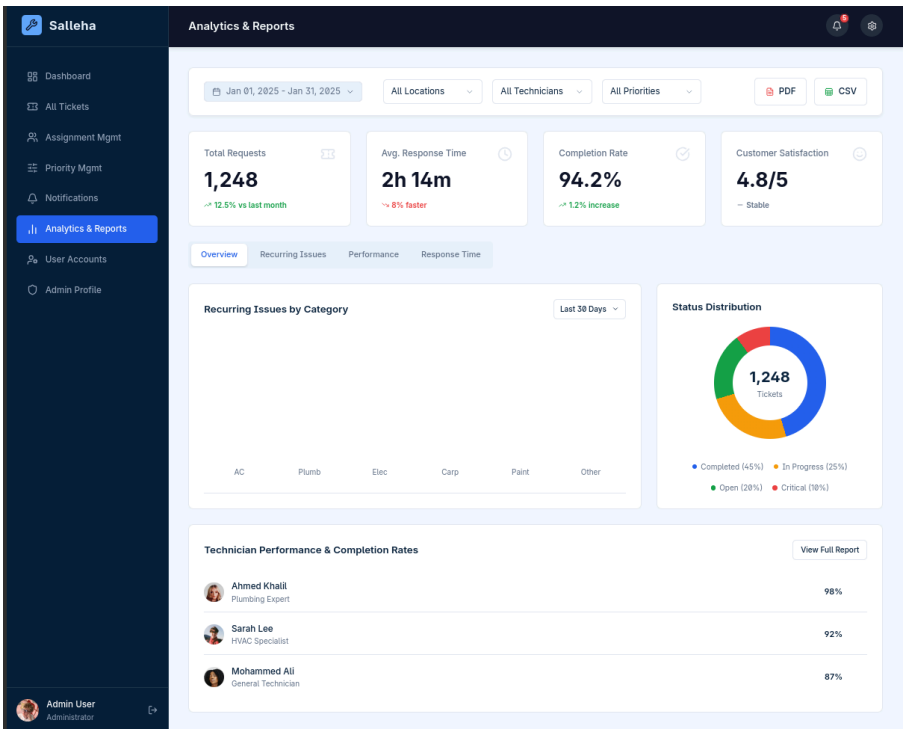


Figure 31: Administrator – Analytics & Reports

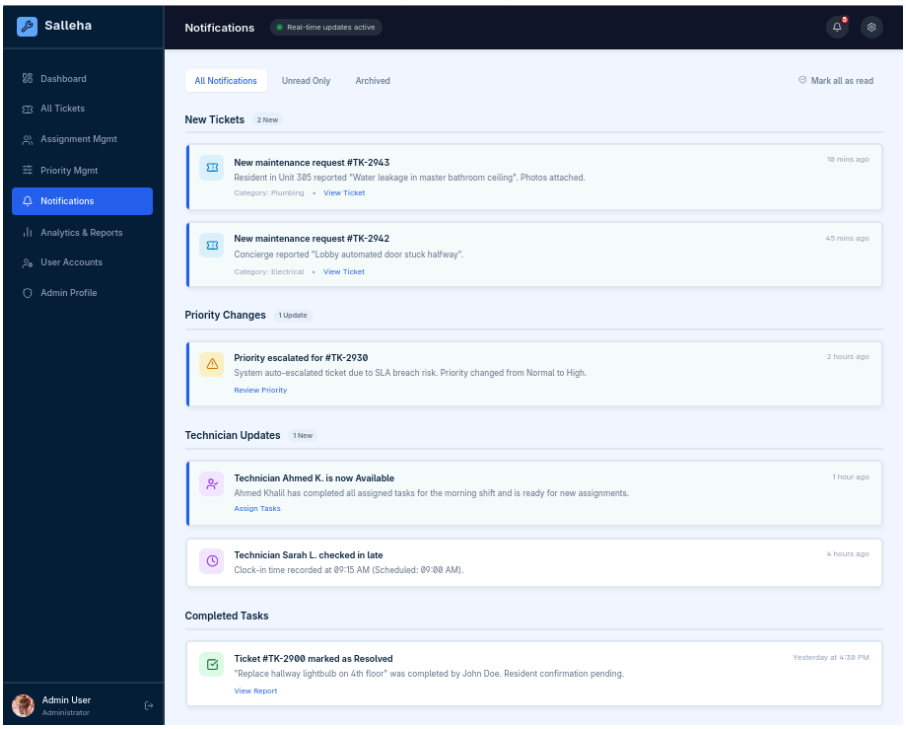


Figure 32: Administrator – Notifications

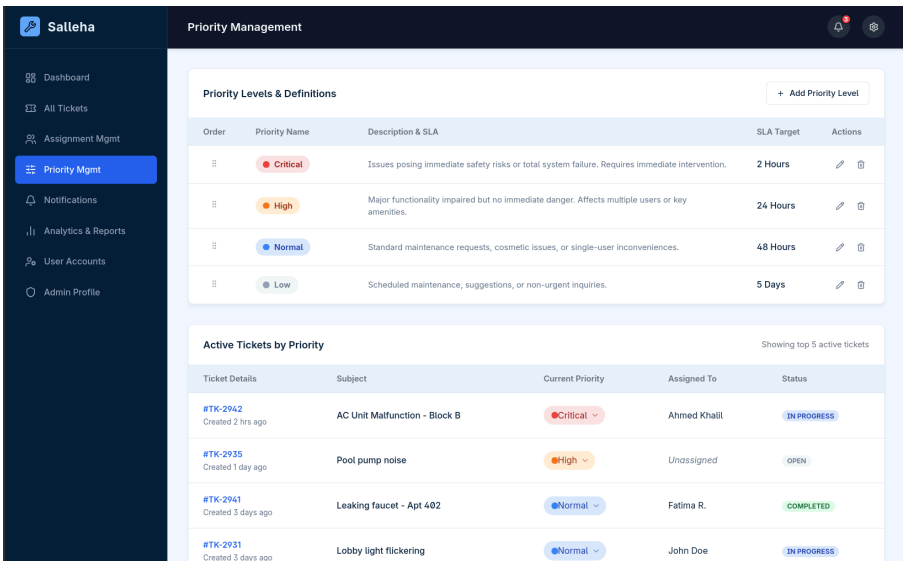


Figure 33: Administrator – Priority Management

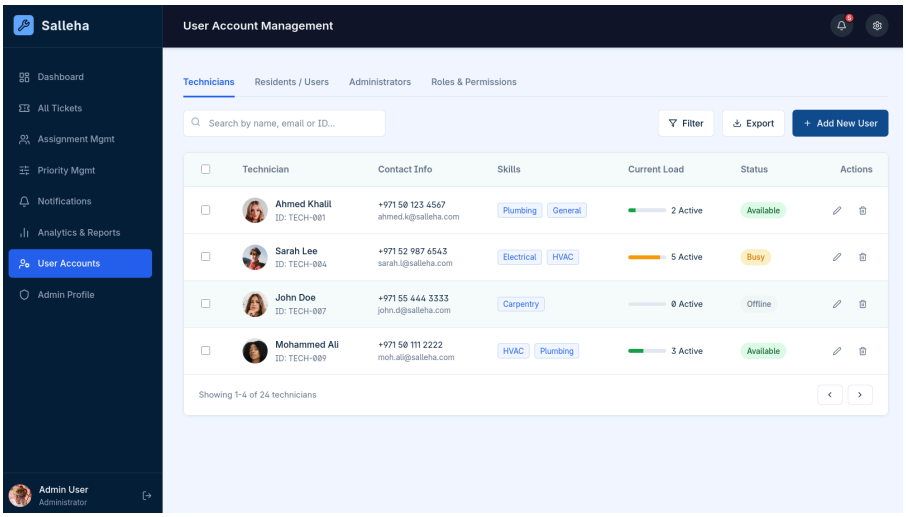


Figure 34: Administrator – User Accounts Management

3.2. User Requirements

Table 20: High level Functional Requirements

FR ID	Function Name	Description	User/Role
FR-1	User Registration	The system shall provide a registration interface where users can create an account by entering full name, email, national ID, username, password, and selecting their role. All fields shall be validated.	All Users
FR-2	User Login	The system shall allow users to log in using only their username and password. The system shall automatically retrieve the user's role from the database and redirect to the appropriate dashboard.	All Users

FR-3	Logout	The system shall provide a logout option accessible from the navigation menu that securely ends the session and redirects to the login page.	All Users
FR-4	Password Recovery	The system shall provide a password recovery mechanism using registered email and national ID with a one-time verification code.	All Users
FR-5	Resident Dashboard	Upon login, residents shall see a dashboard displaying an overview of reported tickets and their current statuses.	Resident
FR-6	Resident Navigation Menu	Residents shall have a navigation menu containing Home, Report Status, Open Ticket, Notifications, Profile.	Resident
FR-7	View Reports Feed	Residents shall view a feed of reported maintenance issues with ticket title, category, location, and status (Open, In Progress, Fixed).	Resident
FR-8	Open Maintenance Ticket	Residents shall open a maintenance ticket with issue category, description, location, priority, and optional images.	Resident
FR-9	Track Ticket Status	Residents shall track the real-time status of their submitted tickets.	Resident
FR-10	Notifications for Residents	Residents shall receive notifications when ticket status changes.	Resident
FR-11	Avoid Duplicate Reports	Residents shall see existing reports to avoid duplicate ticket submissions.	Resident
FR-12	Profile Management	Residents shall view and edit their profile, excluding their role.	Resident
FR-13	Technician Dashboard	Technicians shall see a dashboard showing assigned tasks, pending requests, and statuses.	Technician
FR-14	Technician Navigation Menu	Technicians shall have a navigation menu with Home, Assigned Tasks, Maintenance History, Notifications, Profile.	Technician
FR-15	Accept Maintenance Requests	Technicians shall view and accept maintenance requests assigned by the system or administrators.	Technician
FR-16	Update Task Status	Technicians shall update task status (Open, In Progress, Fixed) as work progresses.	Technician

FR-17	Upload Maintenance Evidence	Technicians shall upload images or notes as evidence of maintenance completion.	Technician
FR-18	Technician Notifications	Technicians shall receive notifications when a new task is assigned or updated.	Technician
FR-19	View Maintenance History	Technicians shall view maintenance history, including equipment and location details.	Technician
FR-20	Technician Analytics	Technicians shall see basic analytics for completed tasks, equipment, and areas.	Technician
FR-21	Admin Dashboard	Administrators shall see a dashboard displaying total tickets, open issues, resolved issues, and technician workload.	Administrator
FR-22	Admin Navigation Menu	Administrators shall have a navigation menu with Home, Ticket Management, Technician Management, Analytics and Reports, Profile.	Administrator
FR-23	Assign Tickets	Administrators shall assign maintenance tickets to technicians based on priority, availability, and expertise.	Administrator
FR-24	Set Ticket Priority	Administrators shall set and modify ticket priority levels.	Administrator
FR-25	Admin Notifications	Administrators shall receive notifications of new tickets, overdue tasks, and completed maintenance work.	Administrator
FR-26	View Analytics	Administrators shall view analytics dashboards for maintenance trends, frequently reported areas, equipment performance, and task completion times.	Administrator
FR-27	Export Reports	Administrators shall export maintenance reports and analytics in PDF or Excel.	Administrator
FR-28	Manage Users	Administrators shall manage registered users, including activating, deactivating, or updating accounts.	Administrator

3.3. Context Diagram

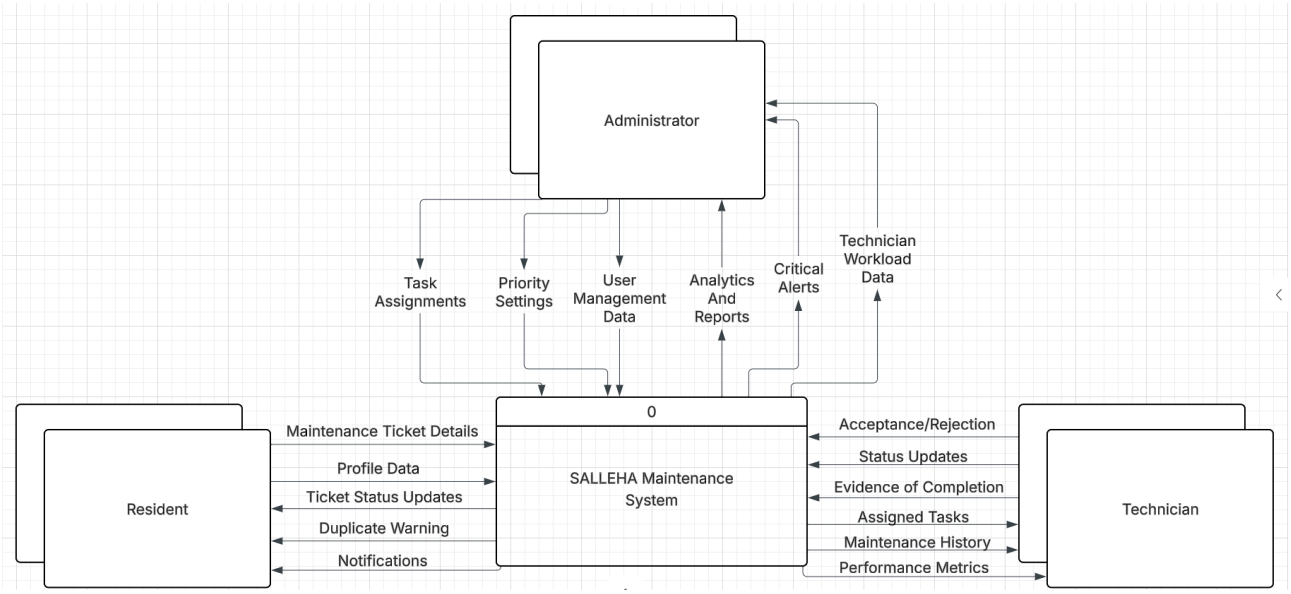


Figure 35: Context Diagram

3.4. Use Case Model

3.4.1. Use Case diagram

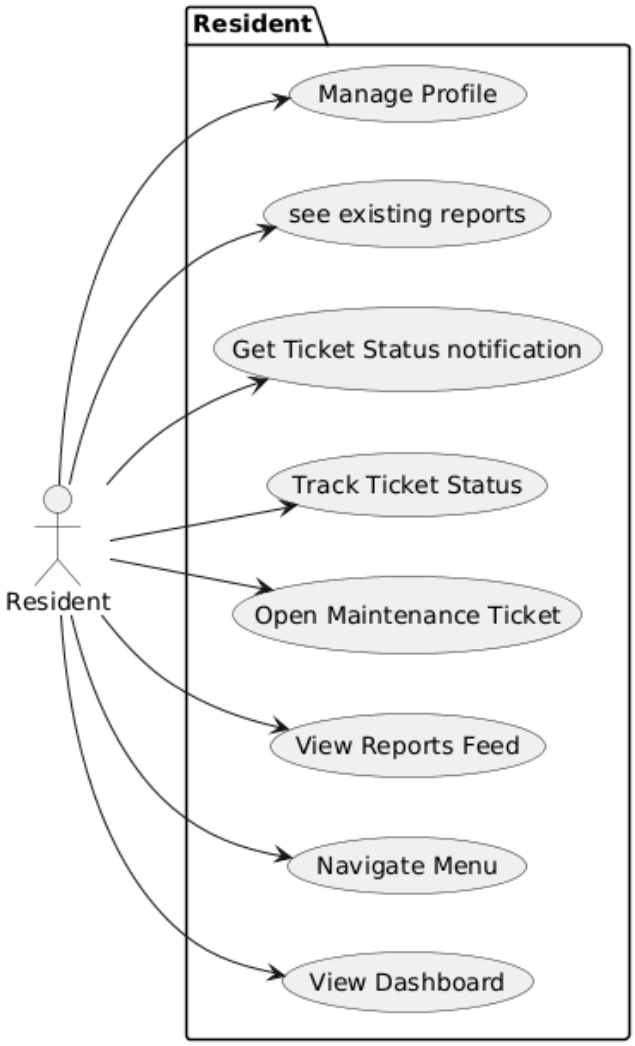


Figure 36: Use Case Diagram - Resident

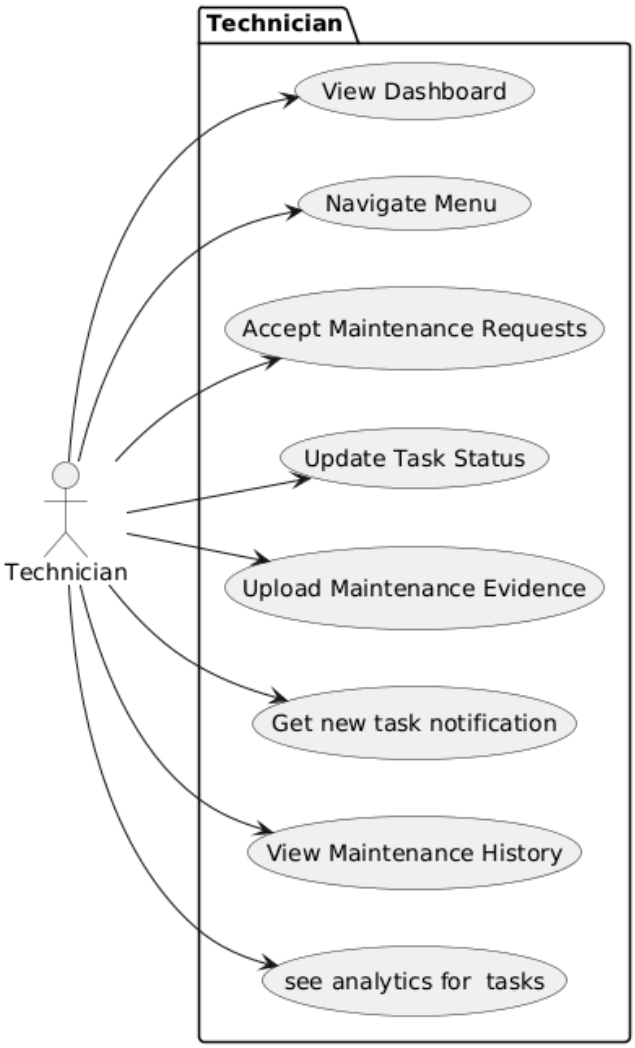


Figure 37: Use Case Diagram - Technician

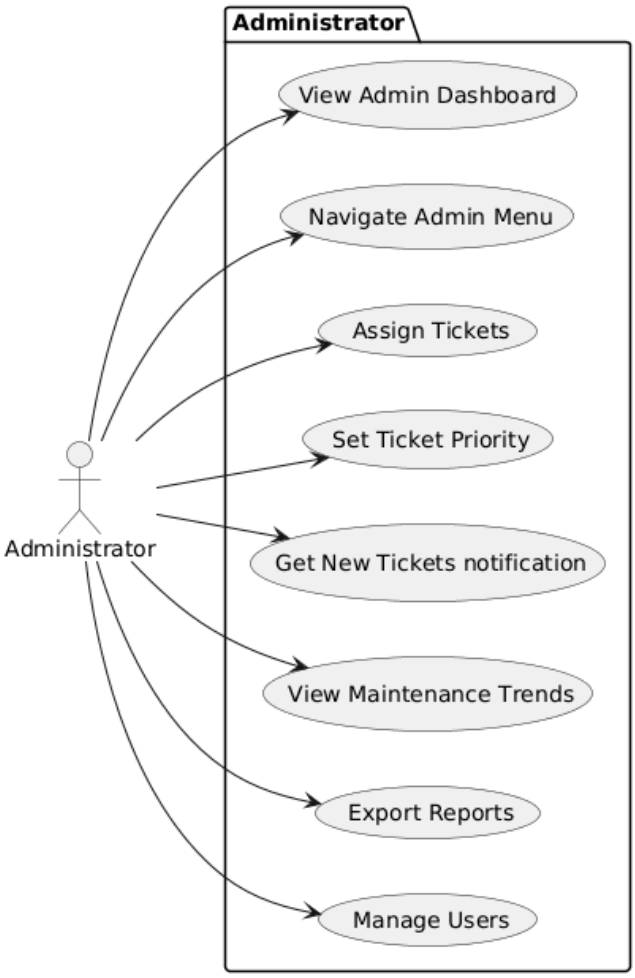


Figure 38: Use Case Diagram - Administrator

3.4.2. Use Case descriptions

Table 21: Use Cases Descriptions

Use Case	Brief Use Case description
Sign Up	Allows users to create an account by entering and validating personal information and role selection.
Sign In	Allows users to log in using username and password and redirects them to the appropriate dashboard.
Reset Password	Allows users to reset their password using email, national ID, and a one-time verification code.
Logout	Allows users to securely end their session and return to the login page.
Manage Profile	Allows users to view and edit their personal profile information.
See existing reports	Displays existing reports to prevent submitting duplicate maintenance tickets.
Get Ticket Status notification	Notifies users when the status of their maintenance tickets changes.
Track Ticket Status	Allows users to monitor the real-time status of submitted maintenance tickets.
Open Maintenance Ticket	Allows users to submit a new maintenance ticket with issue details and optional images.
View Reports Feed	Allows users to view a list of reported maintenance issues with details and status.
Navigate resident Menu	Provides access to system features through a resident navigation menu.
View resident Dashboard	Displays an overview of reported tickets and their current statuses after login.
View Technician Dashboard	Displays assigned tasks, pending requests, and task statuses.
Navigate Technician Menu	Provides technicians with access to task-related features through a navigation menu
Accept Maintenance Requests	Allows technicians to view and accept assigned maintenance tasks.
Update Task Status	Allows technicians to update the status of maintenance tasks as work progresses.
Upload Maintenance Evidence	Allows technicians to upload images or notes as proof of task completion.
Get new task notification	Notifies technicians when tasks are assigned or updated.

View Maintenance History	Allows technicians to view completed maintenance tasks with location and equipment details.
see analytics for tasks	Displays basic analytics related to completed maintenance tasks and areas.
View Admin Dashboard	Displays an overview of tickets, issue statuses, and technician workload
Navigate Admin Menu	Provides administrators with access to management and analytics features.
Assign Tickets	Allows administrators to assign maintenance tickets to technicians based on criteria.
Set Ticket Priority	Allows administrators to set or modify maintenance ticket priority levels.
Get New Tickets notification	Notifies administrators of new, overdue, or completed maintenance tasks.
View Maintenance Trends	Allows administrators to analyze maintenance trends and performance metrics.
Export Reports	Allows administrators to export maintenance reports and analytics in PDF or Excel format.
Manage Users	Allows administrators to activate, deactivate, or update user accounts.

3.5. Functional Requirements Specification

Table 22: FR-1 User Registration

Number	Description
FR-1.1	The system shall allow users to create an account by providing necessary information such as full name, email address, national ID number, username, and password.
FR-1.2	The system shall require users to select their role during registration: Resident, Technician, or Administrator.
FR-1.3	The system shall validate all input fields in real-time, including email format, password strength (minimum 8 characters with uppercase, lowercase, number, and special character), and national ID uniqueness.
FR-1.4	The system shall display clear error messages when validation fails and prevent form submission until all fields are valid.
FR-1.5	The system shall securely hash and store passwords in the database using industry-standard encryption algorithms.
FR-1.6	The system shall send a confirmation email to the user's registered email address with an activation link to verify the account.
FR-1.7	The system shall require email verification before allowing the user to log in to the system.
FR-1.8	The system shall prevent duplicate registrations using the same email address or national ID number.

Table 23: FR-2 User Login

Number	Description
FR-2.1	The system shall provide a login interface where users can enter their username and password.
FR-2.2	The system shall authenticate users against stored credentials in the database.
FR-2.3	The system shall automatically retrieve the user's role from the database upon successful authentication.
FR-2.4	The system shall redirect users to their role-specific dashboard after successful login: Resident Dashboard, Technician Dashboard, or Administrator Dashboard.
FR-2.5	The system shall display a CAPTCHA challenge after 3 consecutive failed login attempts to prevent brute-force attacks.
FR-2.6	The system shall lock the account temporarily after 5 consecutive failed login attempts, requiring administrative intervention or email verification to unlock.
FR-2.7	The system shall maintain user sessions with appropriate timeout periods: 30 minutes of inactivity for Residents, 15 minutes for Technicians, and 60 minutes for Administrators.
FR-2.8	The system shall provide a "Remember Me" option that extends the session duration to 30 days for convenience.

Table 24: FR-3 Logout

Number	Description
FR-3.1	The system shall provide a logout option accessible from the main navigation menu on all pages.
FR-3.2	The system shall immediately terminate the user's session upon logout.
FR-3.3	The system shall clear all session data and authentication tokens.
FR-3.4	The system shall redirect the user to the login page after successful logout.
FR-3.5	The system shall display a confirmation message indicating successful logout.

Table 25: FR-4 Password Recovery

Number	Description
FR-4.1	The system shall provide a "Forgot Password" link on the login page.
FR-4.2	The system shall require users to enter their registered email address and national ID number to initiate password recovery.
FR-4.3	The system shall verify that the email and national ID match a valid user account in the database.
FR-4.4	The system shall generate a unique, time-limited (15 minutes) one-time verification code and send it to the user's registered email.
FR-4.5	The system shall require users to enter the verification code received via email.
FR-4.6	The system shall allow users to create a new password only after successful verification code validation.
FR-4.7	The system shall require password confirmation (re-typing the new password) to prevent typographical errors.

FR-4.8	The system shall enforce the same password strength requirements as during registration.
FR-4.9	The system shall prevent reusing the last 3 previously used passwords.
FR-4.10	The system shall send a confirmation email to the user's registered email address once the password has been successfully changed.

Table 26: FR-5 Resident Dashboard

Number	Description
FR-5.1	The system shall display a personalized dashboard for residents upon successful login.
FR-5.2	The dashboard shall show an overview of all tickets submitted by the resident, categorized by status: Open, In Progress, Fixed.
FR-5.3	The system shall display a summary widget showing: total tickets submitted, currently open tickets, and tickets fixed in the last 30 days.
FR-5.4	The system shall provide a quick action button to "Report New Issue" prominently displayed on the dashboard.
FR-5.5	The system shall show recent activity, including status changes to tickets, with timestamps.
FR-5.6	The system shall provide a search functionality to filter tickets by ticket ID, category, or date range.
FR-5.7	The system shall display notifications count in a badge on the dashboard header.
FR-5.8	The dashboard shall be responsive and adapt to different screen sizes (desktop, tablet, mobile).

Table 27: FR-6 Resident Navigation Menu

Number	Description
FR-6.1	The system shall provide a consistent navigation menu accessible from all resident pages.
FR-6.2	The navigation menu shall include the following items: Home (Dashboard), Report Status, Open Ticket, Notifications, Profile.
FR-6.3	The system shall highlight the currently active menu item for user orientation.
FR-6.4	The navigation menu shall collapse to a hamburger menu on mobile devices.
FR-6.5	The system shall display the user's name and role in the navigation header.
FR-6.6	The system shall include a logout button in the navigation menu.

Table 28: FR-7 Maintenance Issue Feed

Number	Description
FR-7.1	The system shall display a feed of reported maintenance issues accessible to residents.
FR-7.2	Each entry in the feed shall display: Ticket Title, Category, Location (Building/Room), Status (Open/In Progress/Fixed), and Date Reported.
FR-7.3	The system shall allow residents to filter the feed by: Category (Plumbing, Electrical, HVAC, Structural, Other), Status, and Date Range.

FR-7.4	The system shall provide a search functionality within the feed to find specific issues by keywords.
FR-7.5	The system shall indicate tickets submitted by the logged-in resident with a "Your Ticket" badge.
FR-7.6	The system shall prevent display of sensitive information or personal details of other residents.
FR-7.7	The system shall update the feed in real-time when new tickets are submitted or statuses change.
FR-7.8	The system shall paginate results if more than 20 items are displayed.

