

SMART MAINTENANCE

Salleha

Systems Analysis and Design
First Semester
2025/2026

University of Jordan

Systems Analysis and Design Project



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 of the software documentation. Added Chapter 1 (Project Initiation) and Chapter 2 (Project Management Plan).
Version 2.0	Updated version of the software documentation. Added Chapter 3 (Software Requirements Specifications).
Version 3.0	Updated version of the software documentation. Added Chapter 4 (System Analysis and Design), Chapter 5 (User Manual), and Chapter 6 (References), excluding implementation details.

Executive Summary

Contents

1. Project initiation	1
1.1. Project Overview	1
1.2. Problem Definition	1
1.2.1. Issues	1
1.2.2. Objectives	1
1.2.3. Requirements	1
1.2.4. Constraints	2
1.2.5. Vision Document	2
1.2.5.1. Problem Description	2
1.2.5.2. System Capabilities	2
1.2.5.3. Business Benefits	3
1.3. Feasibility Studies	3
1.3.1. Technical Feasibility	3
1.3.2. Operational Feasibility	4
1.3.3. Economic Feasibility	5
1.3.4. Schedule Feasibility	6
1.3.5. Legal Feasibility	6
1.4. Recommended Solution and Expected Project Deliverables	7
1.5. Local and Global Impact of the Proposed Solution	8
2. Project Management plan	9
2.1. Project Organization	9
2.2. Roles and Responsibilities	9
2.3. Software Process Model	11
2.3.1. Main Phases :	11
2.4. Project Environment	13
2.4.1. Procedures	13
2.4.2. Tools	13
2.4.3. Hardware (HW) Resources:	13
2.4.4. Software (SW) Resources:	14
2.5. Project tasks	15
2.6. Project Schedule	25
2.6.1. Activity Network	25
2.6.2. Gantt Chart	26
2.7. Assigning Team Members to Tasks	27
2.8. Monitoring and Controlling Mechanisms	28
2.9. Risk Analysis	31
2.9.1. Effects and Causes	31
2.10. Communication Plan	31
3. Software Requirements Specifications	31
3.1. System Stakeholders and Requirement Sources	31
3.1.1. System Stakeholders	31
3.1.2. Information Gathering	32
3.1.2.1. Interviews	33

Systems Analysis and Design Project

3.1.2.2. Questionnaires	35
3.1.2.3. Document Analysis	45
3.1.2.4. Observation	46
3.1.2.5. Prototype	46
3.2. User Requirements	53
3.3. Context Diagram	56
3.4. Use Case Model	57
3.4.1. Use Case diagram	57
3.4.2. Use Case descriptions	59
3.5. Functional Requirements Specification	60
3.6. Data Requirements	72
3.7. Non-Functional Requirements	77
3.8. Requirements Validation and Review Summary	77
3.8.1. How We Verified the Requirements	77
3.8.1.1. Team Review	77
3.8.1.2. Stakeholder Review	78
3.8.2. How We Confirmed the Requirements Are Correct, Clear, and Complete ..	78
3.8.2.1. Method 1: Mockups	78
3.8.2.2. Method 2: Walkthroughs	78
4. System Analysis and Design	79
4.1. DFDs	79
4.2. Data Dictionaries	81
4.2.1. Data flow	81
4.3. Process Specifications	97
4.3.1. Fragment 1: User Management Processes	97
4.3.2. Fragment 2: Maintenance Request Management Processes	109
4.3.3. Fragment 3: Assignment & Task Management Processes	132
4.3.4. Fragment 4: Analytics & Reporting Processes	149
4.3.5. Fragment 5: Notification Management Processes	169
4.3.6. Summary of All Process Specifications	182
4.4. Object-Oriented Analysis	184
4.4.1. Activity Diagrams	184
4.5. System Architecture Design	186
4.5.1. Architectural Style	186
4.5.1.1. Primary Architecture Pattern: Three-Tier Client-Server Architecture	186
4.5.1.2. Architectural Characteristics	186
4.5.2. Technology Stack	186
4.5.2.1. Frontend Technologies	186
4.5.2.1.1. Web Application	186
4.5.2.1.2. Mobile Application	186
4.5.2.2. Backend Technologies	187
4.5.2.2.1. API Server	187
4.5.2.2.2. Database Layer	187