Table 29: FR-8 Open Maintenance Ticket

Number	Description
FR-8.1	The system shall provide a form for residents to submit new maintenance tickets.
FR-8.2	The form shall require the following mandatory fields: Issue Category (dropdown), Description (text area), Location (dropdown or text), Priority (Low/Medium/High/Urgent).
FR-8.3	The system shall allow residents to upload up to 5 images (JPEG, PNG, max 5MB each) to illustrate the issue.
FR-8.4	The system shall provide a preview of uploaded images before submission.
FR-8.5	The system shall validate that the description contains at least 20 characters to ensure adequate detail.
FR-8.6	The system shall check for duplicate tickets by comparing category, location, and description with recent (last 7 days) submissions.
FR-8.7	The system shall show a warning if a similar ticket exists, with an option to proceed or cancel.
FR-8.8	The system shall generate a unique ticket ID (format: TKT-YYYYMMDD-XXXXX) upon successful submission.
FR-8.9	The system shall display a confirmation page with the ticket ID and estimated response time based on priority.

Table 30: FR-9 Ticket Status Tracking

Number	Description
FR-9.1	The system shall allow residents to track the real-time status of their submitted tickets.
FR-9.2	The system shall display a visual status indicator showing the current stage: Submitted → Assigned → In Progress → Fixed → Closed.
FR-9.3	The system shall show the timestamp for each status change.
FR-9.4	For tickets "In Progress," the system shall display the assigned technician's name (if available).
FR-9.5	The system shall provide estimated time to completion based on priority and historical data.
FR-9.6	The system shall allow residents to add follow-up comments to their tickets, which will be visible to technicians and administrators.

FR-9.7	The system shall prevent residents from modifying the original ticket details after submission, except for adding comments.
FR-9.8	The system shall provide a “Mark as Urgent” option for tickets that have exceeded their estimated completion time.

Table 31: FR-10 Status Change Notifications

Number	Description
FR-10.1	The system shall automatically notify residents when their ticket status changes.
FR-10.2	Notifications shall be delivered through: In-app notification bell, Email (optional), SMS (optional, configurable).
FR-10.3	The notification shall include: Ticket ID, New Status, Timestamp, and a direct link to view the ticket details.
FR-10.4	The system shall allow residents to configure notification preferences for each type of status change.
FR-10.5	The system shall group related notifications (multiple status changes within a short period) to avoid notification overload.
FR-10.6	The system shall maintain a notification history accessible to residents for 90 days.

Table 32: FR-11 Duplicate Ticket Prevention

Number	Description
FR-11.1	The system shall show residents existing reports in the same location/category before they submit a new ticket.
FR-11.2	The system shall automatically detect potential duplicates by comparing: issue category, location, description keywords, and submission date (within last 7 days).
FR-11.3	When a potential duplicate is detected, the system shall display a warning message with links to the existing similar tickets.
FR-11.4	The system shall provide residents with the option to: proceed with new submission, add a comment to existing ticket, or cancel submission.
FR-11.5	The system shall allow residents to “follow” existing tickets to receive updates on their progress.
FR-11.6	The system shall track duplicate detection accuracy and allow administrators to adjust sensitivity thresholds.

Table 33: FR-12 Resident Profile Management

Number	Description
FR-12.1	The system shall allow residents to view their profile information including: full name, email, national ID, username, registration date.
FR-12.2	The system shall allow residents to edit the following fields: full name, email address, password, profile picture.
FR-12.3	The system shall prevent residents from editing their role, national ID, and registration date.

FR-12.4	The system shall require password verification before allowing sensitive changes (email, password).
FR-12.5	The system shall validate new email addresses by sending a verification link before applying the change.
FR-12.6	The system shall maintain an audit log of all profile changes with timestamp and IP address.
FR-12.7	The system shall allow residents to export their profile data in PDF format.

Table 34: FR-13 Technician Dashboard

Number	Description
FR-13.1	The system shall display a personalized dashboard for technicians upon successful login.
FR-13.2	The dashboard shall show assigned tasks categorized by: Pending (not started), In Progress, Completed Today, Overdue.
FR-13.3	The system shall display key performance metrics: total assigned tasks, completion rate (%), average resolution time, customer satisfaction rating.
FR-13.4	The system shall provide a calendar view showing scheduled maintenance tasks.
FR-13.5	The dashboard shall display urgent/high-priority tasks in a prominent “Priority Queue” section.
FR-13.6	The system shall show notifications for: new task assignments, task status updates from residents, scheduled maintenance reminders.
FR-13.7	The system shall provide quick action buttons: “Start Next Task”, “View All Tasks”, “Update Status”.
FR-13.8	The dashboard shall display workload distribution across technicians (visible to technicians with team lead permissions).

Table 35: FR-14 Technician Navigation Menu

Number	Description
FR-14.1	The system shall provide a consistent navigation menu accessible from all technician pages.
FR-14.2	The navigation menu shall include: Home (Dashboard), Assigned Tasks, Maintenance History, Notifications, Profile, Logout.
FR-14.3	For technicians with team lead permissions, the menu shall include additional items: Team Schedule, Performance Reports.
FR-14.4	The system shall display the technician’s current status (Available/Busy/On Break) in the navigation header.
FR-14.5	The system shall provide a quick status toggle button in the navigation menu to update availability.
FR-14.6	The navigation menu shall show badge counts for: pending tasks, unread notifications.
FR-14.7	The system shall collapse the navigation menu to an icon-only view on tablet devices.

Table 36: FR-15 Maintenance Request Management

Number	Description
FR-15.1	The system shall allow technicians to view maintenance requests assigned by the system or administrators.
FR-15.2	The system shall display request details including: ticket ID, issue category, location, priority, description, submitted images, resident contact information.
FR-15.3	Technicians shall be able to accept or decline assigned tasks, with a required reason for declining.
FR-15.4	The system shall automatically reassign declined tasks to the next available technician based on expertise and workload.
FR-15.5	Technicians shall be able to filter tasks by: priority, location, category, due date, or status.
FR-15.6	The system shall provide a "Claim Task" feature for technicians to voluntarily take unassigned tasks matching their expertise.
FR-15.7	The system shall display estimated time to complete based on similar historical tasks.
FR-15.8	Technicians shall be able to request additional information from the resident before accepting a task.

Table 37: FR-16 Task Status Updates

Number	Description
FR-16.1	The system shall allow technicians to update task status through the following workflow: Open → In Progress → Fixed → Closed.
FR-16.2	When changing status to "In Progress," the system shall record start time and expected completion time.
FR-16.3	When changing status to "Fixed," the system shall require: completion notes, actual resolution time, and optional images.
FR-16.4	Technicians shall be able to set a task to "On Hold" with reason codes: Waiting for Parts, Requires Specialist, Resident Not Available.
FR-16.5	The system shall automatically escalate tasks that remain "In Progress" beyond the estimated completion time.
FR-16.6	Technicians shall be able to reassign tasks to other technicians with proper justification and administrator approval.
FR-16.7	All status changes shall be timestamped and recorded in the task history.
FR-16.8	The system shall notify the resident and administrator of significant status changes.

Table 38: FR-17 Maintenance Evidence Submission

Number	Description
FR-17.1	The system shall allow technicians to upload images as evidence of maintenance completion.
FR-17.2	The system shall support multiple image formats: JPEG, PNG, with maximum file size of 10MB per image.
FR-17.3	Technicians shall be able to add descriptive captions to each uploaded image.

FR-17.4	The system shall allow technicians to add completion notes describing: work performed, parts used, time spent, any follow-up required.
FR-17.5	The system shall provide a checklist of standard completion criteria based on issue category.
FR-17.6	Technicians shall be able to attach PDF documents such as: warranty information, part specifications, safety checklists.
FR-17.7	The system shall require at least one piece of evidence (image or detailed notes) before marking a task as "Fixed."
FR-17.8	All submitted evidence shall be stored securely with timestamps and technician identification.

Table 39: FR-18 Technician Notifications

Number	Description
FR-18.1	The system shall notify technicians when a new task is assigned to them.
FR-18.2	The system shall notify technicians of updates to assigned tasks, including: resident comments, priority changes, due date adjustments.
FR-18.3	Notifications shall be delivered through: in-app notifications, push notifications (mobile app), SMS for urgent tasks.
FR-18.4	The system shall provide priority-based notification levels: High (immediate), Medium (within 15 minutes), Low (within 1 hour).
FR-18.5	Technicians shall be able to set "Do Not Disturb" periods during which only emergency notifications are delivered.
FR-18.6	The system shall group related notifications to reduce notification fatigue.
FR-18.7	Technicians shall be able to customize notification preferences by: notification type, delivery method, time of day.

Table 40: FR-19 Maintenance History Access

Number	Description
FR-19.1	The system shall provide technicians access to complete maintenance history for equipment and locations.
FR-19.2	The history shall include: ticket ID, issue description, location details, equipment ID, technician assigned, resolution details, date/time stamps.
FR-19.3	Technicians shall be able to filter history by: equipment ID, location, date range, technician, issue category.
FR-19.4	The system shall display recurring issues patterns and suggest preventive maintenance schedules.
FR-19.5	Technicians shall be able to view equipment-specific history including: all previous maintenance, warranty information, manufacturer details.
FR-19.6	The system shall provide a "Similar Issues" feature showing historical resolutions for current problems.
FR-19.7	Technicians shall be able to export maintenance history for specific equipment or locations in CSV format.

Table 41: FR-20 Basic Analytics for Technicians

Number	Description
FR-20.1	The system shall provide technicians with basic analytics on completed tasks.
FR-20.2	Analytics shall include: tasks completed per period (day/week/month), average resolution time, completion rate by category.
FR-20.3	The system shall display equipment performance trends showing frequency of issues by equipment type.
FR-20.4	Technicians shall be able to view area-specific statistics showing most frequently serviced locations.
FR-20.5	The system shall provide personal performance metrics compared to team averages.
FR-20.6	Analytics shall be visualized through: bar charts, line graphs, pie charts, and trend lines.
FR-20.7	Technicians shall be able to set personal performance goals and track progress.
FR-20.8	The system shall provide recommendations for skill development based on frequently assigned task categories.

Table 42: FR-21 Administrator Dashboard

Number	Description
FR-21.1	The system shall display a comprehensive dashboard for administrators upon successful login.
FR-21.2	The dashboard shall show key performance indicators (KPIs) including: total tickets (current month), open issues, resolved issues (last 7 days), average resolution time.
FR-21.3	The system shall display technician workload distribution showing: assigned tasks per technician, completion rates, current availability status.
FR-21.4	The dashboard shall include a real-time ticker showing: new tickets submitted, tickets resolved, overdue tasks.
FR-21.5	The system shall provide geographical heat map showing issue density by building/location.
FR-21.6	Administrators shall be able to customize dashboard widgets and rearrange layout according to preference.
FR-21.7	The dashboard shall display system health metrics: active users, system uptime, database performance.
FR-21.8	The system shall provide quick action buttons: “Assign Pending Tickets”, “Generate Reports”, “Manage Users”.

Table 43: FR-22 Administrator Navigation Menu

Number	Description
FR-22.1	The system shall provide a comprehensive navigation menu accessible from all administrator pages.
FR-22.2	The navigation menu shall include: Home (Dashboard), Ticket Management, Technician Management, Analytics and Reports, System Configuration, User Management, Profile, Logout.

FR-22.3	The system shall display administrator privileges and access level in the navigation header.
FR-22.4	The navigation menu shall include sub-menus for each main category with expanded options.
FR-22.5	The system shall highlight critical alerts in the navigation menu (e.g., “5 Urgent Tickets Pending”).
FR-22.6	Administrators shall be able to pin frequently used menu items to a quick access bar.
FR-22.7	The system shall provide keyboard shortcuts for common navigation actions.

Table 44: FR-23 Ticket Assignment Management

Number	Description
FR-23.1	The system shall allow administrators to assign maintenance tickets to technicians manually or automatically.
FR-23.2	For manual assignment, the system shall show: technician availability, current workload, expertise match, location proximity.
FR-23.3	The system shall support automatic assignment based on: priority level, technician specialization, workload balancing, geographical zones.
FR-23.4	Administrators shall be able to override automatic assignments with manual reassignments.
FR-23.5	The system shall provide bulk assignment functionality for multiple tickets at once.
FR-23.6	When assigning tickets, administrators shall be able to set: expected completion time, special instructions, required tools/parts.
FR-23.7	The system shall maintain assignment history showing all assignment changes with timestamps and reasoning.
FR-23.8	Administrators shall be able to set up assignment rules and automation policies for recurring ticket types.

Table 45: FR-24 Ticket Priority Management

Number	Description
FR-24.1	The system shall allow administrators to set and modify ticket priority levels: Low, Medium, High, Urgent, Emergency.
FR-24.2	Priority levels shall determine: response time expectations, assignment order, escalation rules.
FR-24.3	Administrators shall be able to bulk update priorities for multiple tickets based on criteria.
FR-24.4	The system shall automatically adjust priorities based on: time since submission, number of similar issues, affected users count.
FR-24.5	Administrators shall be able to define custom priority rules based on: location criticality, time of day, equipment importance.
FR-24.6	Priority changes shall trigger notifications to assigned technicians and residents.
FR-24.7	The system shall maintain audit trail of all priority changes with justification notes.

Table 46: FR-25 Administrator Notifications

Number	Description
FR-25.1	The system shall notify administrators of new high-priority ticket submissions.
FR-25.2	Administrators shall receive notifications for: overdue tasks (exceeding expected completion time), unassigned tickets exceeding threshold time.
FR-25.3	The system shall notify administrators when maintenance work is completed, requiring quality assurance review.
FR-25.4	Notifications shall be categorized by: urgency (Immediate/High/Medium/Low), department, location.
FR-25.5	Administrators shall be able to configure notification thresholds and escalation paths.
FR-25.6	The system shall provide notification summary reports showing: notification volume, response times, unresolved alerts.
FR-25.7	Administrators shall be able to snooze notifications or set “Out of Office” auto-replies.

Table 47: FR-26 Analytics and Reporting

Number	Description
FR-26.1	The system shall provide comprehensive analytics dashboards showing maintenance trends over time.
FR-26.2	Analytics shall include: frequently reported areas/equipment, recurring issue patterns, seasonal trends.
FR-26.3	The system shall display equipment performance analytics: MTBF (Mean Time Between Failures), maintenance costs, downtime analysis.
FR-26.4	Administrators shall be able to analyze task completion times by: technician, category, location, priority.
FR-26.5	The system shall provide predictive analytics suggesting preventive maintenance schedules.
FR-26.6	Analytics shall include cost analysis: labor costs, parts costs, total maintenance expenditure by period.
FR-26.7	Administrators shall be able to create custom reports using drag-and-drop report builder.
FR-26.8	The system shall support comparative analysis: month-over-month, year-over-year, location comparisons.

Table 48: FR-27 Report Export Functionality

Number	Description
FR-27.1	The system shall allow administrators to export maintenance reports in multiple formats: PDF, Excel (XLSX), CSV.
FR-27.2	PDF exports shall include: company logo, report title, date range, pagination, professional formatting.
FR-27.3	Excel exports shall preserve: formulas, charts, filters, and data validation where applicable.

FR-27.4	Administrators shall be able to schedule automated report generation and email distribution.
FR-27.5	The system shall provide pre-built report templates: monthly maintenance summary, technician performance, equipment history.
FR-27.6	Exported reports shall include all relevant metadata: generation timestamp, exported by, report parameters.
FR-27.7	Administrators shall be able to export raw data for external analysis in statistical software.
FR-27.8	The system shall maintain export history with download logs for audit purposes.

Table 49: FR-28 User Account Management

Number	Description
FR-28.1	The system shall allow administrators to manage all registered user accounts.
FR-28.2	Administrators shall be able to: activate, deactivate, suspend, or delete user accounts.
FR-28.3	For technician accounts, administrators shall be able to: assign specializations, set work zones, define skill levels.
FR-28.4	Administrators shall be able to update user information: contact details, role changes (with approval workflow), access permissions.
FR-28.5	The system shall provide bulk user management operations: import users from CSV, bulk role assignment, mass communication.
FR-28.6	Administrators shall be able to reset user passwords and force password change on next login.
FR-28.7	The system shall maintain comprehensive audit logs of all user management activities.
FR-28.8	Administrators shall be able to set account expiration dates and receive renewal reminders.

3.6. Data Requirements

Data requirements is the specification for the information the system will depend on, store in the database, and process to achieve the business goals. These requirements are important for effective system design and implementation. For our smart maintenance request system, the core data entities, including their attributes, constraints, and the key relationships between them, are:

Table 50: Data Requirements

Entity	**Description**	**Attributes**	**Constraints**	**Key relationships**
Users	The Users entity represents any system user, including technicians, staff, admins, and requestors.	UserID (PK), full name, password hash, NationalID to initiate password recovery, IsEmailVerified, role (which can	UserID is the Primary Key, Email and NationalID have a uniqueness constraint, role can be User, Technician,	• One user can submit many requests (one to many). • One user receives multiple

		be User, Technician, Admin, or other staff), registration date, account status (Active, Suspended, Locked), LastLoginAt for security, account expiration date, failed login attempts, and profile picture. Email address will also be stored for communication, and a Phone number (is an optional field).	Admin, or other staff.	notifications (one to many). • A user can be linked to zero or one technician record depending on their role (one to one or one to zero).
Technician	The Technician entity represents technicians or teams responsible for carrying out maintenance tasks.	This entity tracks their skills, an availability status to indicate whether they are available, busy, or on break. Other fields are for skill level, certifications, specialization, work zone, work schedules, TeamLeadFlag to identify technicians with team lead permissions, average resolution time (analytics), customer satisfaction rating. It also	TechnicianID is the Primary Key, UserID is a Foreign Key referring to the User table, specialization is required.	• One technician may handle many requests. • One technician can have many scheduled tasks.

		tracks their performance, such as the number of tasks they have completed. Each technician has a unique TechnicianID (PK) and a UserID, which is a FK referring to the User table.		
Maintenance Request	The Maintenance Request entity documents the actual problem submitted by users or a need for service.	A request captures details such as the issue description, TicketID, title, category, the requester's ID (a FK referring to the User table), TechnicianID (FK), LocationID (FK), ReportedAt, AssignedAt, and ResolvedAt. It also includes priority (Low, Medium, High, Critical), current status (new, in progress, completed, rejected), estimated completion time, IsUrgentFlag, SimilarityFlag to prevent duplicate tickets, escalation level for escalate tasks that remain "In Progress" beyond the estimated	TicketId is PK, requester's ID, TechnicianID, and LocationID are required Foreign keys, description minimum length is 20 chars, ReportedAt is required, AssignedAt is optional, ResolvedAt is optional, ImageURL is optional	• Each maintenance request is assigned to one location. • One location can have many requests.

		completion time, OnHoldReason to allow technicians to set a task to “On Hold” for specific reasons, ResidentEditableFlag (to enforce FR-9.7), and an ImageURL that is optional.		
Location	The Location entity represents the building or place where maintenance is required, facilitating easy navigation and assignment of work to the correct place.	Each location has a LocationID (PK). It also includes fields for zone, building name, location criticality, GeoCoordinates for heat map, and a column for other details if there is a specific description of the components needed for repairs. Additional useful fields include room number and floor number and they are optional.	LocationID is the Primary Key, BuildingName is required	Each request is associated with exactly one location (one to many).
Maintenance Schedule	The Maintenance Schedule entity includes a list of planned maintenance tasks.	Each schedule has a ScheduleID (PK), TechnicianID (FK referring to Technician), LinkedRequestID to indicate which maintenance request the schedule is	ScheduleID is the Primary Key, TechnicianID is a Foreign Key, scheduled start time is required.	One technician can have many scheduled tasks.

		created for, expected duration, scheduled start time, recurrence pattern (None (default), Daily, Weekly, Monthly, Yearly), schedule status (Scheduled, Completed, Cancelled) and notes (String, optional).		
Notifications	The Notifications entity tracks alerts sent to users.	NotificationID (PK), UserID (FK), RelatedTiketID (FK to Maintenance Request table), expiration date (90 days as it is set in the rules), notification type (StatusChange, Assignment, Alert…), message (String), IsRead (Boolean), CreatedAt (DateTime).	NotificationID is the Primary Key, UserID is a Foreign Key, Message is required, CreatedAt is required.	One user receives many notifications.

And to allow the system to automate workflows, these entities we listed are interconnected in a centralized database, interacting with each other to streamline the maintenance lifecycle from issue reporting to resolution and performance optimization.

3.7. Non-Functional Requirements

The non-functional requirements for the Maintenance Management System are outlined in the table below. Each requirement is assigned a unique number for reference.

Table 51: Non-Functional Requirements

Number	Non-Functional Requirements	Description
1	Performance	Users must be able to submit and track maintenance requests without delays or timeouts, even during peak usage times.
2	Dependability	The System must operate 24/7 with minimal downtime.
3	Security	The system must prevent unauthorized access and encrypt data such as login credentials and ticket details.
4	Usability	The system must have a simple , user-friendly interface that is easy to understand for all users.
5	Operational and Environmental Constraints	The system must run in web browsers and mobile devices and require a stable internet connection ; data will be stored using a database management system.
6	Maintainability and Supportability	The system must be easy to maintain , and it should deal with errors and solve them when they appear in the system.

3.8. Requirements Validation and Review Summary

3.8.1. How We Verified the Requirements

We verified the requirements by reviewing them with both our team and the stakeholders.

3.8.1.1. Team Review

All team members **(Anas, Shaima, Orjoan, Musa, and Hanen)** reviewed the requirements together. We went through each requirement to ensure everyone fully understood it. Each member provided feedback and suggestions, which were used to refine the requirements.

3.8.1.2. Stakeholder Review

We presented the requirements to **Dr. Hamad** for validation. We also gathered feedback from three classmates acting as potential users. We updated and improved the requirements based on their comments and suggestions.

3.8.2. How We Confirmed the Requirements Are Correct, Clear, and Complete

We used two main methods to validate the clarity and correctness of the requirements: **Mockups** and **Walkthroughs**.

3.8.2.1. Method 1: Mockups

What we did: We created UI mockups using Figma.

What we showed:

- User dashboard showing submitted maintenance requests
- Technician dashboard with assigned tasks organized by priority
- Admin panel for monitoring and managing maintenance requests

- Notification screen showing status updates

How this helped:
- Clarified how users will report maintenance issues
- Confirmed that role-based dashboards function correctly
- Verified that priority assignment and task flow work as intended

3.8.2.2. Method 2: Walkthroughs

What we did: We conducted a full system walkthrough during a meeting.

How we did it: We explained each feature step-by-step and demonstrated how it works for the different user roles. We walked through realistic scenarios such as:
- "A student reports a broken AC."
- "A technician receives the notification and accepts the task."
- "An admin monitors all requests and reassigns urgent tasks."

How this helped:
- Ensured everyone clearly understood the maintenance workflow
- Confirmed that email and system notifications meet requirements
- Verified that priority-based task assignment is correct
- Ensured that requirements for maintenance history and analytics are complete

4. System Analysis and Design

4.1. DFDs

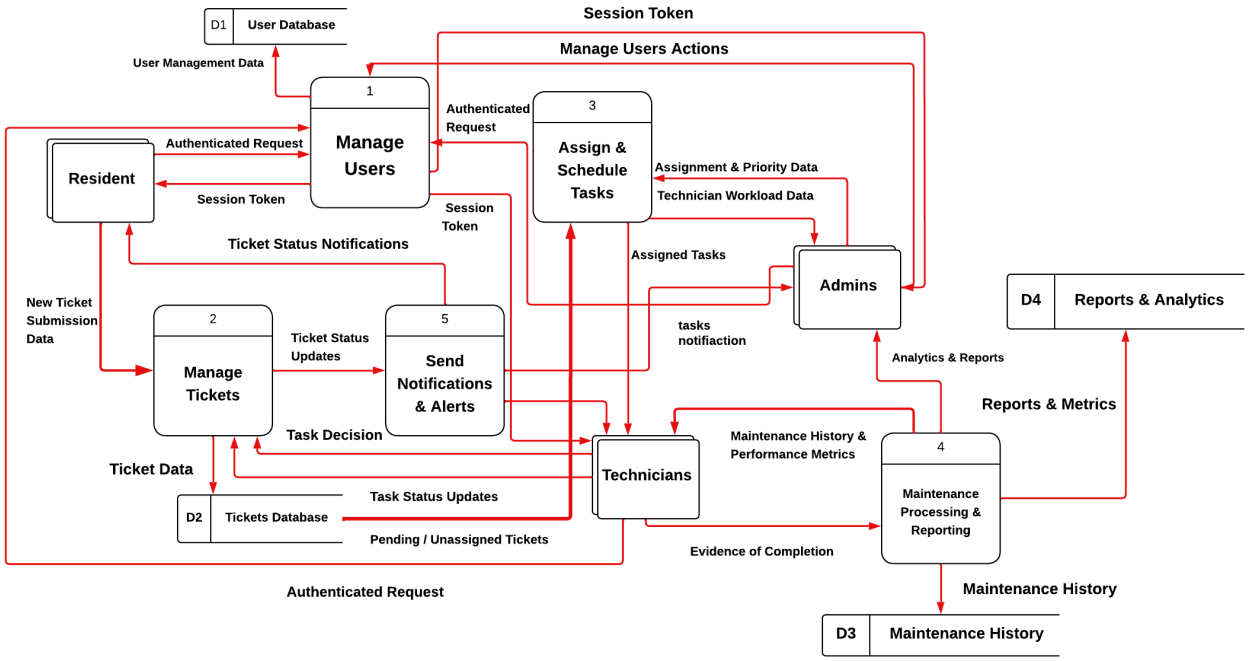


Figure 39: DFD Level 0

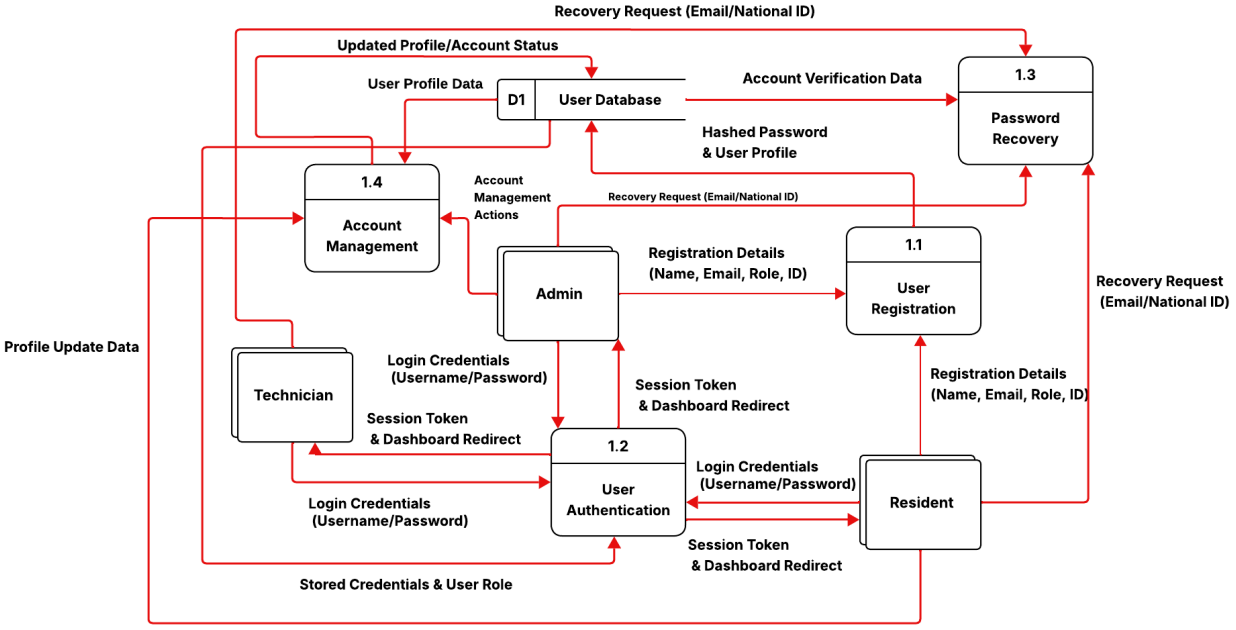


Figure 40: DFD Level 1-Fragment 1

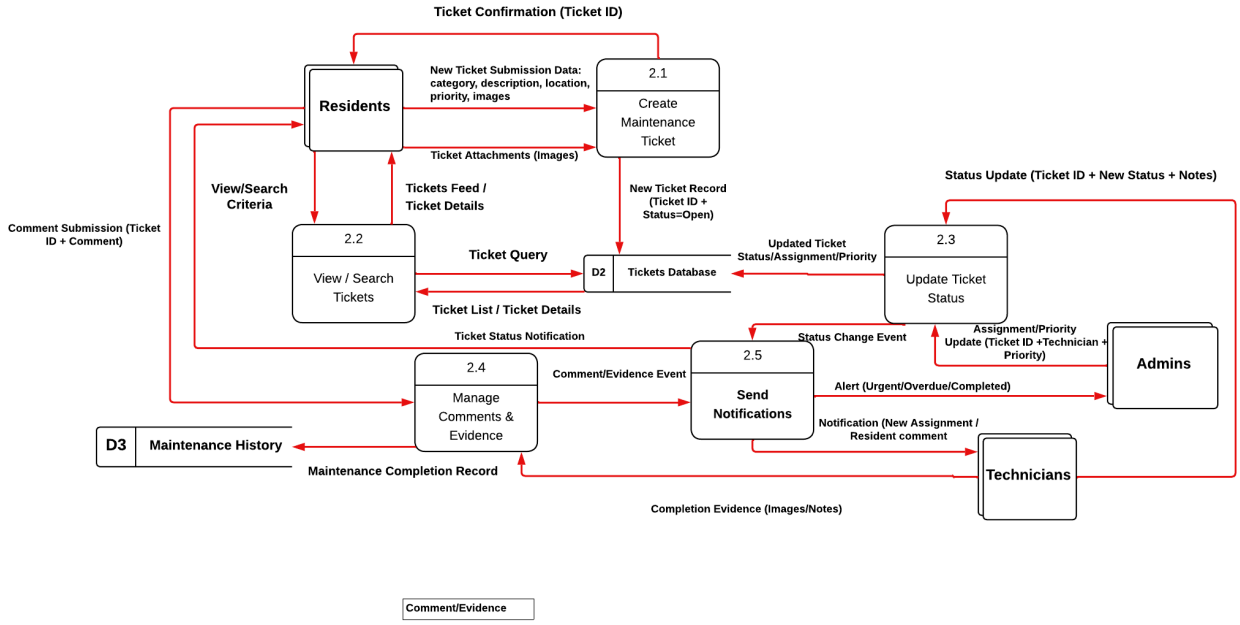


Figure 41: DFD Level 1-Fragment 2

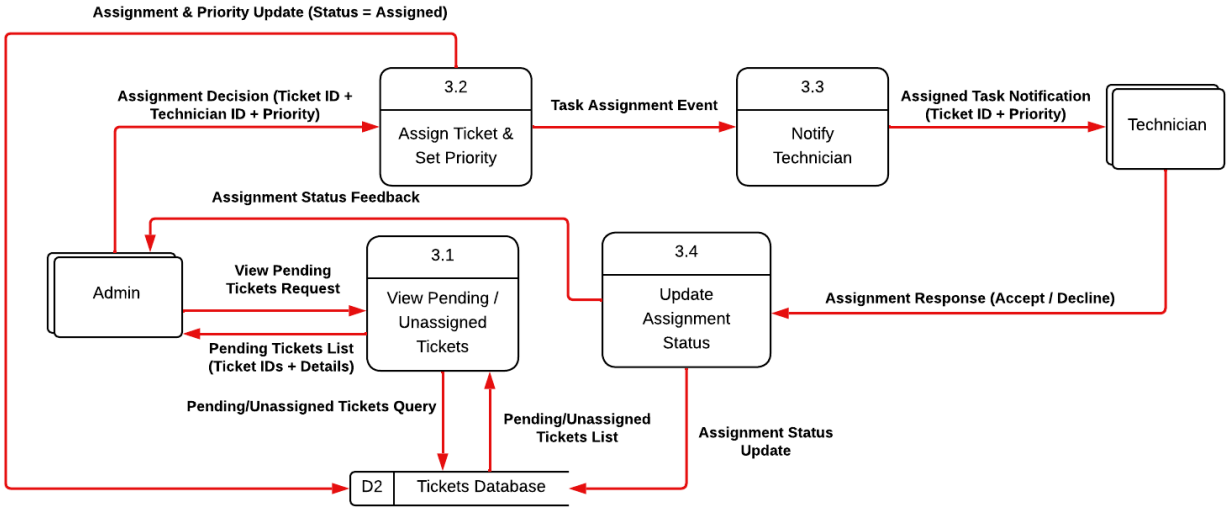


Figure 42: DFD Level 1-Fragment 3

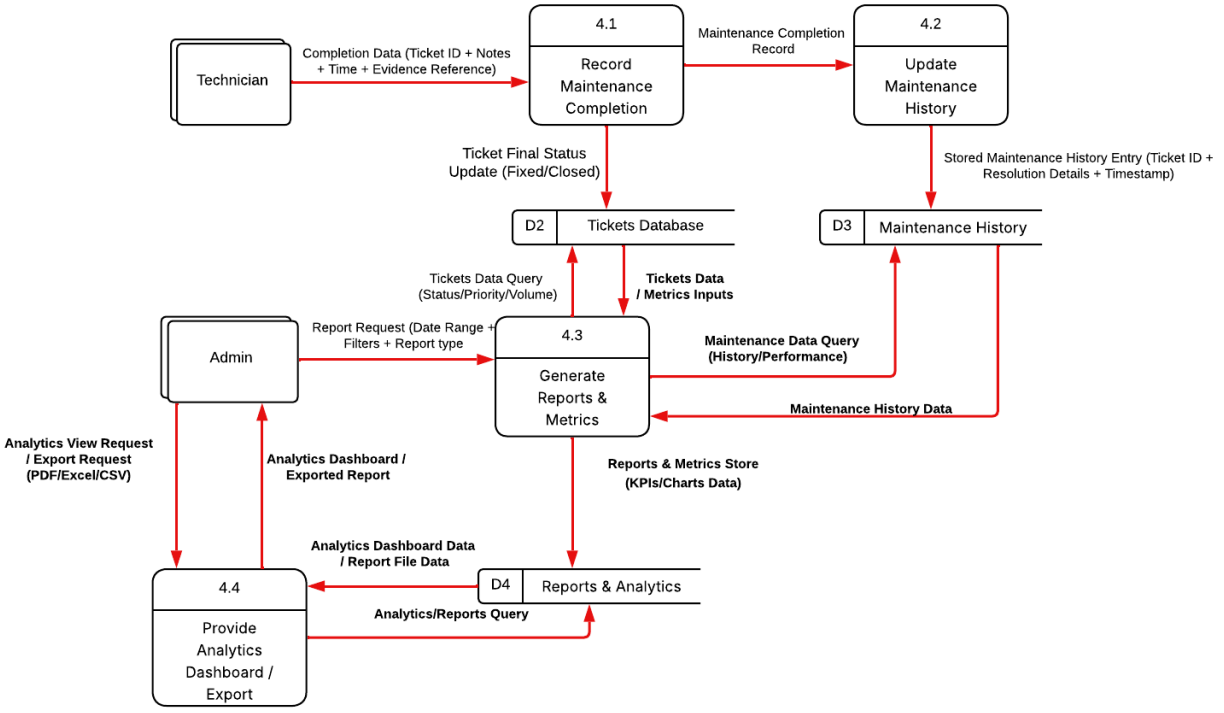


Figure 43: DFD Level 1-Fragment 4

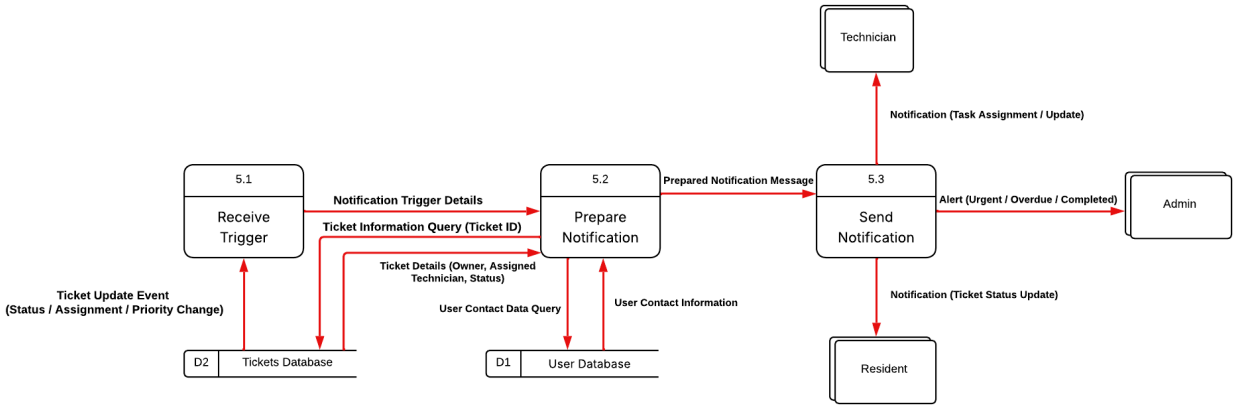


Figure 44: DFD Level 1-Fragment 5

4.2. Data Dictionaries

4.2.1. Data flow

Table 52: Data Flow 1 Details

Field	Details
Name	Registration Details
Description	User information submitted during account registration.
Source	External Entities: Resident / Admin
Destination	Process 1.1: User Registration
Type	Screen
Data Structure	Name, Email, Role, National ID
Volume/Time	Per registration
Comments	Used to create a new user account.

Table 53: Data Flow 2 Details

Field	Details
Name	User Profile Data
Description	User profile information stored or retrieved from database.
Source	User Database (D1)
Destination	Processes 1.4
Type	Database Read
Data Structure	User ID, Name, Email, Role, Account Status
Volume/Time	On demand
Comments	Represents stored user account information.

Table 54: Data Flow 3 Details

Field	Details
Name	Login Credentials
Description	Credentials submitted by user for authentication.

Source	External Entities: Resident / Technician / Admin
Destination	Process 1.2: User Authentication
Type	Screen
Data Structure	Username, Password
Volume/Time	Per login attempt
Comments	Validated against stored credentials.

Table 55: Data Flow 4 Details

Field	Details
Name	Stored Credentials & User Role
Description	Stored authentication data retrieved for validation.
Source	User Database (D1)
Destination	Process 1.2: User Authentication
Type	Database Read
Data Structure	Username, Hashed Password, Role, Account Status
Volume/Time	Per login attempt
Comments	Used to verify user identity and access level.

Table 56: Data Flow 5 Details

Field	Details
Name	Session Token & Dashboard Redirect
Description	Authentication response indicating successful login.
Source	Process 1.2: User Authentication
Destination	External Entities: Resident / Technician / Admin
Type	System Response
Data Structure	Session Token, User Role, Redirect Path
Volume/Time	Per successful login
Comments	Grants access to authorized system functions.

Table 57: Data Flow 6 Details

Field	Details
Name	Recovery Request
Description	User request to recover account access.
Source	External Entity: Resident,Admin,Technician
Destination	Process 1.3: Password Recovery
Type	Screen
Data Structure	Email or National ID
Volume/Time	On demand
Comments	Used to initiate password recovery.

Table 58: Data Flow 7 Details

Field	Details
Name	Account Verification Data
Description	User data used to verify identity during recovery.
Source	User Database (D1)
Destination	Process 1.3: Password Recovery
Type	Database Read
Data Structure	User ID, Email, National ID, Account Status
Volume/Time	Per recovery request
Comments	Ensures recovery request is valid.

Table 59: Data Flow 8 Details

Field	Details
Name	Updated Profile / Account Status
Description	Updated user profile or account status information.
Source	Process 1.4: Account Management
Destination	User Database (D1)
Type	Database Write
Data Structure	User ID, Updated Fields, Account Status
Volume/Time	Per update
Comments	Reflects administrative or user profile changes.

Table 60: Data Flow 9 Details

Field	Details
Name	Profile Update Data
Description	Profile changes submitted by resident for account update.
Source	External Entity: Resident
Destination	Process 1.4: Account Management
Type	Screen
Data Structure	User ID, Updated Profile Fields
Volume/Time	On demand
Comments	Used to update resident profile information.

Table 61: Data Flow 10 Details

Field	Details
Name	New Ticket Submission Data
Description	Resident submits information to create a new maintenance ticket.
Source	External Entity: Residents
Destination	Process 2.1: Create Maintenance Ticket
Type	Screen / Form
Data Structure	Category, Description, Location, Priority
Volume/Time	On demand (per new ticket)
Comments	Core input required to open a new ticket.

Table 62: Data Flow 11 Details

Field	Details
Name	Ticket Attachments (Images)
Description	Images uploaded by resident as supporting evidence for the ticket.
Source	External Entity: Residents
Destination	Process 2.1: Create Maintenance Ticket
Type	File Upload
Data Structure	Image File(s), File Type, File Size, (Optional) Attachment Reference
Volume/Time	0..N images per ticket
Comments	Stored as files or references/URLs depending on implementation.

Table 63: Data Flow 12 Details

Field	Details
Name	Ticket Confirmation (Ticket ID)
Description	Confirmation sent to resident after successful ticket creation.
Source	Process 2.1: Create Maintenance Ticket
Destination	External Entity: Residents
Type	Screen / Message
Data Structure	Ticket ID, Status (=Open), Created Date/Time
Volume/Time	Per ticket submission
Comments	Used by resident to track the ticket.

Table 64: Data Flow 13 Details

Field	Details
Name	New Ticket Record
Description	New ticket record written into the tickets database.
Source	Process 2.1: Create Maintenance Ticket
Destination	Tickets Database (D2)
Type	Database Write
Data Structure	Ticket ID, Resident ID, Category, Description, Location, Priority, Status=Open, Created Date, Attachment References
Volume/Time	Per new ticket
Comments	Initializes the ticket life-cycle in D2.

Table 65: Data Flow 14 Details

Field	Details
Name	View/Search Criteria
Description	Criteria entered by resident to search or filter tickets (query merged here).
Source	External Entity: Residents
Destination	Process 2.2: View / Search Tickets
Type	Screen
Data Structure	Ticket ID (optional), Status, Category, Date Range, Keyword
Volume/Time	On demand
Comments	Used to define ticket search conditions.

Table 66: Data Flow 15 Details

Field	Details
Name	Ticket List / Ticket Details
Description	Matching tickets returned from D2 for list view or detailed view.
Source	Tickets Database (D2)
Destination	Process 2.2: View / Search Tickets
Type	Database Read Result
Data Structure	Ticket ID, Category/Title, Description, Location, Priority, Status, Created Date, Assigned Technician (optional), Last Updated, Notes (optional)
Volume/Time	Depends on number of matches
Comments	Retrieved ticket data for display purposes

Table 67: Data Flow 16 Details

Field	Details
Name	Comment Submission (Ticket ID + Comment)
Description	Resident submits a comment related to an existing ticket.
Source	External Entity: Residents
Destination	Process 2.4: Manage Comments & Evidence
Type	Screen
Data Structure	Ticket ID, Resident ID, Comment Text, Comment Date/Time
Volume/Time	0..N per ticket
Comments	Stored and used for communication/updates.

Table 68: Data Flow 17 Details

Field	Details
Name	Completion Evidence (Images/Notes)
Description	Technician uploads evidence/notes for progress or completion.
Source	External Entity: Technicians
Destination	Process 2.4: Manage Comments & Evidence
Type	Screen / File Upload
Data Structure	Ticket ID, Technician ID, Notes, Evidence Image(s)/References, Evidence Date/Time
Volume/Time	Per update/completion
Comments	Supports verification and maintenance history logging.

Table 69: Data Flow 18 Details

Field	Details
Name	Maintenance Completion Record
Description	Completion details stored in maintenance history.
Source	Process 2.4: Manage Comments & Evidence
Destination	Maintenance History (D3)
Type	Database Write
Data Structure	Ticket ID, Technician ID, Work Summary/Notes, Completion Date/Time, Evidence References, Final Status
Volume/Time	Typically once per completed ticket
Comments	Archived record used for reporting and future reference.

Table 70: Data Flow 19 Details

Field	Details
Name	Assignment/Priority Update
Description	Admin assigns a technician and sets/updates ticket priority.
Source	External Entity: Admins
Destination	Process 2.3: Update Ticket Status
Type	Screen
Data Structure	Ticket ID, Technician ID, Priority, (Optional) Notes
Volume/Time	Per assignment/change
Comments	Changes are written to D2 and may trigger notifications.

Table 71: Data Flow 20 Details

Field	Details
Name	Status Update (Ticket ID + New Status + Notes)
Description	Technician updates ticket status with optional notes.
Source	External Entity: Technicians
Destination	Process 2.3: Update Ticket Status
Type	Screen
Data Structure	Ticket ID, New Status, Notes (optional), Updated Date/Time
Volume/Time	Per status change
Comments	Example statuses: In Progress, Completed, On Hold.

Table 72: Data Flow 21 Details

Field	Details
Name	Updated Ticket Status/Assignment/Priority
Description	Updated ticket information written back to the tickets database.
Source	Process 2.3: Update Ticket Status
Destination	Tickets Database (D2)
Type	Database Write
Data Structure	Ticket ID, Status, Technician ID (optional), Priority (optional), Notes (optional), Last Updated
Volume/Time	Per update
Comments	Ensures D2 reflects the latest ticket state.

Table 73: Data Flow 22 Details

Field	Details
Name	Notification
Description	System-generated notification sent to users based on ticket updates or comments.
Source	Process 2.5: Send Notifications
Destination	External Entities: Residents / Technicians / Admins
Type	Notification
Data Structure	Ticket ID, Notification Type, Status/Priority (as applicable), Message Summary, Timestamp
Volume/Time	As triggered
Comments	Unified structure for all system notifications.

Table 74: Data Flow 23 Details

Field	Details
Name	View Pending / Tickets Request
Description	Request sent by Admin to view all pending or unassigned tickets.
Source	External Entity: Admin
Destination	Process 3.1: View Pending / Unassigned Tickets
Type	Screen
Data Structure	Request Parameters
Volume/Time	On demand
Comments	Used to retrieve tickets that are not yet assigned.

Table 75: Data Flow 24 Details

Field	Details
Name	Pending Tickets List
Description	List of pending or unassigned tickets returned from database.
Source	Tickets Database (D2)
Destination	Process 3.1: View Pending / Unassigned Tickets
Type	Database
Data Structure	Ticket ID, Title, Description, Status, Created Date
Volume/Time	Depends on number of tickets
Comments	Displayed to admin for assignment decision.

Table 76: Data Flow 25 Details

Field	Details
Name	Assignment Decision
Description	Admin selects technician and sets ticket priority.
Source	External Entity: Admin
Destination	Process 3.2: Assign Ticket & Set Priority
Type	Screen
Data Structure	Ticket ID, Technician ID, Priority
Volume/Time	Per assignment
Comments	Ticket status becomes Assigned.

Table 77: Data Flow 26 Details

Field	Details
Name	Assignment & Priority Update
Description	Updates ticket assignment and priority in database.
Source	Process 3.2: Assign Ticket & Set Priority
Destination	Tickets Database (D2)
Type	Database
Data Structure	Ticket ID, Technician ID, Priority, Status
Volume/Time	Per assignment
Comments	Status is set to Assigned.

Table 78: Data Flow 27 Details

Field	Details
Name	Assigned Task Notification
Description	Notification sent to technician for assigned ticket.
Source	Process 3.3: Notify Technician
Destination	External Entity: Technician
Type	Notification
Data Structure	Ticket ID, Priority
Volume/Time	Per assignment
Comments	Notification via system, email, or sms.

Table 79: Data Flow 28 Details

Field	Details
Name	Assignment Response
Description	Technician accepts or declines assigned task.
Source	External Entity: Technician
Destination	Process 3.4: Update Assignment Status
Type	Screen
Data Structure	Ticket ID, Response Status
Volume/Time	Per notification
Comments	Determines next assignment status.

Table 80: Data Flow 29 Details

Field	Details
Name	Assignment Status Update
Description	Updates ticket status based on technician response.
Source	Process 3.4: Update Assignment Status
Destination	Tickets Database (D2)
Type	Database
Data Structure	Ticket ID, Status
Volume/Time	Per response
Comments	Status may be Accepted or Declined.

Table 81: Data Flow 30 Details

Field	Details
Name	Completion Data
Description	Maintenance completion information submitted by technician.
Source	External Entity: Technician
Destination	Process 4.1: Record Maintenance Completion
Type	Screen
Data Structure	Ticket ID, Completion Notes, Completion Time, Evidence Reference
Volume/Time	Per completed ticket
Comments	Used to record task completion.

Table 82: Data Flow 31 Details

Field	Details
Name	Maintenance Completion Record
Description	Completed maintenance record sent for history update.
Source	Process 4.1: Record Maintenance Completion
Destination	Process 4.2: Update Maintenance History
Type	Internal Data
Data Structure	Ticket ID, Resolution Details, Completion Time, Evidence Reference
Volume/Time	Per completion
Comments	Passed for archival storage.

Table 83: Data Flow 32 Details

Field	Details
Name	Ticket Final Status Update
Description	Final ticket status written after maintenance completion.
Source	Process 4.1: Record Maintenance Completion
Destination	Tickets Database (D2)
Type	Database Write
Data Structure	Ticket ID, Final Status (Fixed/Closed), Last Updated
Volume/Time	Per completed ticket
Comments	Closes the ticket life-cycle.

Table 84: Data Flow 33 Details

Field	Details
Name	Stored Maintenance History Entry
Description	Archived maintenance history entry.
Source	Process 4.2: Update Maintenance History
Destination	Maintenance History (D3)
Type	Database Write
Data Structure	Ticket ID, Resolution Details, Completion Timestamp
Volume/Time	Per completion
Comments	Used for reporting and audits.

Table 85: Data Flow 34 Details

Field	Details
Name	Report Request
Description	Request parameters for generating reports or analytics.
Source	External Entity: Admin
Destination	Process 4.3: Generate Reports & Metrics
Type	Screen
Data Structure	Date Range, Filters, Report Type
Volume/Time	On demand
Comments	Defines report scope.

Table 86: Data Flow 35 Details

Field	Details
Name	Tickets Data / Metrics Inputs
Description	Ticket data used for analytics and KPI calculations.
Source	Tickets Database (D2)
Destination	Process 4.3: Generate Reports & Metrics
Type	Database Read
Data Structure	Ticket ID, Status, Priority, Timestamps
Volume/Time	As requested
Comments	Provides operational metrics.

Table 87: Data Flow 36 Details

Field	Details
Name	Maintenance History Data
Description	Historical maintenance data retrieved for analysis.
Source	Maintenance History (D3)
Destination	Process 4.3: Generate Reports & Metrics
Type	Database Read
Data Structure	Ticket ID, Resolution Details, Completion Time
Volume/Time	As requested
Comments	Supports performance analysis.

Table 88: Data Flow 37 Details

Field	Details
Name	Reports & Metrics Store
Description	Generated reports and analytics data stored for access.
Source	Process 4.3: Generate Reports & Metrics
Destination	Reports & Analytics (D4)
Type	Database Write
Data Structure	Report ID, KPI Data, Charts Data, Generated Date
Volume/Time	Per report
Comments	Enables reuse and export.

Table 89: Data Flow 38 Details

Field	Details
Name	Analytics View / Export Request
Description	Admin request to view or export analytics reports.
Source	External Entity: Admin
Destination	Process 4.4: Provide Analytics Dashboard / Export
Type	Screen
Data Structure	Report ID, Export Format (PDF/Excel/CSV)
Volume/Time	On demand
Comments	Triggers dashboard or export.

Table 90: Data Flow 39 Details

Field	Details
Name	Analytics Dashboard Data / Report File Data
Description	Analytics data or exported report returned to admin.
Source	Reports & Analytics (D4)
Destination	Process 4.4: Provide Analytics Dashboard / Export
Type	Database Read
Data Structure	KPI Values, Charts Data, Report File
Volume/Time	Per request
Comments	Prepared for presentation or download.

Table 91: Data Flow 40 Details

Field	Details
Name	Analytics Dashboard / Exported Report
Description	Final analytics dashboard or exported report.
Source	Process 4.4: Provide Analytics Dashboard / Export
Destination	External Entity: Admin
Type	Screen / File
Data Structure	Dashboard View or Report File
Volume/Time	On demand
Comments	Supports monitoring and decision-making.

Table 92: Data Flow 41 Details

Field	Details
Name	Notification Trigger Details
Description	Trigger details passed for notification preparation.
Source	Process 5.1: Receive Trigger
Destination	Process 5.2: Prepare Notification
Type	Internal Data
Data Structure	Ticket ID, Trigger Type (Status/Assignment/Priority)
Volume/Time	Per trigger
Comments	Defines what notification is needed.

Table 93: Data Flow 42 Details

Field	Details
Name	Ticket Information Query
Description	Request to retrieve ticket details using the ticket ID.
Source	Process 5.2: Prepare Notification
Destination	Tickets Database (D2)
Type	Database Read
Data Structure	Ticket ID
Volume/Time	Per trigger
Comments	Fetches ticket context for the message.

Table 94: Data Flow 43 Details

Field	Details
Name	Ticket Details
Description	Ticket information returned for notification preparation.
Source	Tickets Database (D2)
Destination	Process 5.2: Prepare Notification
Type	Database Read Result
Data Structure	Ticket ID, Owner, Assigned Technician, Status, Priority
Volume/Time	Per query
Comments	Used to personalize notification content.

Table 95: Data Flow 44 Details

Field	Details
Name	User Contact Data Query
Description	Request to retrieve contact info for notification delivery.
Source	Process 5.2: Prepare Notification
Destination	User Database (D1)
Type	Database Read
Data Structure	User ID(s) (Owner/Technician/Admin)
Volume/Time	Per trigger
Comments	Retrieves recipient contact details.

Table 96: Data Flow 45 Details

Field	Details
Name	User Contact Information
Description	Recipient contact details returned from user database.
Source	User Database (D1)
Destination	Process 5.2: Prepare Notification
Type	Database Read Result
Data Structure	User ID, Name, Email, Phone (optional), Role
Volume/Time	Per query
Comments	Used to route notifications correctly.

Table 97: Data Flow 46 Details

Field	Details
Name	Prepared Notification Message
Description	Final notification message prepared for sending.
Source	Process 5.2: Prepare Notification
Destination	Process 5.3: Send Notification
Type	Internal Data
Data Structure	Ticket ID, Recipient Role, Notification Type, Message Summary, Timestamp
Volume/Time	Per notification
Comments	Ready for delivery to recipients.

Table 98: Data Flow 47 Details

Field	Details
Name	Notification
Description	Notification delivered to system users.
Source	Process 5.3: Send Notification
Destination	External Entities: Technician / Admin / Resident
Type	Notification
Data Structure	Ticket ID, Notification Type, Message Summary, Timestamp
Volume/Time	As triggered
Comments	Unified structure for all recipients.