Systems Analysis and Design Project

4.5.2.3. Infrastructure & DevOps	187
4.5.2.4. Development & Testing Tools	187
4.5.3. Component/Module-Level View	188
4.5.3.1. System Components Diagram	188
4.5.3.2. Core Modules and Components	188
4.5.3.2.1. MODULE 1: Authentication & Authorization Module	188
4.5.3.2.2. MODULE 2: User Management Module	189
4.5.3.2.3. MODULE 3: Maintenance Request Management Module	189
4.5.3.2.4. MODULE 4: Task Assignment & Scheduling Module .	190
4.5.3.2.5. MODULE 5: Technician Workflow Module	190
4.5.3.2.6. MODULE 6: Notification & Communication Module .	190
4.5.3.2.7. MODULE 7: Analytics & Reporting Module	191
4.5.3.2.8. MODULE 8: Location & Equipment Management Module	191
4.5.3.2.9. MODULE 9: Data Access Layer (DAL)	192
4.5.4. Module Responsibilities Summary Table	193
4.5.5. Architectural Design Decisions	193
4.5.5.1. Why Three-Tier Architecture?	193
4.5.5.2. Why RESTful API?	193
4.5.5.3. Why Node.js + Java Hybrid Backend?	194
4.5.5.4. Why MySQL?	194
4.5.5.5. Why Redis for Sessions?	194
4.5.6. Security Architecture	194
4.5.6.1. Security Measures	194
4.5.6.2. Data Privacy	194
4.5.7. Performance Optimization Strategies	194
4.5.8. Deployment Architecture	195
4.5.8.1. Production Environment	195
4.5.8.2. Continuous Integration/Deployment	195
4.5.9. Monitoring & Observability	196
4.5.10. Scalability Considerations	196
4.5.11. Alignment with Non-Functional Requirements	196
4.5.12. Future Enhancements	196
4.6. Object-Oriented Design	197
4.6.1. Class Diagram	197
4.6.2. Sequence Diagrams	198
4.6.3. Classes and Components Design	212
4.7. ERD analysis and Database Design	222
4.7.1. Database Design (ERD)	222
4.7.2. Object to ER Mapping	223
5. User Manual	229
6. References	251

Systems Analysis and Design Project

Tables

Table 1	Development Costs	5
Table 2	Operational Costs	5
Table 3	Intangible Benefits	5
Table 4	Benefit and Payback Analysis	6
Table 5	Project Development Schedule	6
Table 6	Team Roles Assignments and Responsibilities	9
Table 7	Roles and Responsibilities	9
Table 8	Tools	13
Table 9	Hardware (HW) Resources	13
Table 10	Software (SW) Resources	14
Table 11	Project Tasks	15
Table 12	Task-to-Team Member Assignment	27
Table 13	Earned Value Management Progress Tracking	28
Table 14	Time and Cost Options	29
Table 15	Schedule Expediting Table	30
Table 16	Operational Stakeholders Interviews	33
Table 17	Internal Stakeholders Interviews	33
Table 18	Executive Stakeholder Interviews	34
Table 19	External Stakeholder Interview	34
Table 20	High level Functional Requirements	53
Table 21	Use Cases Descriptions	59
Table 22	FR-1 User Registration	60
Table 23	FR-2 User Login	61
Table 24	FR-3 Logout	61
Table 25	FR-4 Password Recovery	61
Table 26	FR-5 Resident Dashboard	62
Table 27	FR-6 Resident Navigation Menu	62
Table 28	FR-7 Maintenance Issue Feed	63
Table 29	FR-8 Open Maintenance Ticket	63
Table 30	FR-9 Ticket Status Tracking	64
Table 31	FR-10 Status Change Notifications	64
Table 32	FR-11 Duplicate Ticket Prevention	64
Table 33	FR-12 Resident Profile Management	65
Table 34	FR-13 Technician Dashboard	65
Table 35	FR-14 Technician Navigation Menu	66
Table 36	FR-15 Maintenance Request Management	66
Table 37	FR-16 Task Status Updates	66
Table 38	FR-17 Maintenance Evidence Submission	67
Table 39	FR-18 Technician Notifications	67
Table 40	FR-19 Maintenance History Access	68
Table 41	FR-20 Basic Analytics for Technicians	68
Table 42	FR-21 Administrator Dashboard	68
Table 43	FR-22 Administrator Navigation Menu	69