4.3. Object-Oriented Analysis

4.3.1. Activity Diagrams

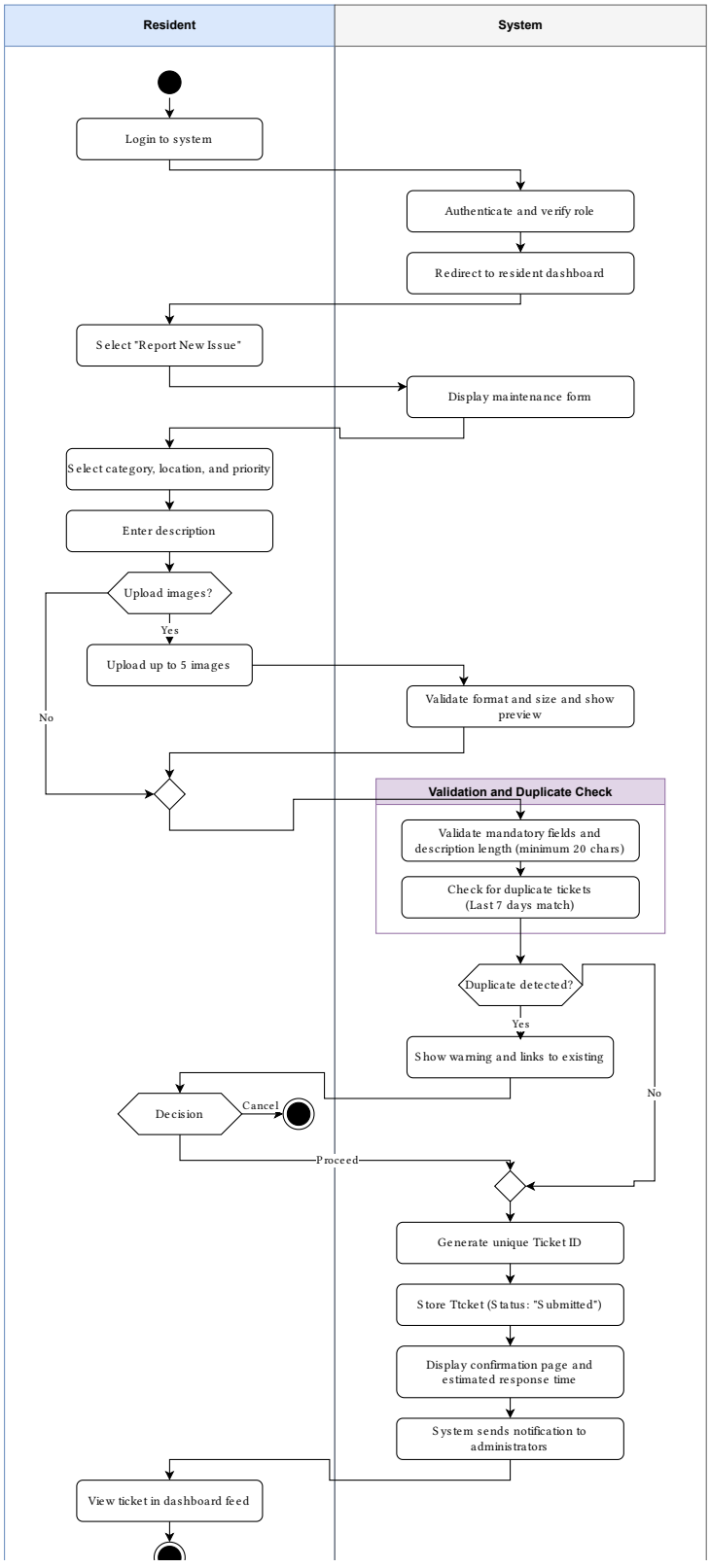


Figure 45: Resident Activity Diagram

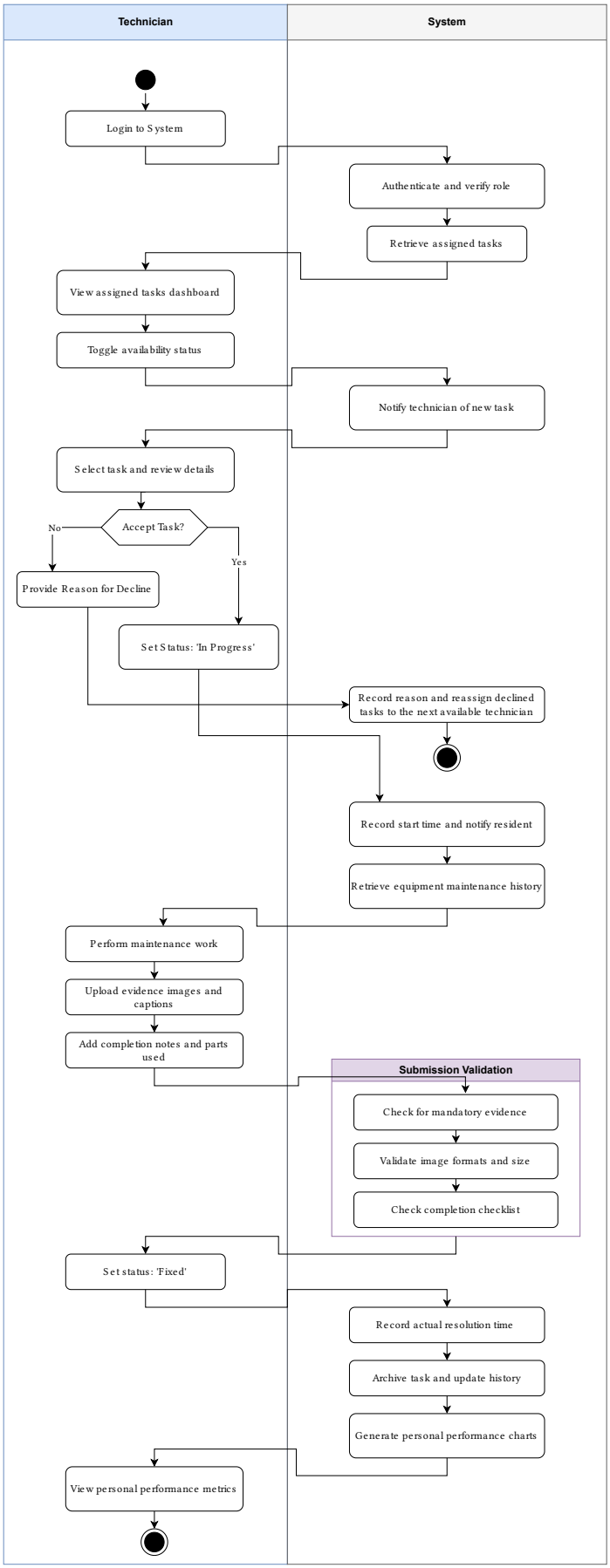


Figure 46: Technician Activity Diagram

4.4. System Architecture Design

4.4.1. Architectural Style

4.4.1.1. Primary Architecture Pattern: Three-Tier Client-Server Architecture

The SALLEHA system follows a **three-tier client-server architecture** with clear separation of concerns:

- **Presentation Tier**: Web (React) and Mobile (Flutter) applications
- **Application Tier**: RESTful API backend services (Node.js + Java)
- **Data Tier**: MySQL relational database with ORM layer

4.4.1.2. Architectural Characteristics

Layered Architecture: Each tier operates independently with well-defined interfaces:
- Enhances maintainability (NFR-6)
- Supports scalability for 24/7 operation (NFR-2)
- Enables parallel development across teams

Service-Oriented Design: Backend exposes RESTful APIs consumed by multiple clients (web + mobile), promoting reusability and platform independence.

Event-Driven Components: Real-time notification system using push notifications and email alerts for status changes.

4.4.2. Technology Stack

4.4.2.1. Frontend Technologies

4.4.2.1.1. Web Application

- **Framework**: React 18.x
- **UI Library**: Bootstrap 5.x + Tailwind CSS (for responsive design)
- **State Management**: React Context API / Redux (for complex state)
- **HTTP Client**: Axios
- **Form Validation**: Formik + Yup
- **Routing**: React Router v6
- **Real-time Updates**: WebSocket (Socket.io-client)
- **Charts/Analytics**: Recharts or Chart.js

4.4.2.1.2. Mobile Application

- **Framework**: Flutter 3.x
- **Language**: Dart
- **State Management**: Provider / Riverpod
- **HTTP Client**: Dio
- **Local Storage**: Shared Preferences
- **Push Notifications**: Firebase Cloud Messaging (FCM)
- **Image Handling**: image_picker, cached_network_image

4.4.2.2. Backend Technologies

4.4.2.2.1. API Server

- **Primary Runtime**: Node.js (v18 LTS) with Express.js framework
- **Secondary Services**: Java (Spring Boot) for computationally intensive analytics
- **API Architecture**: RESTful services with JSON data exchange
- **Authentication**: JWT (JSON Web Tokens) + bcrypt for password hashing
- **Session Management**: Redis (for session storage and caching)
- **File Upload**: Multer middleware (max 5MB per image)
- **Email Service**: Nodemailer with SMTP
- **SMS Service**: Twilio API (optional notifications)
- **Real-time Communication**: Socket.io for WebSocket connections

4.4.2.2.2. Database Layer

- **RDBMS**: MySQL 8.x
- **ORM**: Sequelize (Node.js) / Hibernate (Java)
- **Migration Management**: Sequelize CLI
- **Connection Pooling**: Built-in Sequelize pooling
- **Backup Strategy**: Automated daily backups with 90-day retention

4.4.2.3. Infrastructure & DevOps

- **Version Control**: GitHub (with Git Flow branching strategy)
- **CI/CD Pipeline**: GitHub Actions (automated testing + deployment)
- **Hosting**:
  - **Web**: Vercel or Netlify (static frontend hosting)
  - **API**: AWS EC2 or DigitalOcean Droplets
  - **Database**: AWS RDS MySQL or managed MySQL hosting
- **Cloud Storage**: AWS S3 (for ticket images and documents)
- **Monitoring**: PM2 (process management), CloudWatch (AWS monitoring)
- **Load Balancing**: Nginx (reverse proxy + load balancer)
- **SSL/TLS**: Let's Encrypt certificates

4.4.2.4. Development & Testing Tools

- **Code Editor**: Visual Studio Code
- **API Testing**: Postman, Thunder Client
- **Unit Testing**: Jest (Node.js), JUnit (Java), Flutter Test
- **Code Quality**: ESLint, Prettier, SonarQube
- **Documentation**: Typst (project docs), Swagger/OpenAPI (API docs)
- **Design & Prototyping**: Figma (UI/UX), Draw.io (diagrams)

4.4.3. Component/Module-Level View

4.4.3.1. System Components Diagram

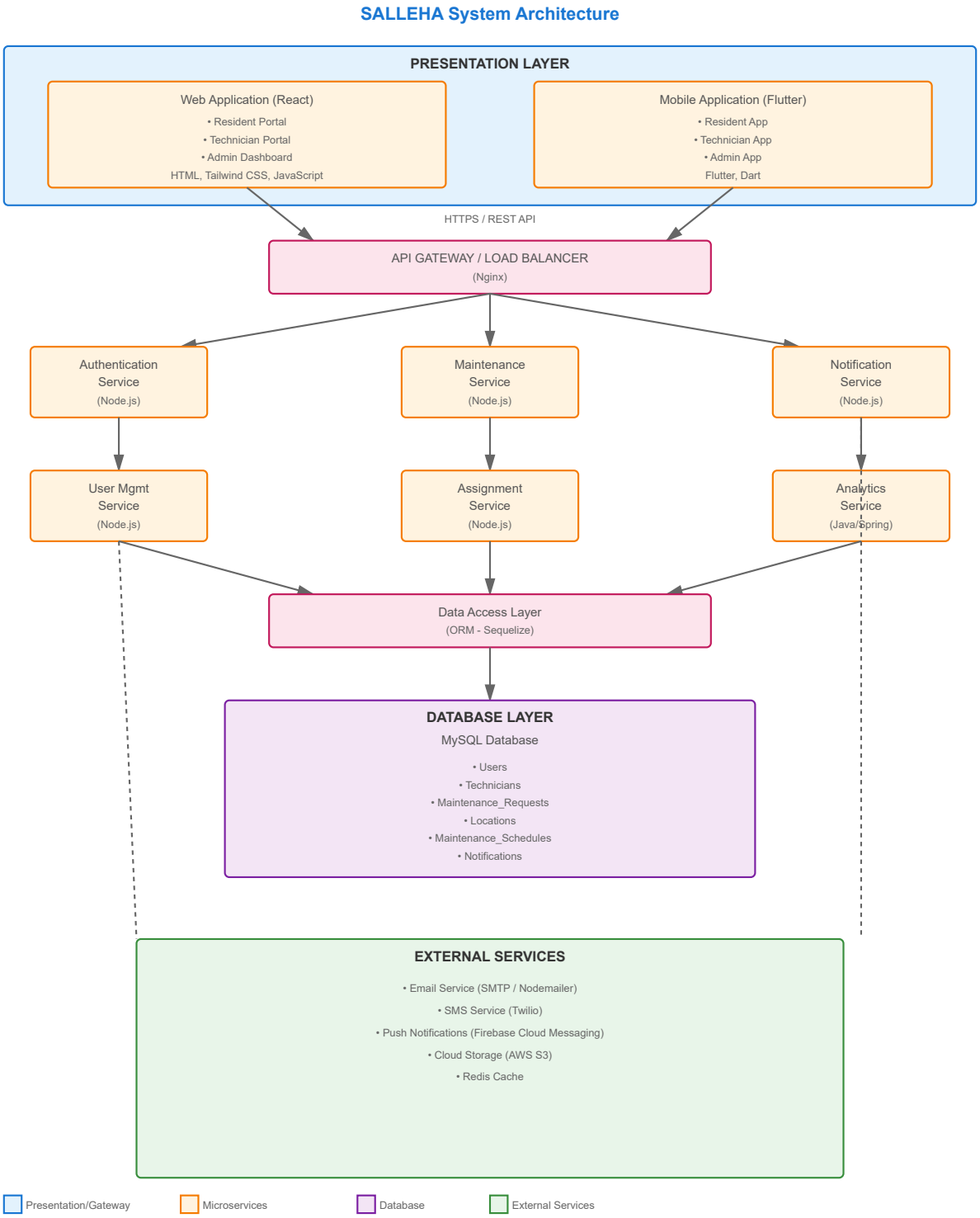


Figure 47: System Components Diagram

4.4.3.2. Core Modules and Components

4.4.3.2.1. MODULE 1: Authentication & Authorization Module

Responsibilities:

- User registration with email verification (FR-1)
- Login authentication with JWT token generation (FR-2)

- Password recovery with one-time verification codes (FR-4)
- Session management with role-based access control (RBAC)
- Account lockout after failed login attempts (FR-2.6)
- CAPTCHA integration for security (FR-2.5)

Components:
- `AuthController`: Handles authentication endpoints
- `TokenService`: JWT token generation and validation
- `PasswordHashService`: bcrypt password hashing
- `EmailVerificationService`: Sends and validates verification codes
- `SessionManager`: Redis-based session storage

Technologies: Node.js, JWT, bcrypt, Redis

4.4.3.2.2. MODULE 2: User Management Module

Responsibilities:
- User profile management (FR-12, FR-28)
- Role assignment and permissions management
- User account activation/deactivation (FR-28.2)
- Profile picture upload and storage
- Audit logging of user activities
- Bulk user operations (FR-28.5)

Components:
- `UserController`: User CRUD operations
- `ProfileService`: Profile update logic
- `RoleManager`: Role-based permission validation
- `AuditLogger`: Tracks all user management activities
- `FileUploadHandler`: Manages profile picture uploads

Technologies: Node.js, Multer, AWS S3

4.4.3.2.3. MODULE 3: Maintenance Request Management Module

Responsibilities:
- Ticket submission with validation (FR-8)
- Duplicate detection algorithm (FR-11)
- Ticket status lifecycle management (FR-9)
- Image upload handling (max 5 images, 5MB each)
- Real-time status tracking (FR-9)
- Maintenance request feed (FR-7)

Components:
- `TicketController`: Ticket CRUD operations
- `DuplicateDetectionService`: Similarity algorithm (category, location, keywords)
- `StatusManager`: Status transition workflow (Submitted → Assigned → In Progress → Fixed → Closed)
- `ImageUploadService`: Image validation, compression, S3 storage

- `TicketValidationService`: 20-character minimum description validation

Technologies: Node.js, Multer, AWS S3, Text similarity algorithms

4.4.3.2.4. MODULE 4: Task Assignment & Scheduling Module

Responsibilities:
- Automatic ticket assignment to technicians (FR-23)
- Manual assignment override by admins
- Priority-based task distribution (FR-24)
- Workload balancing across technicians
- Maintenance scheduling (FR for Maintenance Schedule entity)
- Escalation for overdue tasks

Components:
- `AssignmentEngine`: Auto-assignment algorithm (expertise, workload, location)
- `PriorityManager`: Priority level management (Low/Medium/High/Urgent/Emergency)
- `ScheduleService`: Technician schedule management
- `EscalationService`: Monitors and escalates overdue tasks
- `WorkloadBalancer`: Distributes tasks evenly

Technologies: Node.js, Scheduling algorithms

4.4.3.2.5. MODULE 5: Technician Workflow Module

Responsibilities:
- Task acceptance/decline by technicians (FR-15)
- Task status updates with timestamps (FR-16)
- Maintenance evidence submission (FR-17)
- Access to maintenance history (FR-19)
- Technician performance tracking (FR-20)

Components:
- `TechnicianTaskController`: Task management endpoints
- `EvidenceService`: Image/note uploads as completion proof
- `MaintenanceHistoryService`: Historical data retrieval
- `PerformanceTracker`: Completion rates, resolution time analytics
- `StatusUpdateService`: Handles status transitions with notifications

Technologies: Node.js, AWS S3

4.4.3.2.6. MODULE 6: Notification & Communication Module

Responsibilities:
- Real-time push notifications (FR-10, FR-18, FR-25)
- Email notifications for status changes
- SMS alerts for urgent tickets (optional)
- Notification preference management
- Notification history (90-day retention)

Components:
- `NotificationController`: Notification CRUD operations
- `PushNotificationService`: WebSocket (Socket.io) + FCM integration
- `EmailService`: Nodemailer SMTP integration
- `SMSService`: Twilio API integration
- `NotificationScheduler`: Batches and schedules notifications

Technologies: Node.js, Socket.io, Firebase Cloud Messaging, Nodemailer, Twilio

4.4.3.2.7. MODULE 7: Analytics & Reporting Module

Responsibilities:
- Admin analytics dashboard (FR-26)
- Maintenance trend analysis
- Equipment performance metrics (MTBF)
- Cost analysis (labor, parts, total expenditure)
- Predictive analytics for preventive maintenance
- Custom report generation (FR-27)

Components:
- `AnalyticsEngine`: Data aggregation and statistical analysis (Java/Spring Boot)
- `ReportGenerator`: PDF/Excel export functionality
- `TrendAnalyzer`: Pattern recognition for recurring issues
- `DashboardService`: KPI calculation (total tickets, resolution time, completion rates)
- `PredictiveModel`: ML-based preventive maintenance suggestions

Technologies: Java (Spring Boot), Apache POI (Excel), iText (PDF), Chart.js/Recharts

4.4.3.2.8. MODULE 8: Location & Equipment Management Module

Responsibilities:
- Location data management
- Geographic heat map generation (FR-21.5)
- Equipment tracking and history
- Zone-based assignment

Components:
- `LocationController`: Location CRUD operations
- `HeatMapService`: Generates issue density visualizations
- `EquipmentService`: Equipment history and warranty tracking
- `ZoneManager`: Manages work zones for technician assignment

Technologies: Node.js, Geolocation APIs

4.4.3.2.9. MODULE 9: Data Access Layer (DAL)

Responsibilities:
- Database connection management
- ORM-based CRUD operations
- Query optimization

- Data migration and seeding
- Connection pooling

Components:
- `UserRepository`: User entity operations
- `TicketRepository`: Maintenance request operations
- `TechnicianRepository`: Technician entity operations
- `LocationRepository`: Location entity operations
- `NotificationRepository`: Notification entity operations
- `DatabaseMigrator`: Schema migrations

Technologies: Sequelize ORM (Node.js), MySQL

4.4.4. Module Responsibilities Summary Table

Module	Primary Responsibility	Key Technologies	User Roles Affected
Authentication & Authorization	User login, registration, password recovery, session management	Node.js, JWT, bcrypt, Redis	All Users
User Management	Profile management, role assignment, account lifecycle	Node.js, AWS S3	All Users, Admin
Maintenance Request Management	Ticket submission, duplicate detection, status tracking	Node.js, AWS S3, Similarity algorithms	Resident, Admin
Task Assignment & Scheduling	Automatic/manual assignment, priority management, escalation	Node.js, Scheduling algorithms	Admin, Technician
Technician Workflow	Task acceptance, status updates, evidence upload, history	Node.js, AWS S3	Technician
Notification & Communication	Push notifications, email/SMS alerts, notification history	Socket.io, FCM, Nodemailer, Twilio	All Users
Analytics & Reporting	Dashboard analytics, trend analysis, report export	Java (Spring Boot), Apache POI, iText	Admin, Technician
Location & Equipment Management	Location tracking, heat maps, equipment history	Node.js, Geolocation APIs	Admin, Technician
Data Access Layer	Database operations, ORM, migrations	Sequelize, MySQL	Backend Services

4.4.5. Architectural Design Decisions

4.4.5.1. Why Three-Tier Architecture?

1. **Separation of Concerns**: Presentation, business logic, and data layers are independent
2. **Scalability**: Each tier can be scaled independently (e.g., add more API servers)
3. **Maintainability**: Easier to update one layer without affecting others (NFR-6)
4. **Security**: Database is isolated from direct client access (NFR-3)
5. **Multi-Platform Support**: Same API serves both web (React) and mobile (Flutter)

4.4.5.2. Why RESTful API?

1. **Platform Independence**: Web and mobile apps consume the same endpoints
2. **Statelessness**: Each request contains all necessary information (JWT tokens)
3. **Caching**: HTTP caching mechanisms improve performance (NFR-1)
4. **Industry Standard**: Well-documented and widely supported

4.4.5.3. Why Node.js + Java Hybrid Backend?

1. **Node.js**: Excellent for I/O-bound operations (ticket management, notifications)
2. **Java (Spring Boot)**: Better for CPU-intensive analytics and complex computations (FR-26)
3. **Team Expertise**: Leverages team skills (Table 6, 7)

4.4.5.4. Why MySQL?

1. **Relational Data**: Strong relationships between entities (Users, Tickets, Technicians)
2. **ACID Compliance**: Ensures data integrity for critical operations (NFR-2)
3. **Mature Ecosystem**: Excellent tooling, backup, and recovery options
4. **Cost-Effective**: Open-source with strong community support

4.4.5.5. Why Redis for Sessions?

1. **Performance**: In-memory storage for fast session retrieval (NFR-1)
2. **Expiration**: Built-in TTL for session timeout (FR-2.7)
3. **Scalability**: Supports distributed caching for multiple API servers

4.4.6. Security Architecture

4.4.6.1. Security Measures

1. **Authentication**: JWT tokens with 15-60 minute expiration
2. **Authorization**: Role-based access control (RBAC) middleware
3. **Data Encryption**:
   - Passwords: bcrypt hashing (cost factor 12)
   - Data in transit: HTTPS/TLS 1.3
   - Sensitive data at rest: AES-256 encryption
4. **Input Validation**: Server-side validation for all inputs (SQL injection prevention)
5. **File Upload Security**:
   - MIME type validation
   - File size limits (5MB for images)
   - Malware scanning (ClamAV integration)
6. **Rate Limiting**: API rate limiting to prevent DDoS attacks
7. **CAPTCHA**: After 3 failed login attempts (FR-2.5)
8. **Account Lockout**: After 5 failed attempts (FR-2.6)
9. **Audit Logging**: All sensitive operations logged with timestamps and IP addresses

4.4.6.2. Data Privacy

1. **Email Verification**: Required before account activation (FR-1.7)
2. **Anonymous Reporting**: Residents can submit tickets anonymously (Constraint 1.2.3.4)
3. **90-Day Notification History**: Automatic deletion after retention period
4. **Data Export**: Users can export their profile data (FR-12.7)

4.4.7. Performance Optimization Strategies

1. **Database Indexing**: Indexes on TicketID, UserID, LocationID, Status
2. **Query Optimization**: ORM query optimization, eager loading
3. **Caching**: Redis caching for frequently accessed data (user sessions, notifications)
4. **Image Compression**: Automatic compression before S3 upload
5. **Pagination**: Results limited to 20 items per page (FR-7.8)
6. **Lazy Loading**: Frontend lazy loading of images and components
7. **Connection Pooling**: Database connection pooling (Sequelize built-in)
8. **CDN**: Static assets (CSS, JS) served via CDN
9. **Load Balancing**: Nginx distributes traffic across multiple API servers
10. **Asynchronous Processing**: Background jobs for email/SMS notifications

4.4.8. Deployment Architecture

4.4.8.1. Production Environment

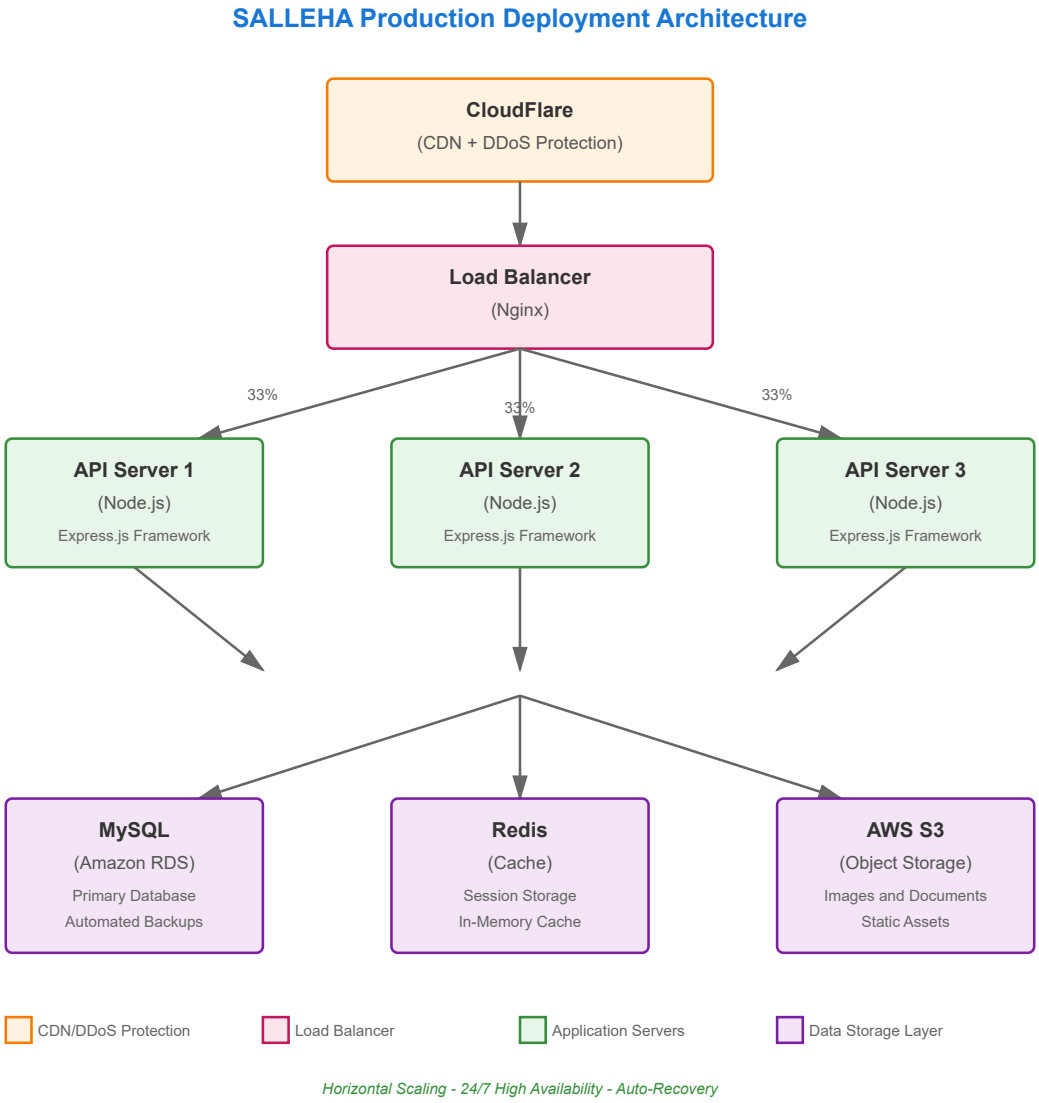


Figure 48: Deployment Diagram

4.4.8.2. Continuous Integration/Deployment

1. **Development**: Feature branches → Pull Request → Code Review
2. **Testing**: Automated tests via GitHub Actions (Jest, JUnit)
3. **Staging**: Deploy to staging environment for QA testing
4. **Production**: Blue-green deployment to minimize downtime

4.4.9. Monitoring & Observability

1. **Application Monitoring**: PM2 for Node.js process management
2. **Performance Monitoring**: AWS CloudWatch metrics (CPU, memory, response time)
3. **Error Tracking**: Sentry for error logging and alerting
4. **Log Aggregation**: ELK Stack (Elasticsearch, Logstash, Kibana)
5. **Uptime Monitoring**: Pingdom or UptimeRobot (24/7 availability check)
6. **Database Monitoring**: Query performance, slow query logs

4.4.10. Scalability Considerations

1. **Horizontal Scaling**: Add more API servers behind load balancer
2. **Database Read Replicas**: MySQL read replicas for analytics queries
3. **Microservices Migration**: Future migration to microservices if needed
4. **Message Queue**: RabbitMQ/Redis for asynchronous task processing
5. **Auto-Scaling**: AWS Auto Scaling Groups based on traffic patterns

4.4.11. Alignment with Non-Functional Requirements

NFR	Architectural Support
Performance (NFR-1)	Redis caching, database indexing, pagination, CDN, load balancing
Dependability (NFR-2)	24/7 hosting, automated backups, health monitoring, redundant servers
Security (NFR-3)	JWT auth, bcrypt, HTTPS, input validation, rate limiting, RBAC
Usability (NFR-4)	React/Flutter for intuitive UIs, responsive design, error messages
Operational Constraints (NFR-5)	Web browsers (React), mobile devices (Flutter), stable internet
Maintainability (NFR-6)	Modular architecture, ORM, comprehensive logging, CI/CD pipeline

4.4.12. Future Enhancements

1. **GraphQL API**: For more flexible data querying
2. **Microservices**: Split monolithic API into independent services
3. **Machine Learning**: Predictive maintenance models
4. **Offline Mode**: Mobile app offline functionality with sync
5. **WebAssembly**: Performance-critical frontend components
6. **Kubernetes**: Container orchestration for better scalability