Systems Analysis and Design Project

Table 44	FR-23 Ticket Assignment Management	69
Table 45	FR-24 Ticket Priority Management	70
Table 46	FR-25 Administrator Notifications	70
Table 47	FR-26 Analytics and Reporting	71
Table 48	FR-27 Report Export Functionality	71
Table 49	FR-28 User Account Management	72
Table 50	Data Requirements	72
Table 51	Non-Functional Requirements	77
Table 52	Data Flow 1 Details	81
Table 53	Data Flow 2 Details	82
Table 54	Data Flow 3 Details	82
Table 55	Data Flow 4 Details	83
Table 56	Data Flow 5 Details	83
Table 57	Data Flow 6 Details	83
Table 58	Data Flow 7 Details	84
Table 59	Data Flow 8 Details	84
Table 60	Data Flow 9 Details	84
Table 61	Data Flow 10 Details	85
Table 62	Data Flow 11 Details	85
Table 63	Data Flow 12 Details	85
Table 64	Data Flow 13 Details	86
Table 65	Data Flow 14 Details	86
Table 66	Data Flow 15 Details	86
Table 67	Data Flow 16 Details	87
Table 68	Data Flow 17 Details	87
Table 69	Data Flow 18 Details	87
Table 70	Data Flow 19 Details	88
Table 71	Data Flow 20 Details	88
Table 72	Data Flow 21 Details	88
Table 73	Data Flow 22 Details	89
Table 74	Data Flow 23 Details	89
Table 75	Data Flow 24 Details	89
Table 76	Data Flow 25 Details	90
Table 77	Data Flow 26 Details	90
Table 78	Data Flow 27 Details	90
Table 79	Data Flow 28 Details	91
Table 80	Data Flow 29 Details	91
Table 81	Data Flow 30 Details	91
Table 82	Data Flow 31 Details	92
Table 83	Data Flow 32 Details	92
Table 84	Data Flow 33 Details	92
Table 85	Data Flow 34 Details	93
Table 86	Data Flow 35 Details	93
Table 87	Data Flow 36 Details	93

Systems Analysis and Design Project

Table 88	Data Flow 37 Details	94
Table 89	Data Flow 38 Details	94
Table 90	Data Flow 39 Details	94
Table 91	Data Flow 40 Details	95
Table 92	Data Flow 41 Details	95
Table 93	Data Flow 42 Details	95
Table 94	Data Flow 43 Details	96
Table 95	Data Flow 44 Details	96
Table 96	Data Flow 45 Details	96
Table 97	Data Flow 46 Details	97
Table 98	Data Flow 47 Details	97
Table 99	99
Table 100	106
Table 101	109
Table 102	112
Table 103	123
Table 104	128
Table 105	132
Table 106	141
Table 107	149
Table 108	152
Table 109	168
Table 110	171
Table 111	176
Table 112	182
Table 113	193
Table 114	196
Table 115	Object to ER Mapping – User	223
Table 116	Object to ER Mapping – Technician	224
Table 117	Object to ER Mapping – Maintenance Request	225
Table 118	Object to ER Mapping – Location	226
Table 119	Object to ER Mapping – Maintenance Schedule	227
Table 120	Object to ER Mapping – Notification	228

Figures

Figure 1	Project Organizational Structure	9
Figure 2	Waterfall Software Process Model	11
Figure 3	Activity Network Diagram	25
Figure 4	Gantt Chart	26
Figure 5	Fishbone Diagram	31
Figure 6	Admin Questionnaire (Nominal)	35
Figure 7	Admin Questionnaire (Ordinal)	36
Figure 8	Admin Questionnaire (Interval)	37
Figure 9	Admin Questionnaire (Ratio)	37
Figure 10	Admin Questionnaire (Open-ended)	38
Figure 11	Resident Questionnaire (Nominal)	39
Figure 12	Resident Questionnaire (Ordinal)	40
Figure 13	Resident Questionnaire (Interval and Ratio)	41
Figure 14	Resident Questionnaire (Open-ended)	42
Figure 15	Technician Questionnaire (Nominal)	43
Figure 16	Technician Questionnaire (Ordinal)	44
Figure 17	Technician Questionnaire (Interval)	45
Figure 18	Technician Questionnaire (Ratio and Open-ended)	45
Figure 19	Login Screen	46
Figure 20	Registration Screen	47
Figure 21	Notifications Screen	47
Figure 22	Resident – Dashboard	48
Figure 23	Resident – Ticket Request	48
Figure 24	Resident – Tickets Screen	49
Figure 25	Technician – Dashboard	49
Figure 26	Technician – Maintenance History	49
Figure 27	Technician – Ticket Details	50
Figure 28	Administrator – Dashboard	51
Figure 29	Administrator – All Tickets	51
Figure 30	Administrator – Assignment Management	51
Figure 31	Administrator – Analytics & Reports	52
Figure 32	Administrator – Notifications	52
Figure 33	Administrator – Priority Management	53
Figure 34	Administrator – User Accounts Management	53
Figure 35	Context Diagram	56
Figure 36	Use Case Diagram - Resident	57
Figure 37	Use Case Diagram - Technician	57
Figure 38	Use Case Diagram - Administrator	58
Figure 39	DFD Level 0	79
Figure 40	DFD Level 1-Fragment 1	79
Figure 41	DFD Level 1-Fragment 2	80
Figure 42	DFD Level 1-Fragment 3	80
Figure 43	DFD Level 1-Fragment 4	81