4.5. Object-Oriented Design

4.5.1. Class Diagram

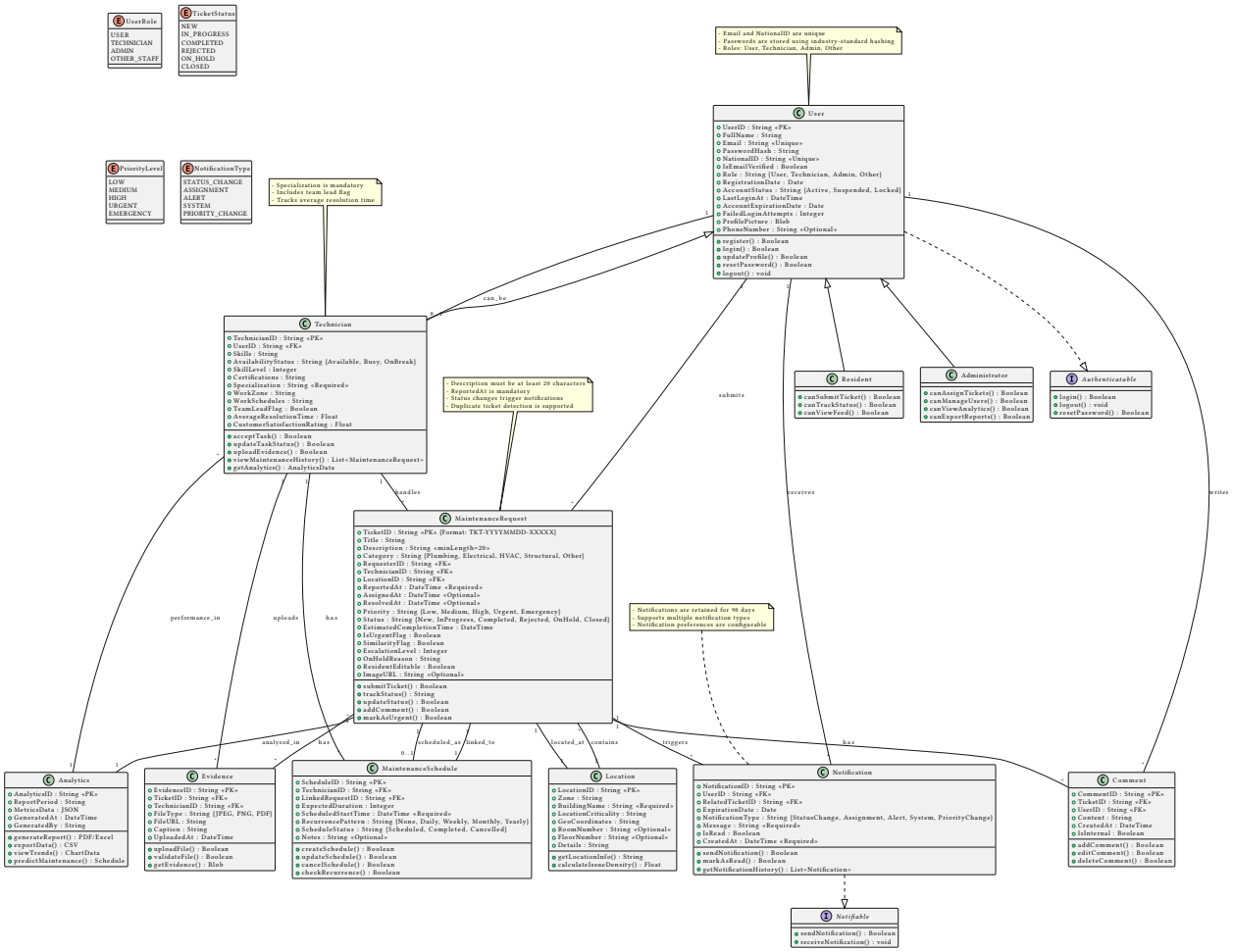


Figure 49: Class Diagram

4.5.2. Sequence Diagrams

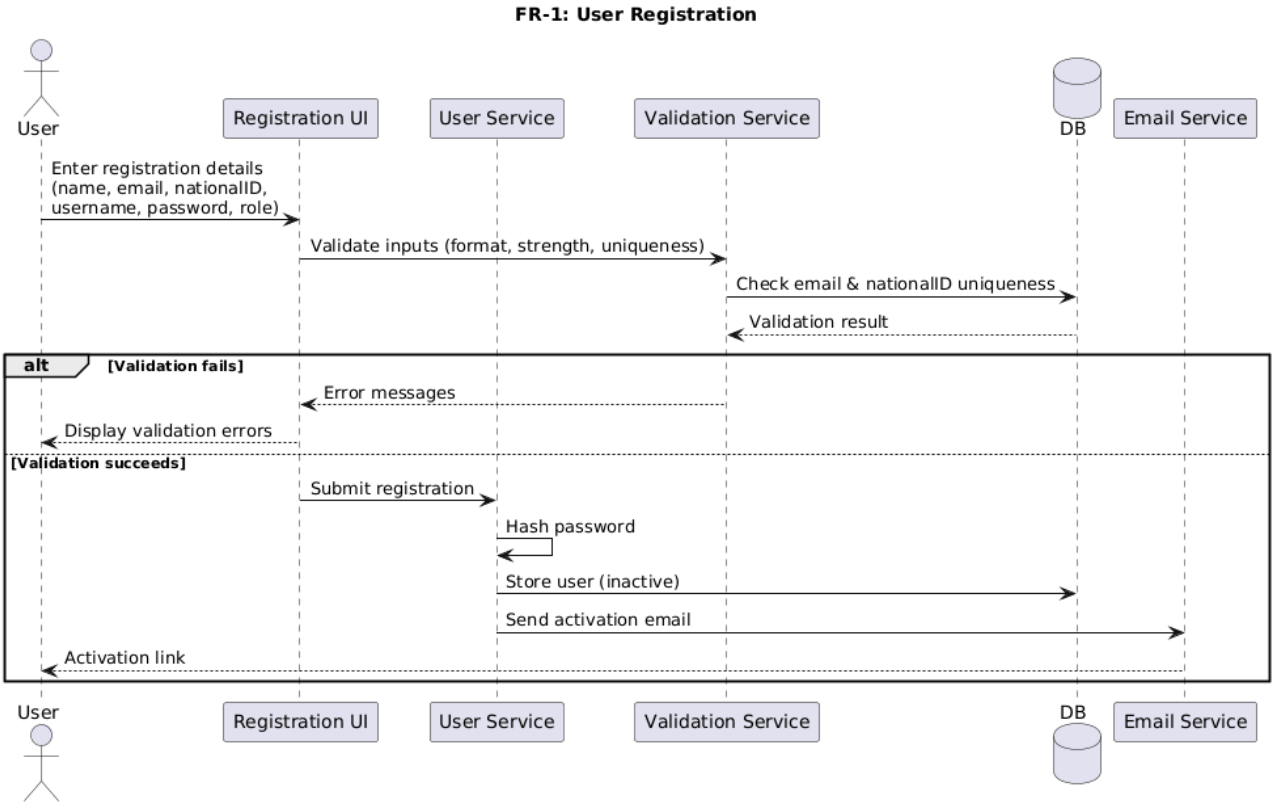


Figure 50: FR-1 Sequence Diagram

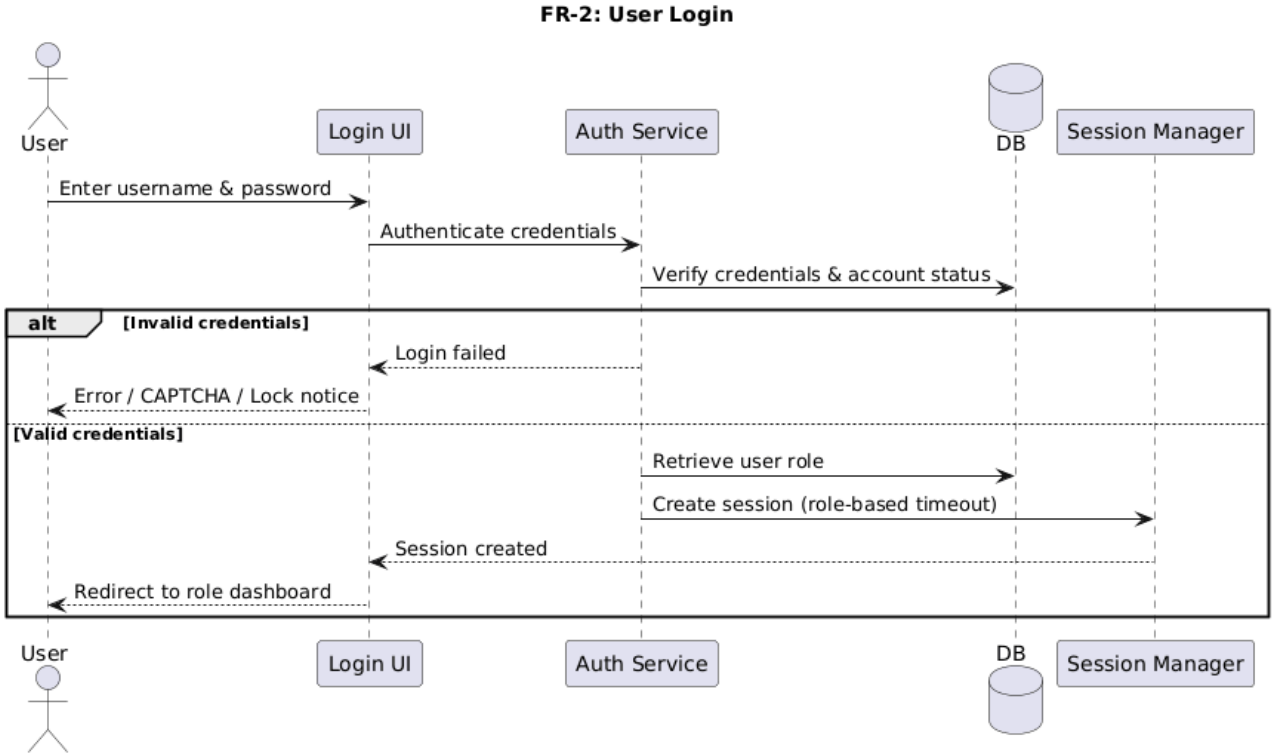


Figure 51: FR-2 Sequence Diagram

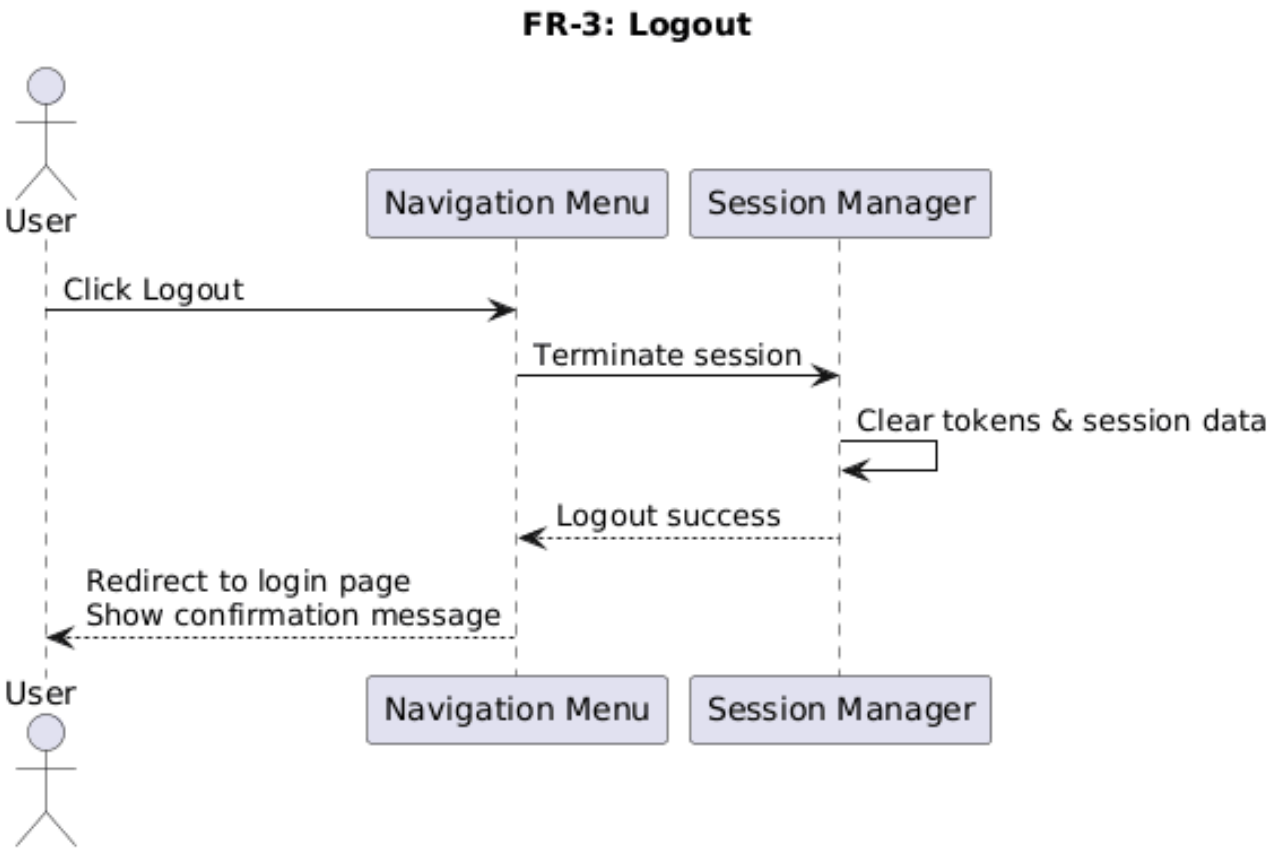


Figure 52: FR-3 Sequence Diagram

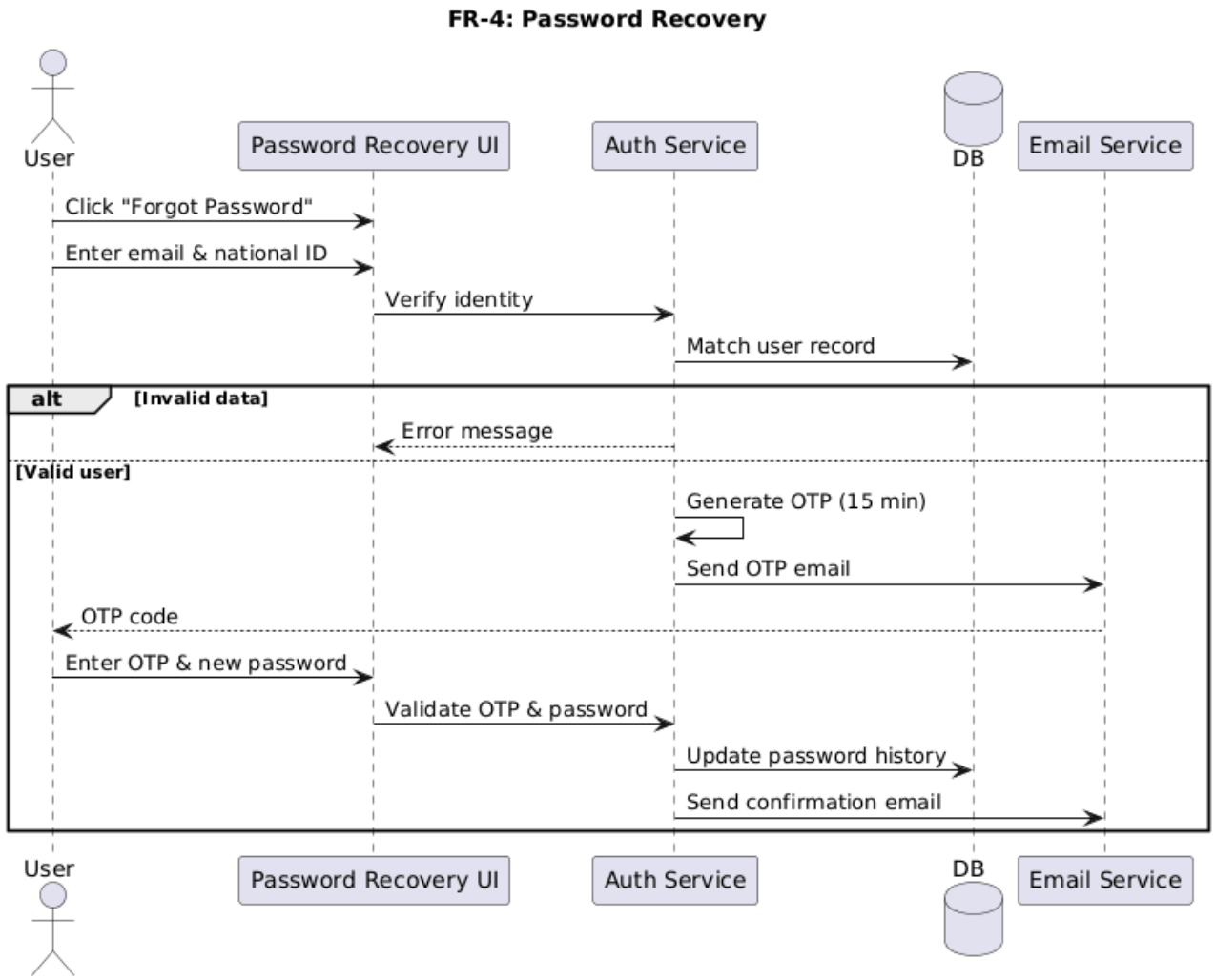


Figure 53: FR-4 Sequence Diagram

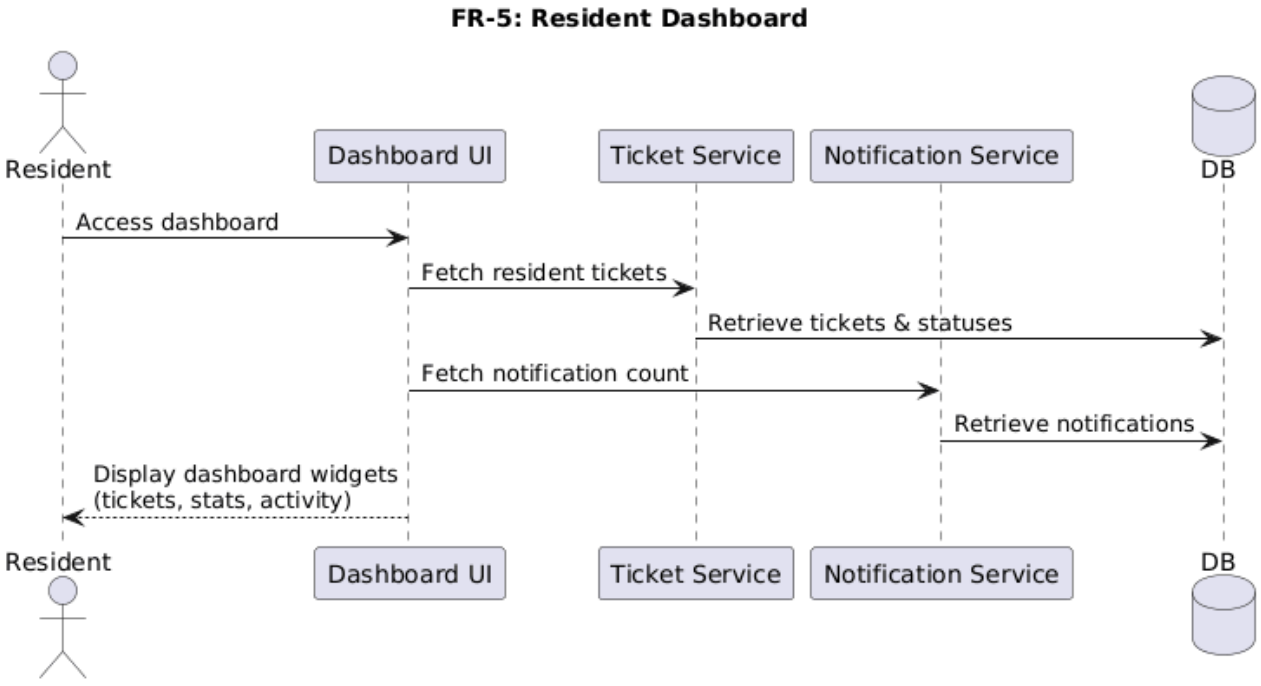


Figure 54: FR-5 Sequence Diagram

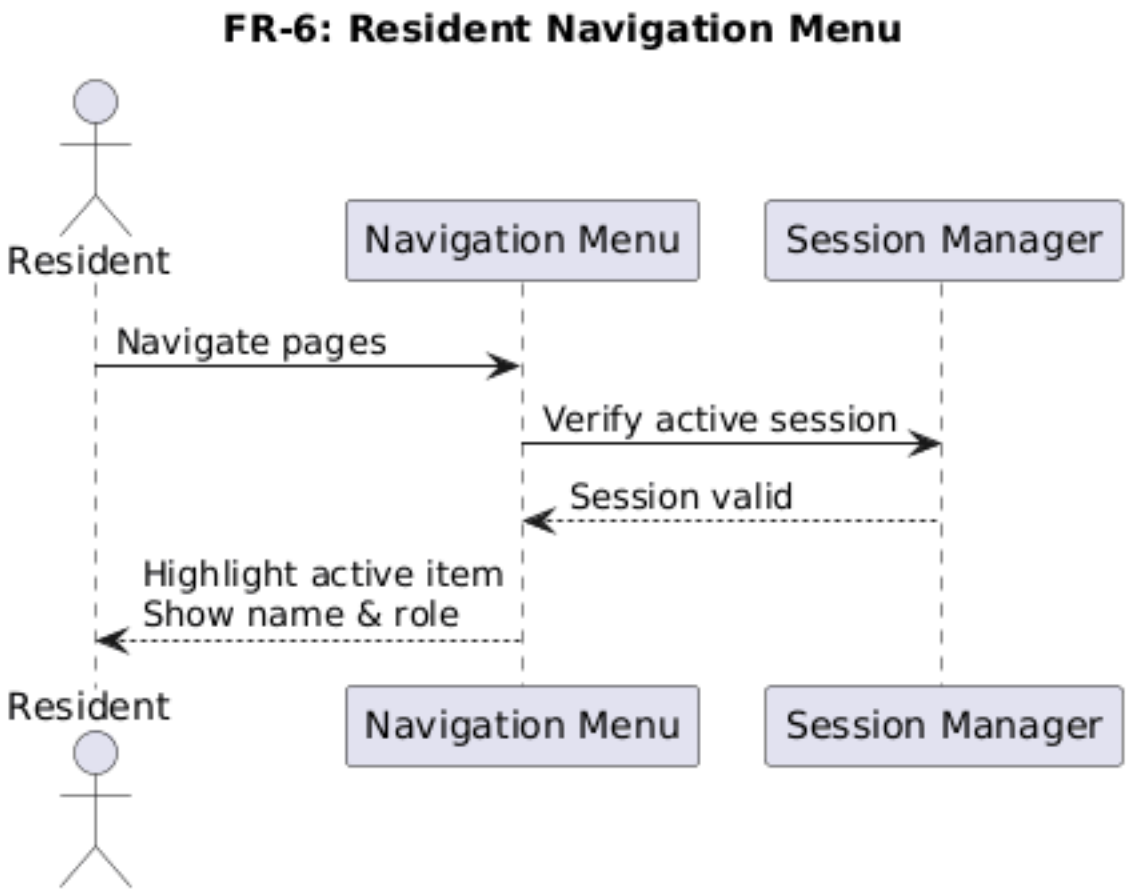


Figure 55: FR-6 Sequence Diagram

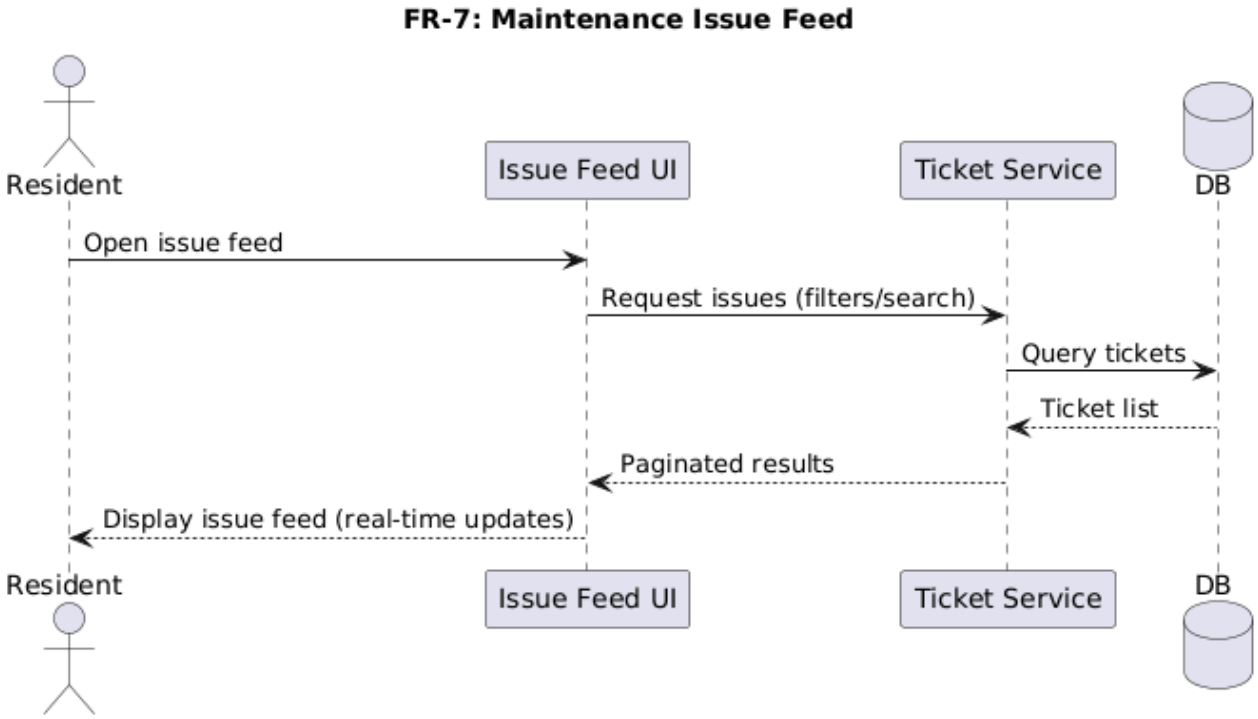


Figure 56: FR-7 Sequence Diagram

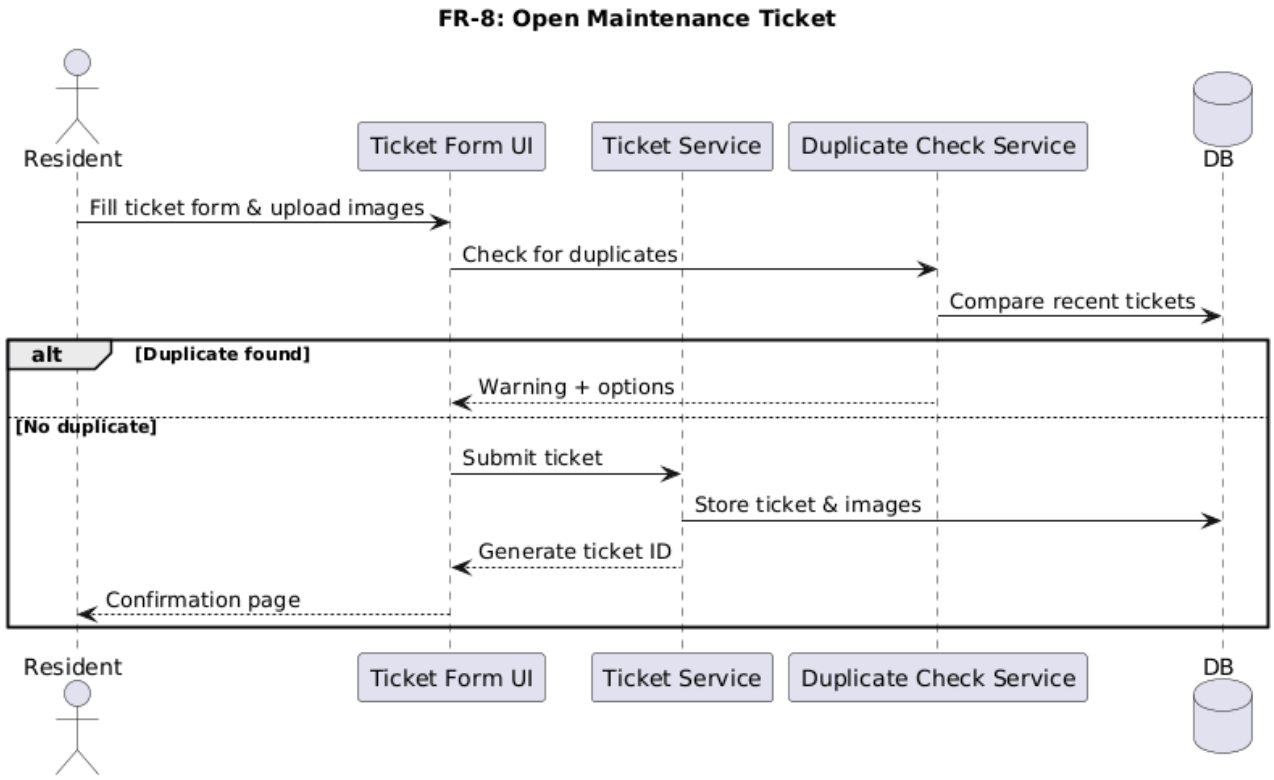


Figure 57: FR-8 Sequence Diagram

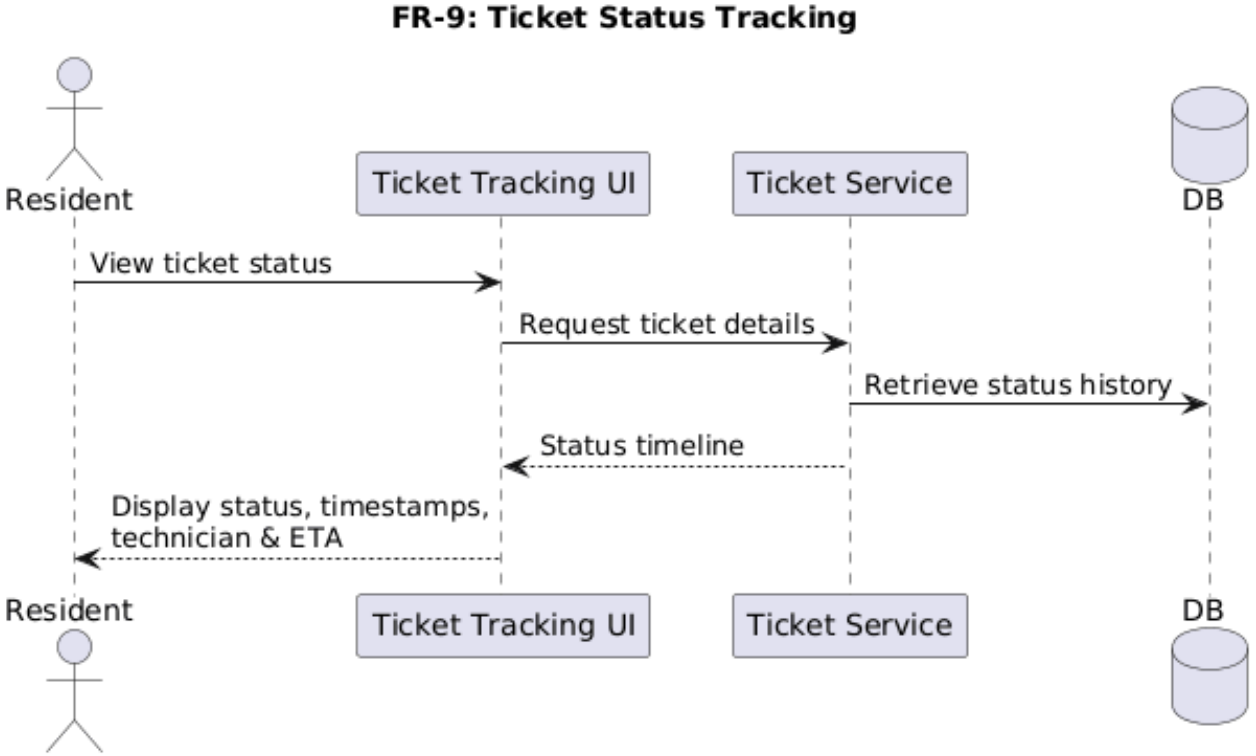


Figure 58: FR-9 Sequence Diagram

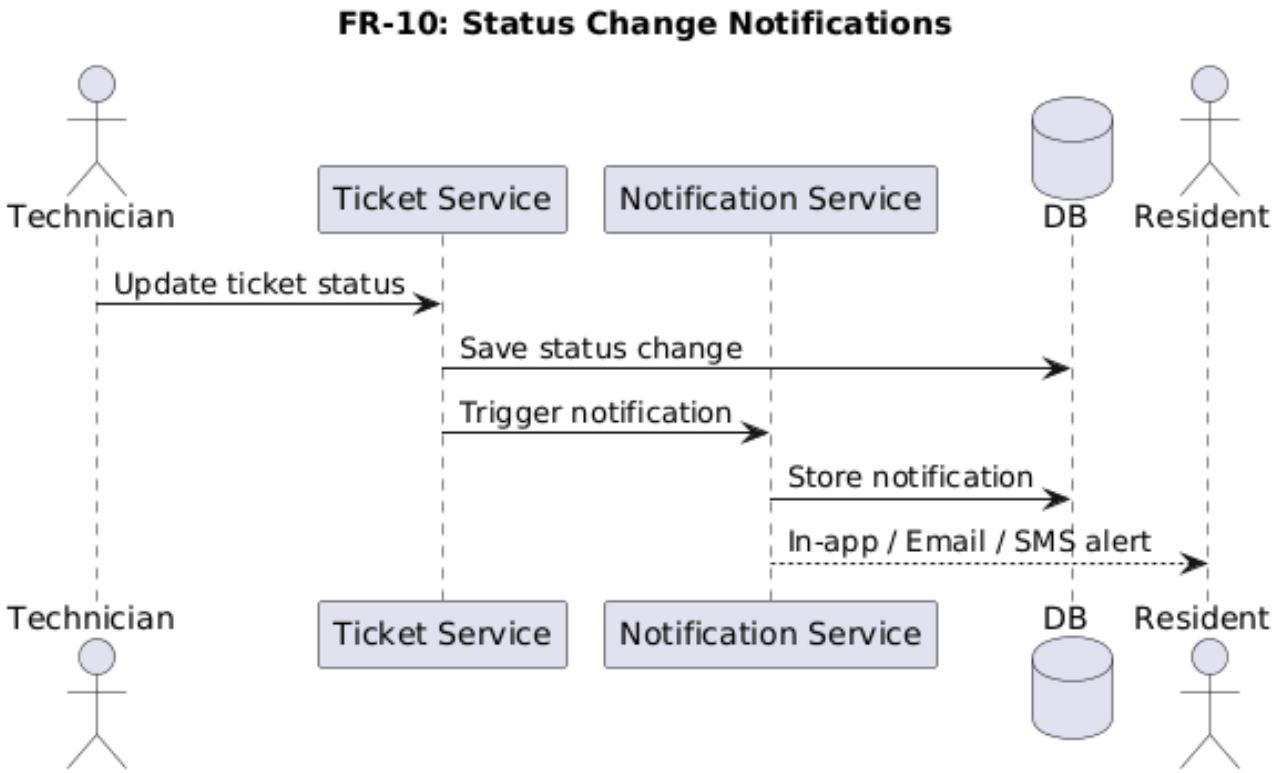


Figure 59: FR-10 Sequence Diagram

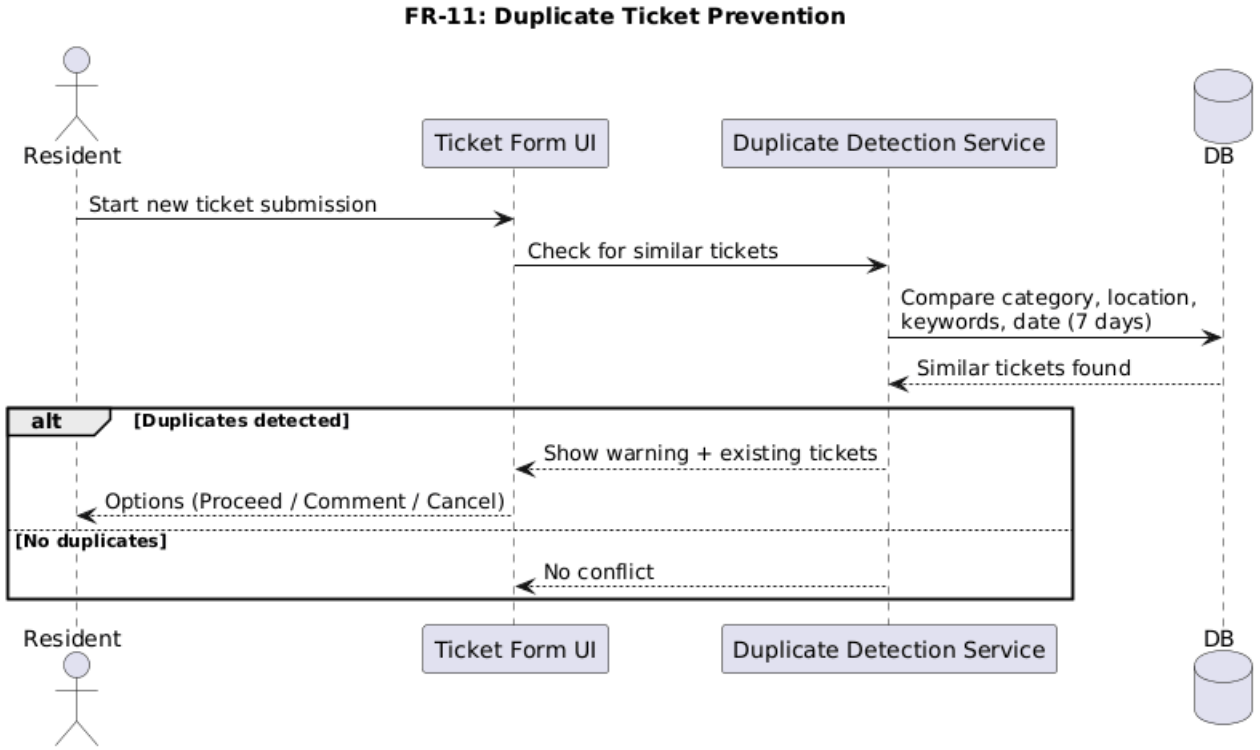


Figure 60: FR-11 Sequence Diagram

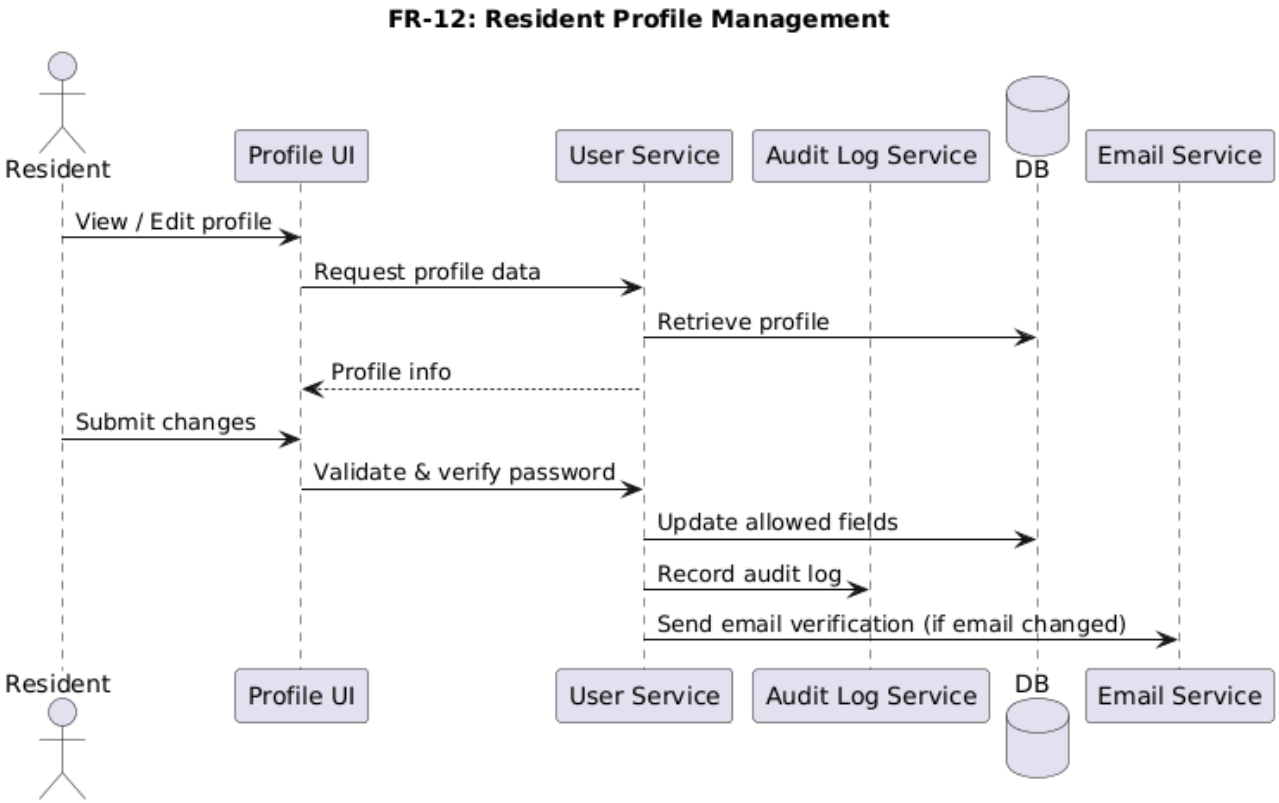


Figure 61: FR-12 Sequence Diagram

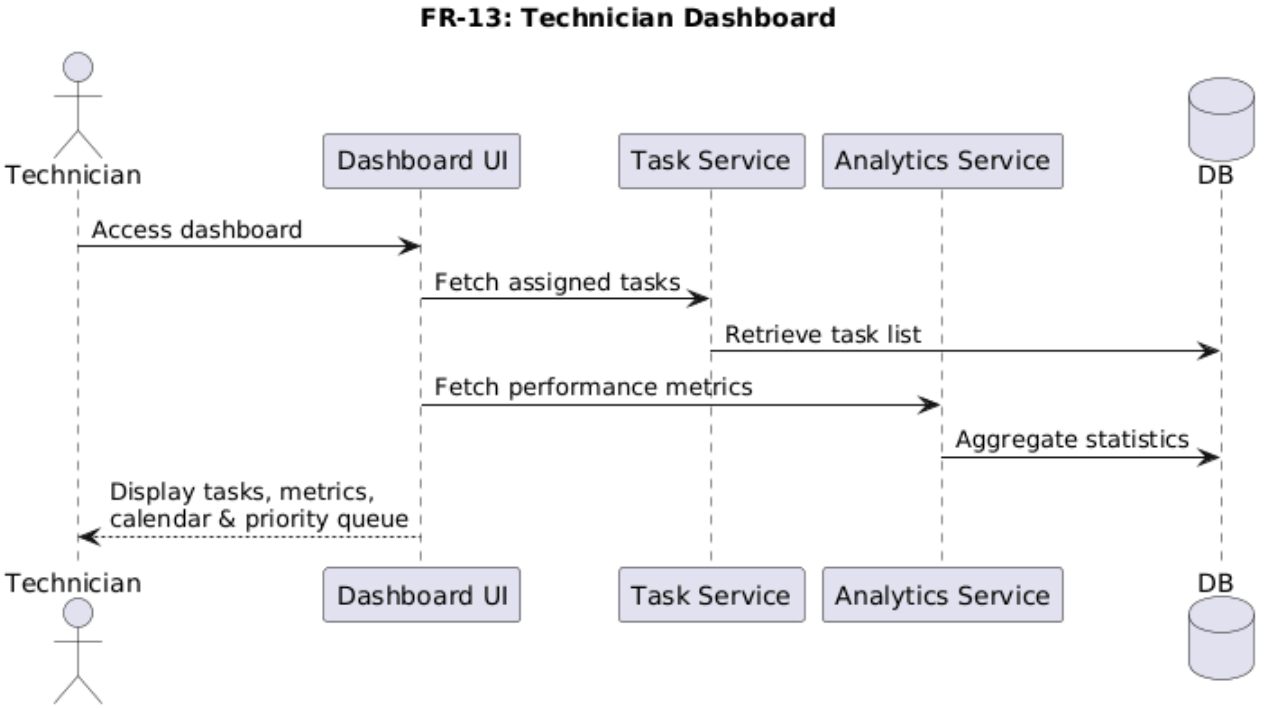


Figure 62: FR-13 Sequence Diagram

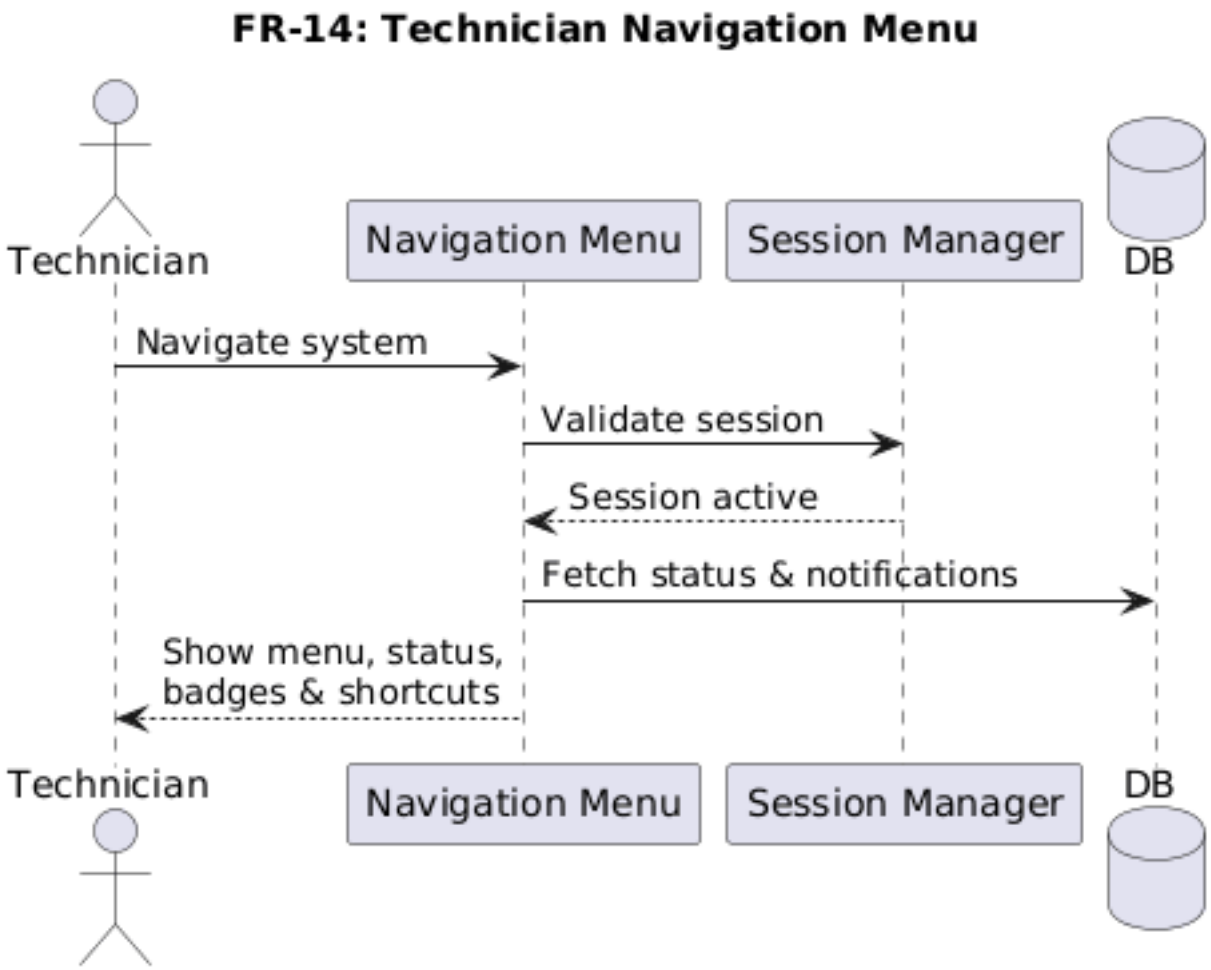


Figure 63: FR-14 Sequence Diagram

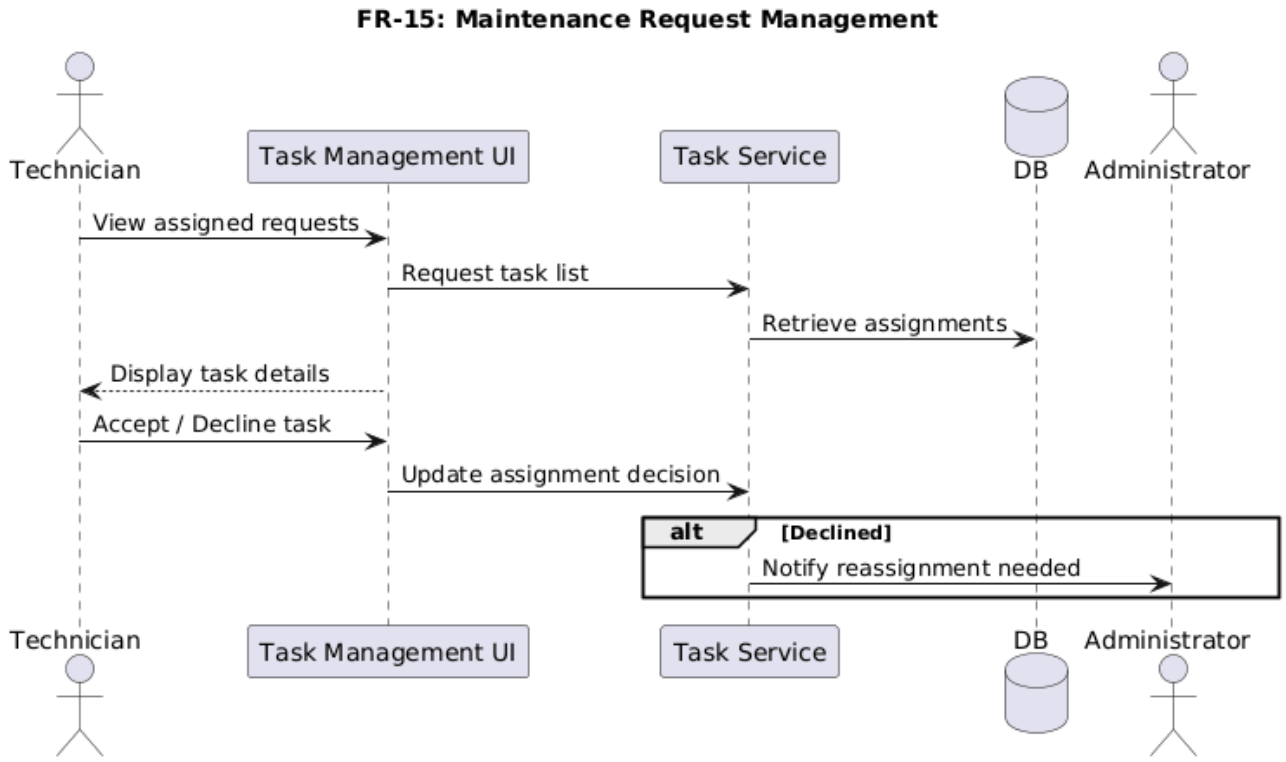


Figure 64: FR-15 Sequence Diagram

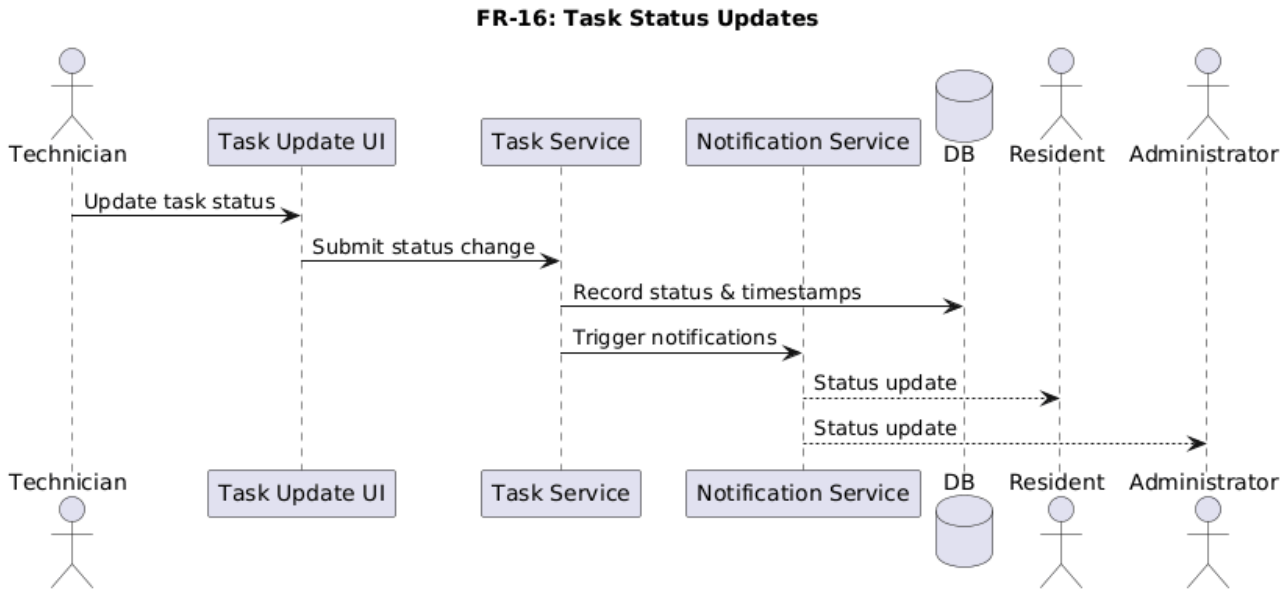


Figure 65: FR-16 Sequence Diagram

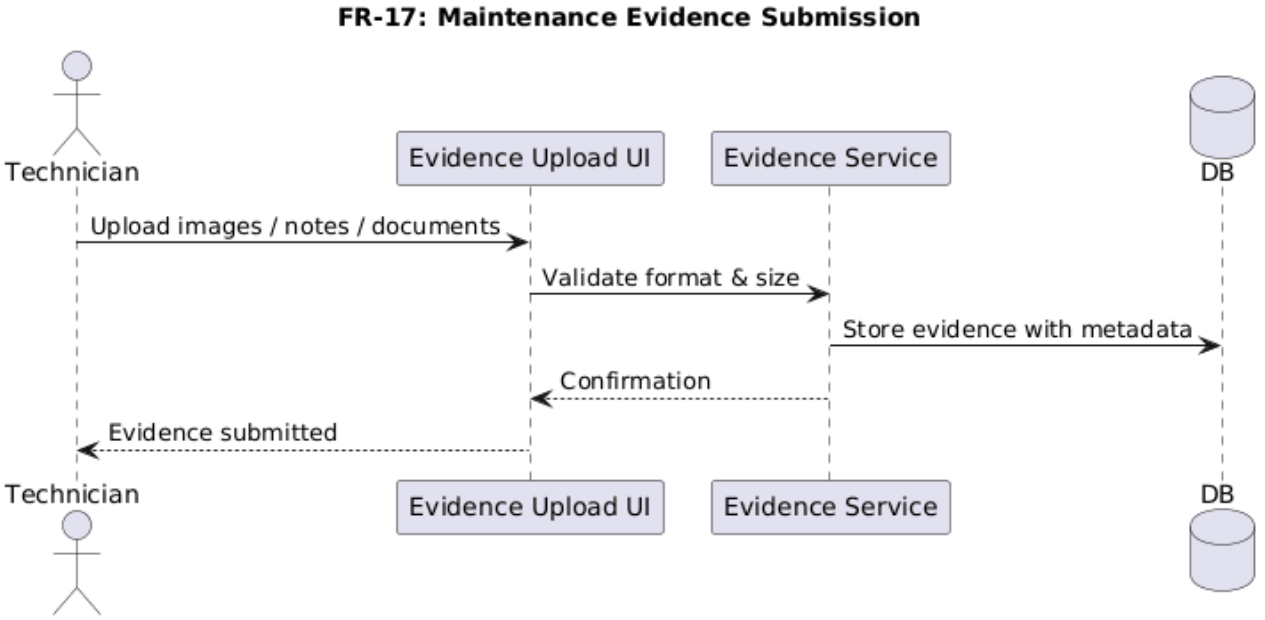


Figure 66: FR-17 Sequence Diagram

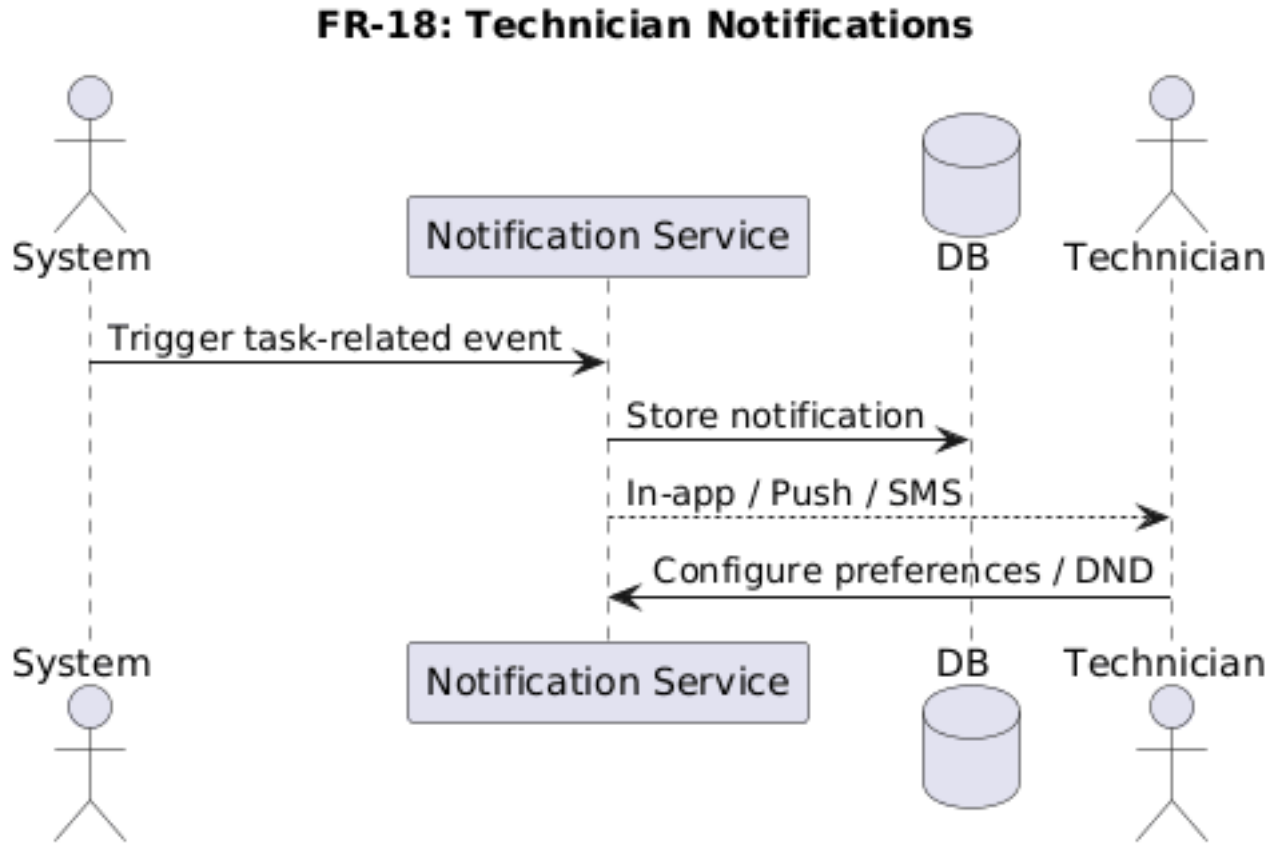


Figure 67: FR-18 Sequence Diagram

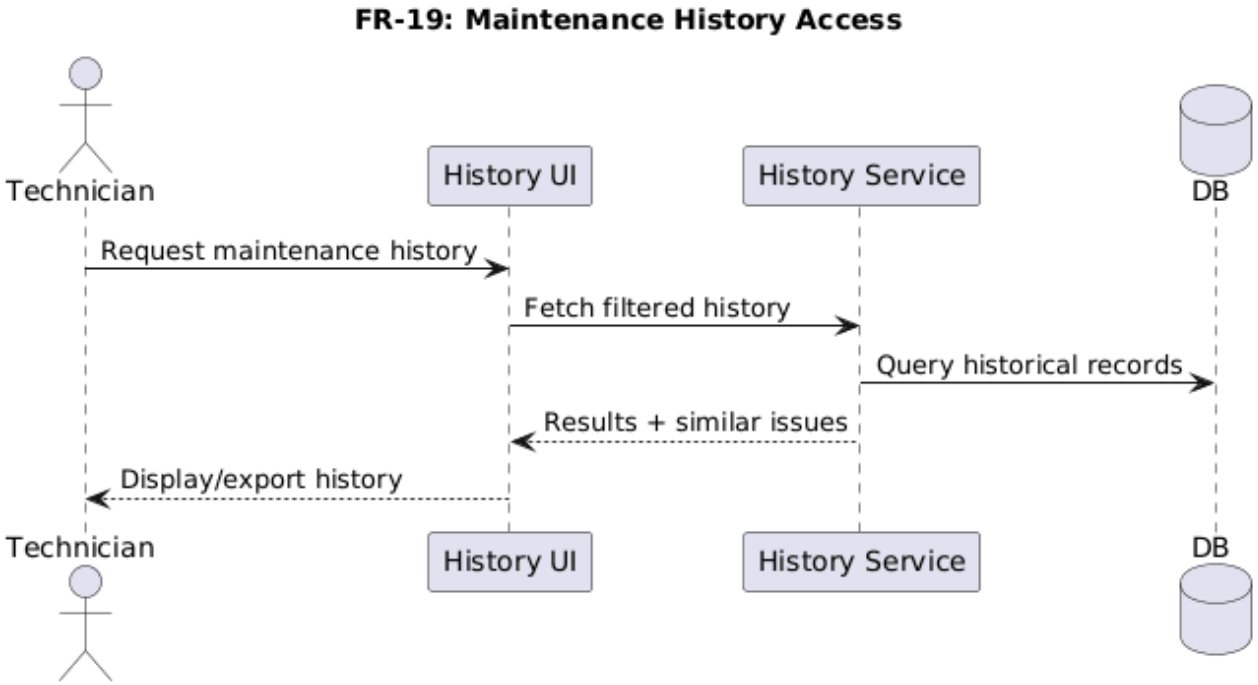


Figure 68: FR-19 Sequence Diagram

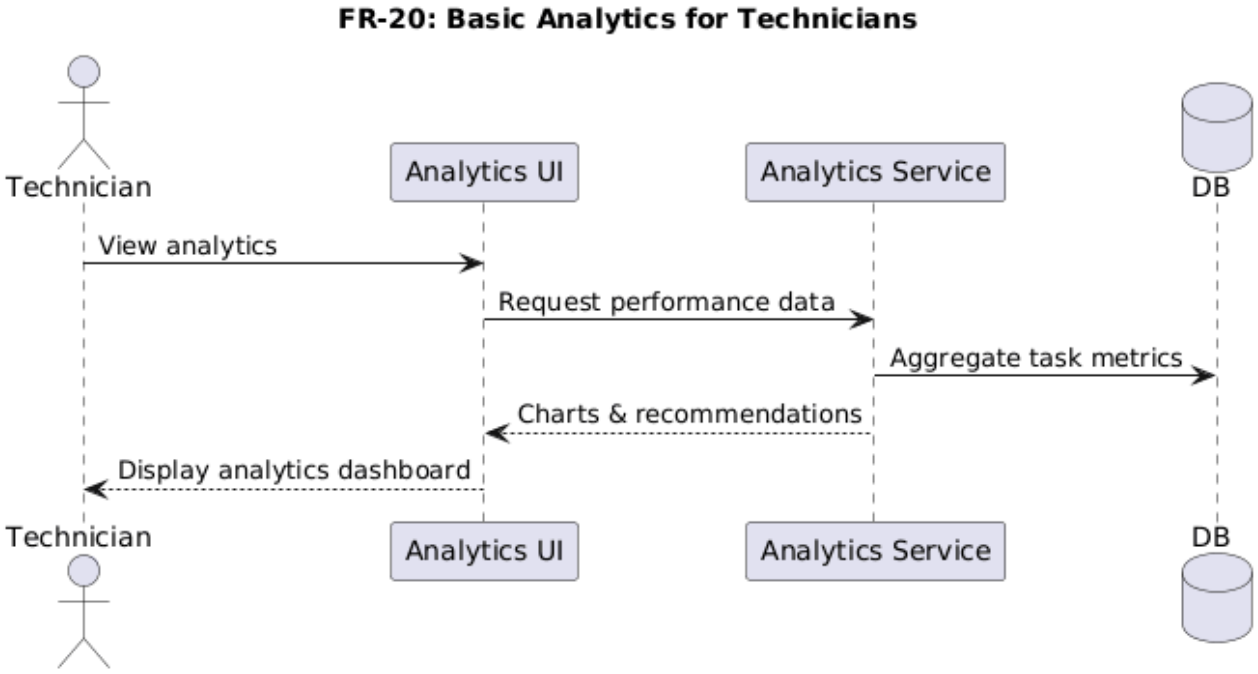


Figure 69: FR-20 Sequence Diagram

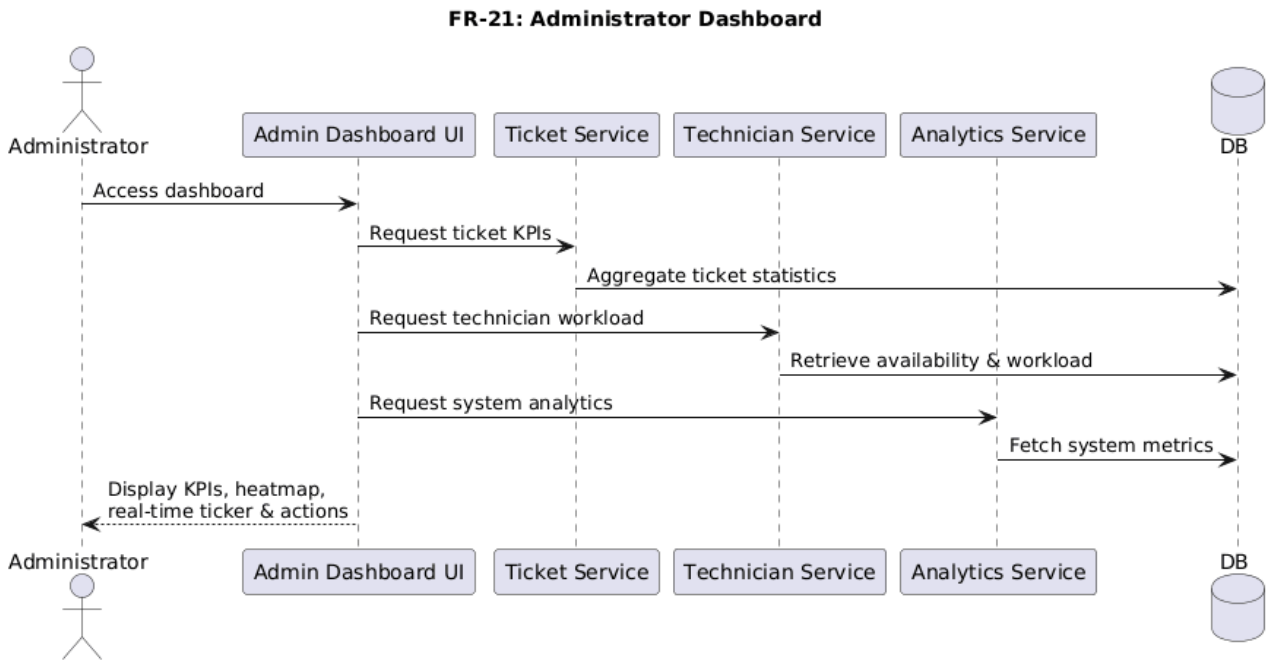


Figure 70: FR-21 Sequence Diagram

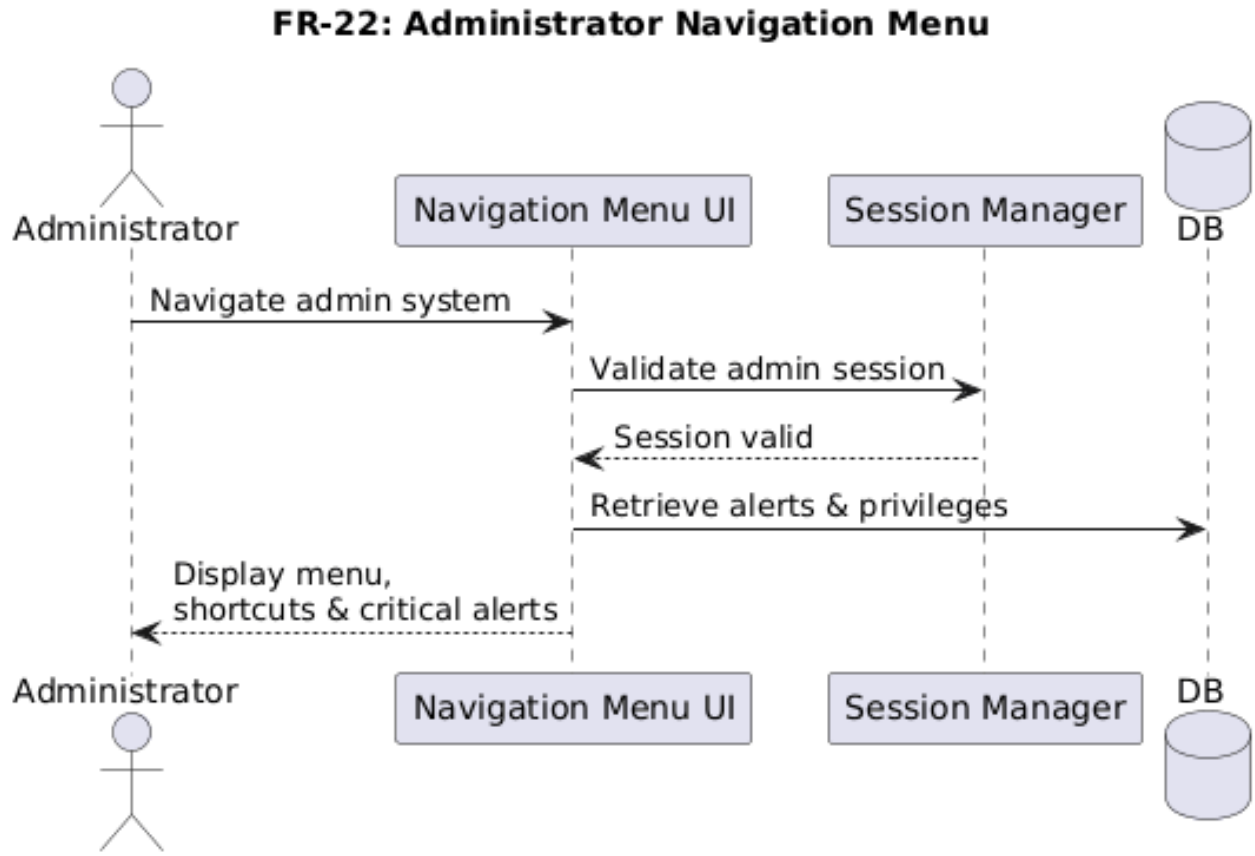


Figure 71: FR-22 Sequence Diagram

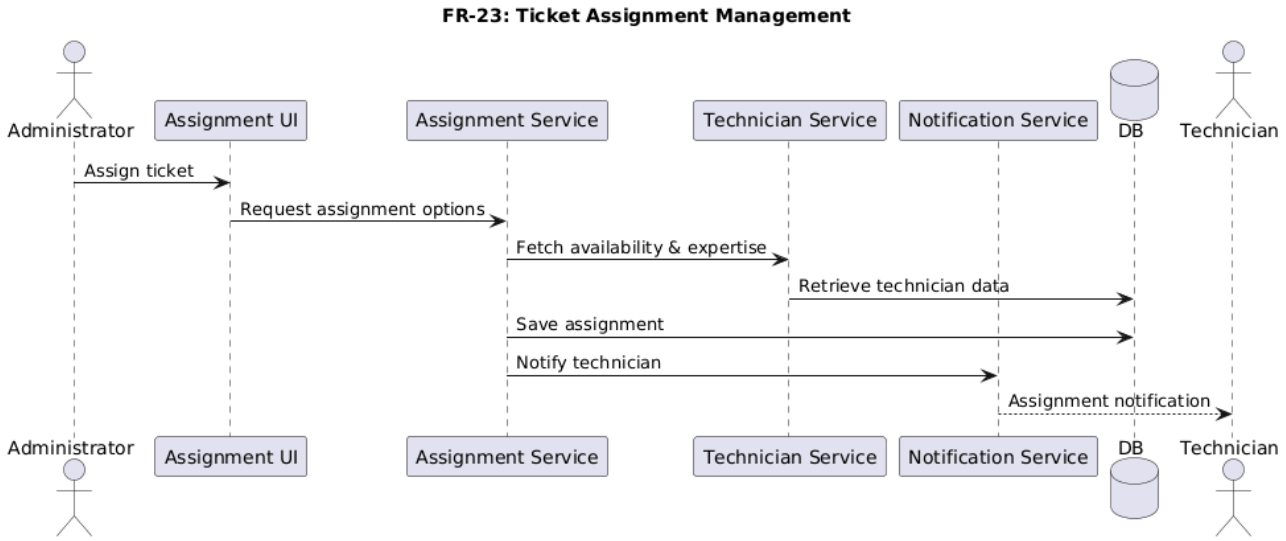


Figure 72: FR-23 Sequence Diagram

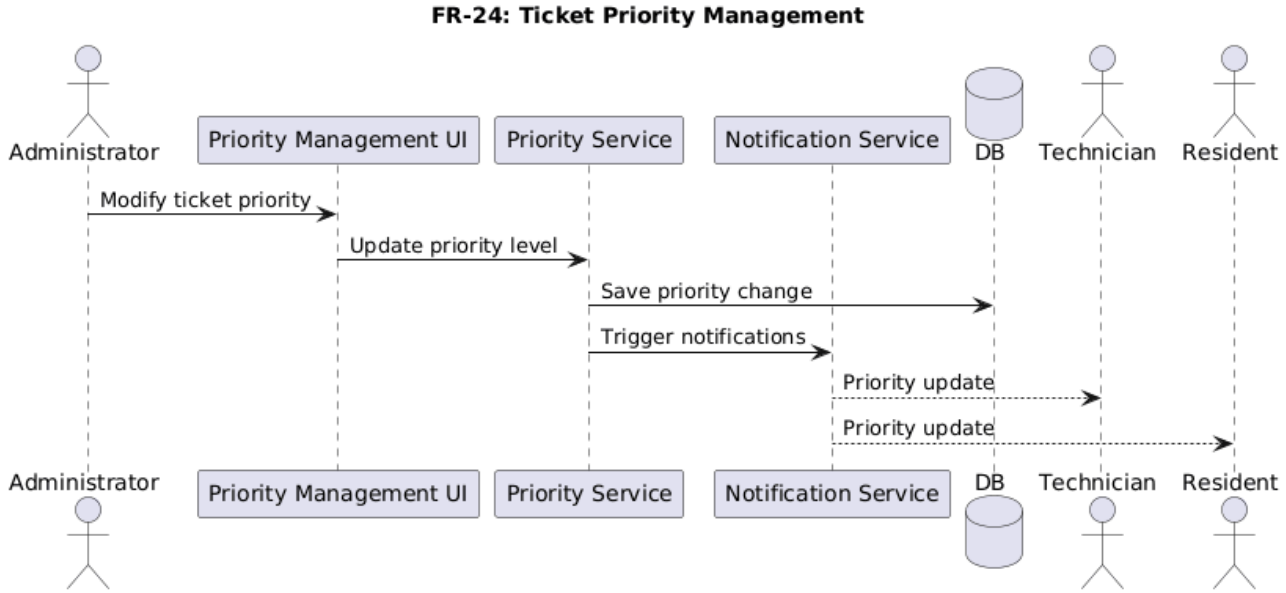


Figure 73: FR-24 Sequence Diagram

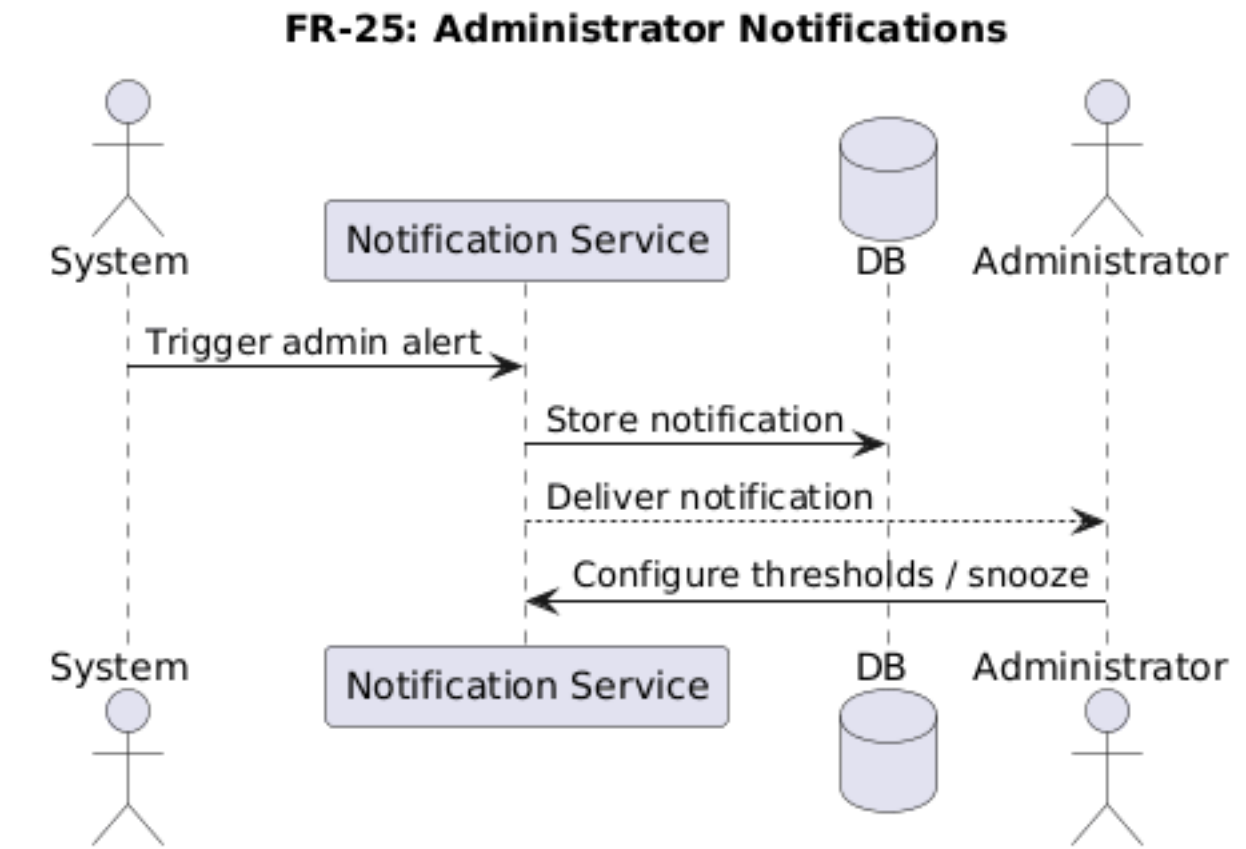


Figure 74: FR-25 Sequence Diagram

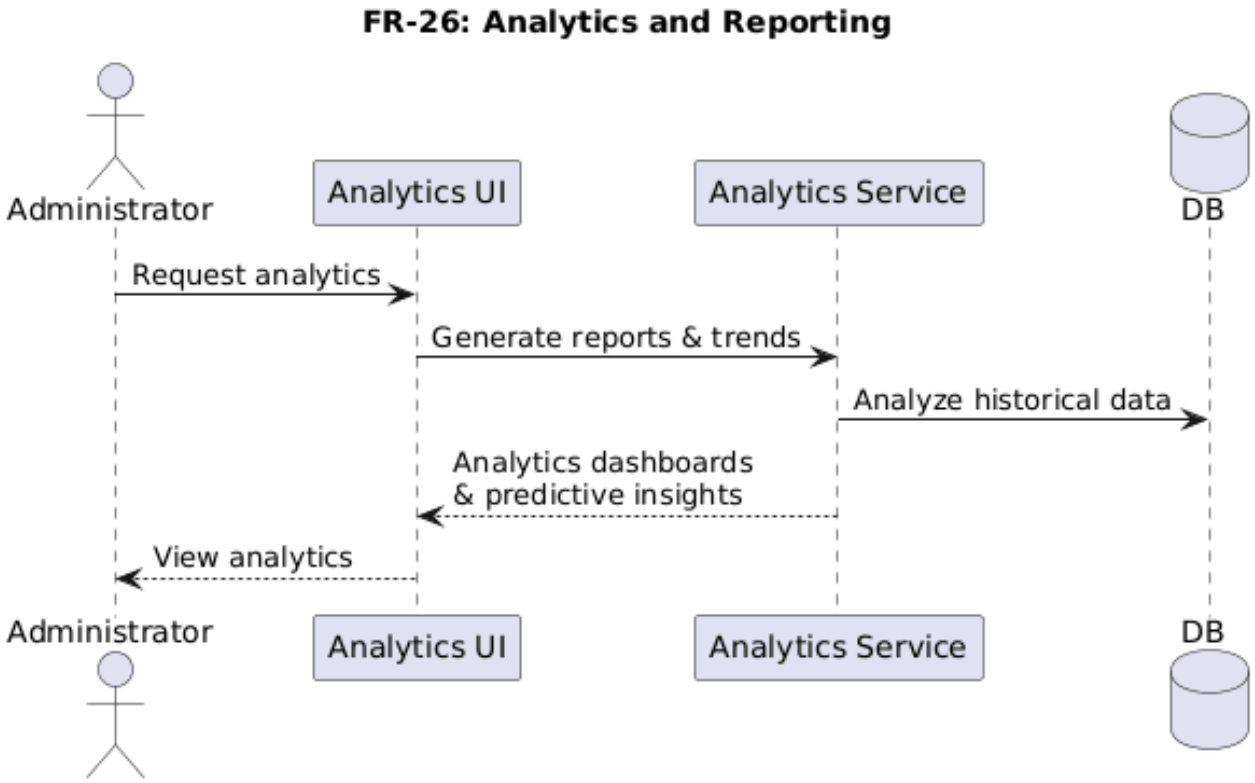


Figure 75: FR-26 Sequence Diagram

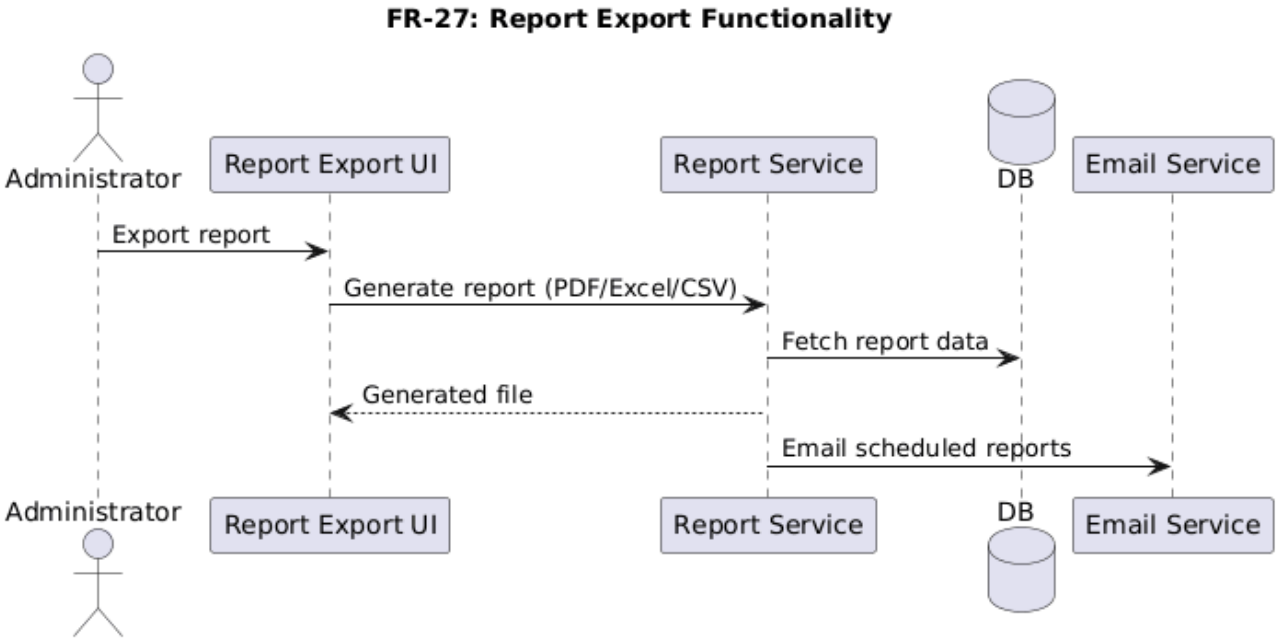


Figure 76: FR-27 Sequence Diagram

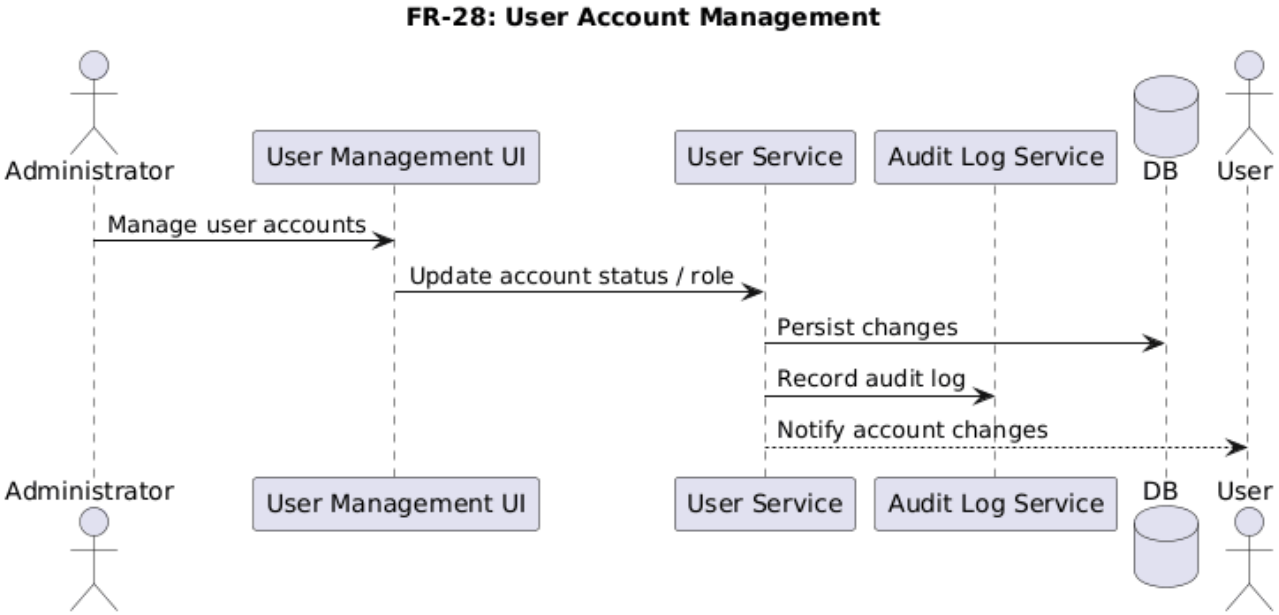


Figure 77: FR-28 Sequence Diagram

4.5.3. Classes and Components Design

Class Specifications

1. User Classes (Abstract Base Class)

Class: User Attributes:

- userID: String (private)
- fullName: String (private)
- email: String (private)
- nationalID: String (private)

- username: String (private)
- passwordHash: String (private)
- phoneNumber: String (private, optional)
- role: String (private)
- isEmailVerified: Boolean (private)
- accountStatus: String (private)
- failedLoginAttempts: Integer (private)
- lastLoginAt: DateTime (private)
- sessionToken: String (private)
- registrationDate: DateTime (private)

Methods:
1. register(fullName: String, email: String, nationalID: String, username: String, password: String, role: String): Boolean
2. login(username: String, password: String, captcha: String): Boolean
3. logout(): Boolean
4. resetPassword(email: String, nationalID: String): Boolean
5. verifyEmail(token: String): Boolean
6. updateProfile(fullName: String, email: String, password: String): Boolean

- isAccountLocked(): Boolean
- validateCredentials(): Boolean
- hashPassword(password: String): String
- validatePasswordStrength(password: String): Boolean
- incrementFailedLoginAttempts(): void
- resetFailedLoginAttempts(): void

Responsibilities:
- Handle user registration, authentication, and session management
- Enforce password security and account locking rules
- Provide base functionality for all system user roles

2. Resident Class (Inherits User)

Class: Resident Attributes:
- profilePictureURL: String (private)

Methods:
1. submitMaintenanceRequest(category: String, description: String, locationID: String, priority: String, images: List): String
2. viewDashboard(): void
3. viewTicketStatus(ticketID: String): String
4. addTicketComment(ticketID: String, comment: String): Boolean
5. markTicketAsUrgent(ticketID: String): Boolean
6. configureNotificationPreferences(): void

Responsibilities:
- Submit and track maintenance requests
- Monitor ticket status and history
- Receive and manage notifications related to submitted tickets

3. Technician Class (Inherits User)

Class: Technician Attributes:
- technicianID: String (private)
- specialization: String (private)
- skillLevel: String (private)
- availabilityStatus: String (private)
- workZone: String (private)
- teamLeadFlag: Boolean (private)
- averageResolutionTime: Double (private)
- customerSatisfactionRating: Double (private)

Methods:
1. viewAssignedTasks(): List
2. acceptTask(ticketID: String): Boolean
3. declineTask(ticketID: String, reason: String): Boolean
4. updateTaskStatus(ticketID: String, status: String): Boolean
5. uploadMaintenanceEvidence(ticketID: String, images: List, notes: String): Boolean
6. requestAdditionalInfo(ticketID: String): Boolean
7. setAvailabilityStatus(status: String): void

Responsibilities:
- Perform maintenance tasks and update their status
- Upload completion evidence and notes
- Communicate with residents and administrators

4. Administrator Class (Inherits User)

Class: Administrator Attributes:
- adminLevel: String (private)

Methods:
1. viewSystemDashboard(): void
2. assignTicket(ticketID: String, technicianID: String): Boolean
3. reassignTicket(ticketID: String, technicianID: String): Boolean
4. manageUserAccount(userID: String, action: String): Boolean
5. updateTicketPriority(ticketID: String, priority: String): Boolean
6. generateReports(type: String, dateRange: String): File
7. configureSystemRules(): void

Responsibilities:

- Oversee system operations and performance
- Manage users, tickets, and technician assignments
- Analyze system data and generate reports

5. Maintenance Request Class

Class: MaintenanceRequest Attributes:
- ticketID: String (private)
- title: String (private)
- description: String (private)
- category: String (private)
- priority: String (private)
- status: String (private)
- reportedAt: DateTime (private)
- assignedAt: DateTime (private)
- resolvedAt: DateTime (private)
- requesterID: String (private)
- technicianID: String (private)
- locationID: String (private)
- estimatedCompletionTime: DateTime (private)
- isUrgentFlag: Boolean (private)
- similarityFlag: Boolean (private)

Methods:
1. create(): Boolean
2. updateStatus(status: String): Boolean
3. addComment(comment: String): Boolean
4. attachImages(images: List): Boolean

- checkForDuplicates(): Boolean
- calculateEstimatedCompletion(): DateTime

Responsibilities:
- Store and manage maintenance request lifecycle
- Prevent duplicate ticket submissions
- Track status, priority, and escalation

6. Notification Class

Class: Notification Attributes:
- notificationID: String (private)
- userID: String (private)
- relatedTicketID: String (private)
- message: String (private)
- notificationType: String (private)
- isRead: Boolean (private)

- createdAt: DateTime (private)
- expirationDate: DateTime (private)

Methods:
1. markAsRead(): void

- deleteExpired(): void
- send(): Boolean

Responsibilities:
- Notify users of important system events
- Track notification history and read status

7. Location Class

Class: Location Attributes:
- locationID: String (private)
- buildingName: String (private)
- zone: String (private)
- floorNumber: String (private)
- roomNumber: String (private)
- geoCoordinates: String (private)

Methods:
1. getLocationDetails(): String

Responsibilities:
- Represent physical locations for maintenance requests
- Support analytics and technician assignment

Note There are private methods represented with (-) because they encapsulate internal implementation details, reduce system complexity, and are only accessed within the class itself, not directly by external classes.

State Diagrams

1. User authentication and account lifecycle States: Logged Out (Initial), Registering, Email verification Pending, Account active, Entering credentials, CAPTCHA validation, Authenticated, Session active, Session expired, Account locked, Password recovery, Password reset successful.

Transitions:
- Logged Out ⟶ Registering [user clicks Sign Up]
- Registering ⟶ Email Verification Pending [all inputs valid and registration submitted]
- Email Verification Pending ⟶ Account Active [email activation link verified]

- Logged Out → Entering Credentials [user clicks Sign In]
- Entering Credentials → CAPTCHA Validation [3 consecutive failed login attempts]
- CAPTCHA Validation → Authenticated [CAPTCHA valid and credentials correct]
- Entering Credentials → Authenticated [credentials correct and attempts < 3]
- Any State → Account Locked [5 consecutive failed login attempts]
- Account Locked → Entering Credentials [account unlocked by admin or email verification]
- Authenticated → Session Active [successful login and role identified]
- Session Active → Session Expired [inactivity timeout reached]
- Session Active → Logged Out [user logs out]
- Logged Out → Password Recovery [forgot password clicked]
- Password Recovery → Password Reset Successful [verification code valid]
- Password Reset Successful → Logged Out [confirmation sent]

Entry Actions:
- Registering: validate input fields in real-time.
- Email Verification Pending: invoke Notification.send() with activation email.
- Authenticated: load role-specific dashboard.
- Session Active: start session timer.
- Account Locked: disable login and invoke Notification.send() with lock message and unlock instructions.
- Password Recovery: invoke Notification.send() with verification code.

Exit Actions:
- Session Active: clear session tokens.
- Password Recovery: invalidate verification code.

2. Maintenance ticket lifecycle States: Draft (Initial), Submitted, Duplicate check, Open, Escalated, Assigned, In Progress, On Hold, Fixed, Closed.

Transitions:
- Draft → Submitted [resident submits ticket]
- Submitted → Duplicate Check [system checks tickets within last 7 days]
- Duplicate Check → Open [no duplicate detected]
- Open → Escalated [resident marks ticket as Urgent]
- Open → Assigned [admin and system assigns technician]
- Escalated → Assigned [technician with high skill level assigned]
- Assigned → In Progress [technician accepts task]
- Assigned → Open [technician declines task]
- In Progress → On Hold [waiting for parts and availability]
- On Hold → In Progress [blocking issue resolved]
- In Progress → Fixed [technician uploads evidence]
- Fixed → Closed [admin review and resident verification completed]
- Any State → Closed [admin manually closes]

Entry Actions:
- Submitted: generate ticket ID and store timestamp.

- Duplicate Check: compare category, location, and description.
- Escalated: recalculate estimated completion time and invoke Notification.send() to Admin.
- Assigned: invoke Notification.send() to technicianID.
- In Progress: record start time.
- Fixed: validate evidence and invoke Notification.send() to requesterID for verification.
- Closed: archive ticket, update analytics, and invoke Notification.send() to resident.

3. Resident interaction States: Dashboard (Initial), Viewing ticket feed, Creating ticket, Viewing ticket details, Adding comment, Viewing notifications, Managing profile.

Transitions:
- Dashboard → Viewing Ticket Feed [view reports feed]

-Dashboard → Creating Ticket [open maintenance ticket]
- Creating Ticket → Viewing Ticket Details [ticket submitted successfully]
- Viewing Ticket Details → Adding Comment [resident adds comment]
- Adding Comment → Viewing Ticket Details [comment saved]
- Any State → Viewing Notifications [notification icon clicked]
- Any State → Managing Profile [profile selected]

Entry Actions:
- Dashboard: load ticket summary widgets.
- Creating Ticket: validate inputs and preview images.
- Viewing Ticket Details: load status timeline.
- Viewing Notifications: invoke Notification.markAsRead().

4. Technician task handling States: Available (Initial), Task assigned, Task accepted, Task In Progress, Task On Hold, Task Fixed, Task reassigned.

Transitions:
- Available → Task Assigned [new ticket assigned]
- Task Assigned → Task Accepted [technician accepts]
- Task Assigned → Task Reassigned [technician declines]
- Task Accepted → Task In Progress [work started]
- Task In Progress → Task On Hold [issue encountered]
- Task In Progress → Task Fixed [work completed and evidence submitted]
- Task Fixed → Available [task closed]

Entry Actions: -Task Assigned: invoke Notification.send() with ticket details.
- Task In Progress: record start time.
- Task Fixed: invoke Notification.send() to admin and resident for evidence review.

5. Administrator ticket management States: Dashboard (Initial), Reviewing tickets, Assigning ticket, Monitoring progress, Reviewing completion, Generating reports.

Transitions:

- Dashboard → Reviewing Tickets [view pending]
- Reviewing Tickets → Assigning Ticket [assign technician]
- Assigning Ticket → Monitoring Progress [assignment completed]
- Monitoring Progress → Reviewing Completion [ticket marked as fixed]
- Reviewing Completion → Dashboard [ticket approved/closed]
- Dashboard → Generating Reports [export analytics]

Entry Actions:
- Reviewing Tickets: sort tickets by priority/urgency.

-Assigning Ticket: check technician workload and expertise.
- Generating Reports: compile analytics data.

Component Dependencies

1. Authentication and user management component

- Dependencies: Data access layer, Email service, CAPTCHA Service, Password hashing library.
- Provides: User registration, login, logout, role-based authentication, account locking, session management.
- Interfaces: IAuthenticationService, IUserAccountManager, ISessionService.

2. Resident management component

- Dependencies: Data access layer, Authentication component, Notification management Ccmponent.
- Provides: Resident dashboard data, profile management, ticket viewing, ticket tracking.
- Interfaces: IResidentService, IProfileManager, IDashboardProvider.

3. Maintenance ticket management component

- Dependencies: Data access layer, Authentication component, File and evidence storage component, Duplicate detection engine, Notification management component.
- Provides: Maintenance ticket creation, ticket validation, duplicate detection, ticket lifecycle management.
- Interfaces: IMaintenanceTicketService, ITicketRepository, IDuplicateChecker.

4. Technician task management component

- Dependencies: Data access layer, Authentication component, Notification management component, File and evidence storage component.
- Provides: Task assignment handling, task acceptance or rejection, task status updates, maintenance evidence submission.
- Interfaces: ITaskManager, ITaskAssignmentService, IEvidenceService.

5. Administrator management component

- Dependencies: Data access layer, Authentication component, Technician task management component, Maintenance ticket management component, Notification management component.
- Provides: Ticket assignment, priority management, user account management, system monitoring.
- Interfaces: IAdminService, ITicketAssignmentManager, IUserManagementService.

6. Notification management component

- Dependencies: Email service, SMS gateway, In-App notification engine, Queue service (for asynchronous delivery).
- Provides: Notification delivery, notification preferences handling, notification history tracking.
- Interfaces: INotificationService, IMessageDispatcher, INotificationRepository.

7. Analytics and reporting component

- Dependencies: Data access layer, Chart libraries, Export services (PDF, Excel), Maintenance ticket management component, Technician task management component.
- Provides: Maintenance analytics, technician performance analysis, trend visualization, report generation.
- Interfaces: IAnalyticsEngine, IReportGenerator, IDataAggregator.

8. File and evidence storage component

- Dependencies: Cloud Storage API (e.g. AWS S3, Azure Blob), Security Access Control.
- Provides: Image upload, document storage, secure file access, file metadata management.
- Interfaces: IFileStorageService, IEvidenceRepository.

9. System configuration and rules engine component

- Dependencies: Data access layer.
- Provides: Priority rules, assignment automation policies (Auto-assign), notification thresholds, system configuration settings.
- Interfaces: IConfigurationService, IRuleEngine.

Component Design Diagram

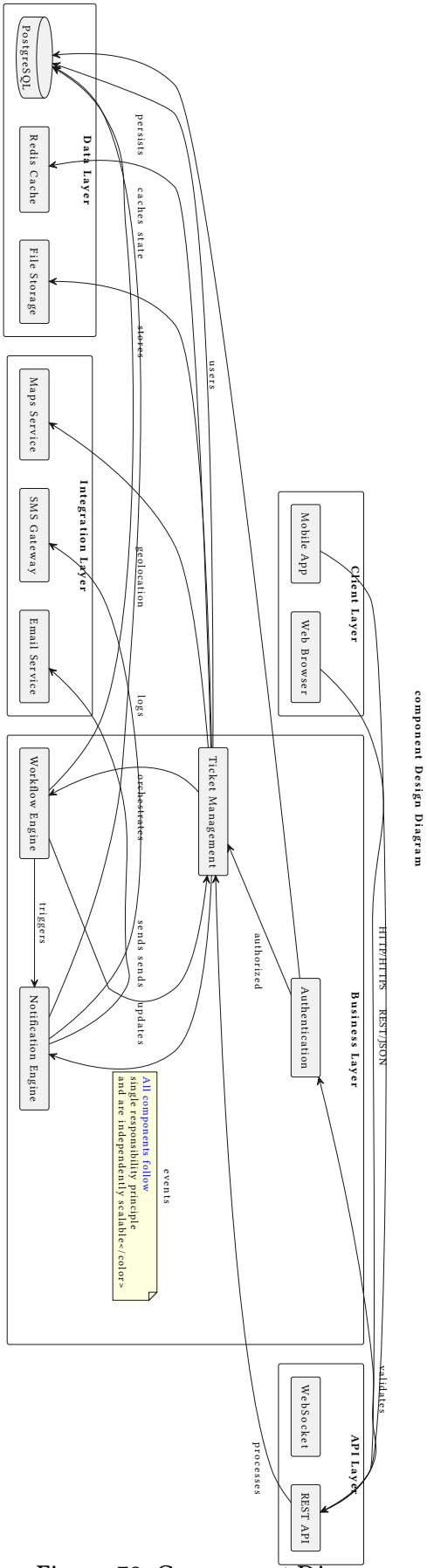


Figure 78: Component Diagram

4.6. ERD analysis and Database Design

4.6.1. Database Design (ERD)

Figure 79: Entity Relation Diagram

4.6.2. Object to ER Mapping

Table 101: Object to ER Mapping – User

User			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
User	users	user_id → user_id (PK) fullName → full_name email → email (UNIQUE) passwordHash → password_hash nationalID → national_id (UNIQUE) isEmailVerified → is_email_verified role → role registrationDate → registration_date accountStatus → account_status lastLoginAt → last_login_at accountExpirationDate → account_expiration_date failedLoginAttempts → failed_login_attempts profilePicture → profile_picture_url phoneNumber → phone_number createdAt → created_at updatedAt → updated_at	One-to-Many with MaintenanceRequest One-to-Many with Notification One-to-Many with Comment One-to-One with Technician

Table 102: Object to ER Mapping – Technician

Technician			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
Technician	technicians	technician_id → technician_id (PK) userID → user_id (FK) skills → skills availabilityStatus → availability_status skillLevel → skill_level certifications → certifications specialization → specialization workZone → work_zone workSchedules → work_schedules teamLeadFlag → team_lead_flag averageResolutionTime → average_resolution_time_minutes customerSatisfactionRating → customer_satisfaction_rating createdAt → created_at updatedAt → updated_at	One-to-One with User One-to-Many with MaintenanceRequest One-to-Many with MaintenanceSchedule One-to-Many with MaintenanceEvidence

Table 103: Object to ER Mapping – Maintenance Request

Maintenance Request			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
MaintenanceRequest	maintenance_requests	ticketID → ticket_id (PK) title → title description → description category → category requesterID → requester_id (FK) technicianID → technician_id (FK) locationID → location_id (FK) reportedAt → reported_at assignedAt → assigned_at resolvedAt → resolved_at priority → priority status → status estimatedCompletionTime → estimated_completion_time isUrgentFlag → is_urgent_flag similarityFlag → similarity_flag escalationLevel → escalation_level onHoldReason → on_hold_reason residentEditable → resident_editable createdAt → created_at updatedAt → updated_at	Many-to-One with User Many-to-One with Technician Many-to-One with Location One-to-Many with TicketImages One-to-Many with TicketComments One-to-Many with MaintenanceEvidence One-to-Many with Notifications One-to-Many with MaintenanceSchedule

Table 104: Object to ER Mapping – Location

Location			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
Location	locations	locationID → location_id (PK) zone → zone buildingName → building_name locationCriticality → location_criticality geoCoordinates → geo_coordinates roomNumber → room_number floorNumber → floor_number details → details createdAt → created_at updatedAt → updated_at	One-to-Many with MaintenanceRequest

Table 105: Object to ER Mapping – Maintenance Schedule

Maintenance Schedule			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
MaintenanceSchedule	maintenance_schedules	scheduleID → schedule_id (PK) technicianID → technician_id (FK) linkedRequestID → linked_request_id (FK) expectedDuration → expected_duration_minutes scheduledStartTime → scheduled_start_time recurrencePattern → recurrence_pattern scheduleStatus → schedule_status notes → notes createdAt → created_at updatedAt → updated_at	Many-to-One with Technician Many-to-One with MaintenanceRequest

Table 106: Object to ER Mapping – Notification

Notification			
Class (Object) Name	**Table Name**	**Attributes (Fields → Columns)**	**Relationships**
Notification	notifications	notificationID → notification_id (PK) userID → user_id (FK) relatedTicketID → related_ticket_id (FK) notificationType → notification_type message → message isRead → is_read deliveryMethod → delivery_method priorityLevel → priority_level expirationDate → expiration_date createdAt → created_at	Many-to-One with User Many-to-One with MaintenanceRequest