Systems Analysis and Design Project

Figure 44 DFD Level 1-Fragment 5	81
Figure 45	102
Figure 46 Decision Tree 1.2	103
Figure 47	116
Figure 48 Decision Tree 2.3	117
Figure 49	135
Figure 50 Decision Tree 3.2	136
Figure 51	144
Figure 52 Decision Tree 3.3	145
Figure 53 Decision Tree 4.3	162
Figure 54 Resident Activity Diagram	184
Figure 55 Technician Activity Diagram	185
Figure 56 System Components Diagram	188
Figure 57 Deployment Diagram	195
Figure 58 Class Diagram	197
Figure 59 FR-1 Sequence Diagram	198
Figure 60 FR-2 Sequence Diagram	198
Figure 61 FR-3 Sequence Diagram	199
Figure 62 FR-4 Sequence Diagram	200
Figure 63 FR-5 Sequence Diagram	200
Figure 64 FR-6 Sequence Diagram	201
Figure 65 FR-7 Sequence Diagram	201
Figure 66 FR-8 Sequence Diagram	202
Figure 67 FR-9 Sequence Diagram	202
Figure 68 FR-10 Sequence Diagram	203
Figure 69 FR-11 Sequence Diagram	203
Figure 70 FR-12 Sequence Diagram	204
Figure 71 FR-13 Sequence Diagram	204
Figure 72 FR-14 Sequence Diagram	205
Figure 73 FR-15 Sequence Diagram	206
Figure 74 FR-16 Sequence Diagram	206
Figure 75 FR-17 Sequence Diagram	207
Figure 76 FR-18 Sequence Diagram	207
Figure 77 FR-19 Sequence Diagram	208
Figure 78 FR-20 Sequence Diagram	208
Figure 79 FR-21 Sequence Diagram	209
Figure 80 FR-22 Sequence Diagram	209
Figure 81 FR-23 Sequence Diagram	210
Figure 82 FR-24 Sequence Diagram	210
Figure 83 FR-25 Sequence Diagram	211
Figure 84 FR-26 Sequence Diagram	211
Figure 85 FR-27 Sequence Diagram	212
Figure 86 FR-28 Sequence Diagram	212
Figure 87 Component Diagram	221

Systems Analysis and Design Project

Figure 88 Entity Relation Diagram 222

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.

Systems Analysis and Design Project

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 fulfill their needs. Current methods are almost obsolete they are inefficient and insufficient because the older methods have lack of transparency, increased delays, and unavailable hence the overall user frustration, not to mention the difficulty of managing the reports.

Without a system to hold everything together it requires 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.

Systems Analysis and Design Project

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. Feasility 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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

1.3.3. Economic Feasibility

Development Costs:

Table 1: Development 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

Systems Analysis and Design Project

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+					

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

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

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.

Systems Analysis and Design Project

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:

Systems Analysis and Design Project

- **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.

Systems Analysis and Design Project

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.

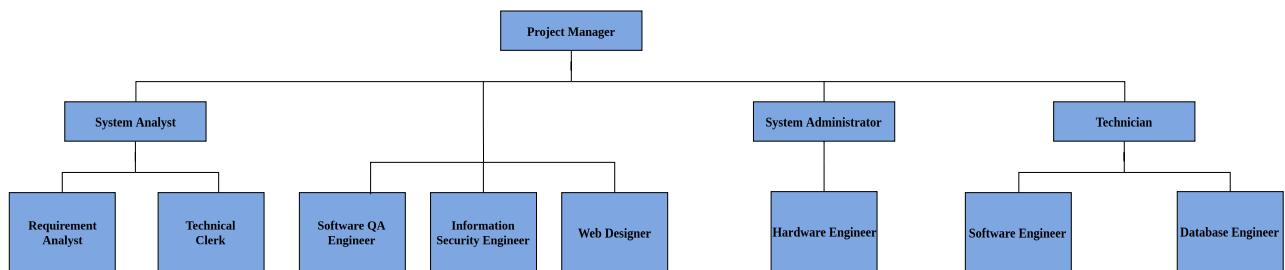


Figure 1: Project Organizational Structure

Table 6: Team Roles Assignments and Responsibilities

Assigned Member	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.

Systems Analysis and Design Project

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 technical records. Supports technical team with admin tasks.
Software QA Engineer	Ensuring that software products meet the highest standards of quality and functionality.
Information Security Engineer	Responsible 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 re-

trieval. 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.

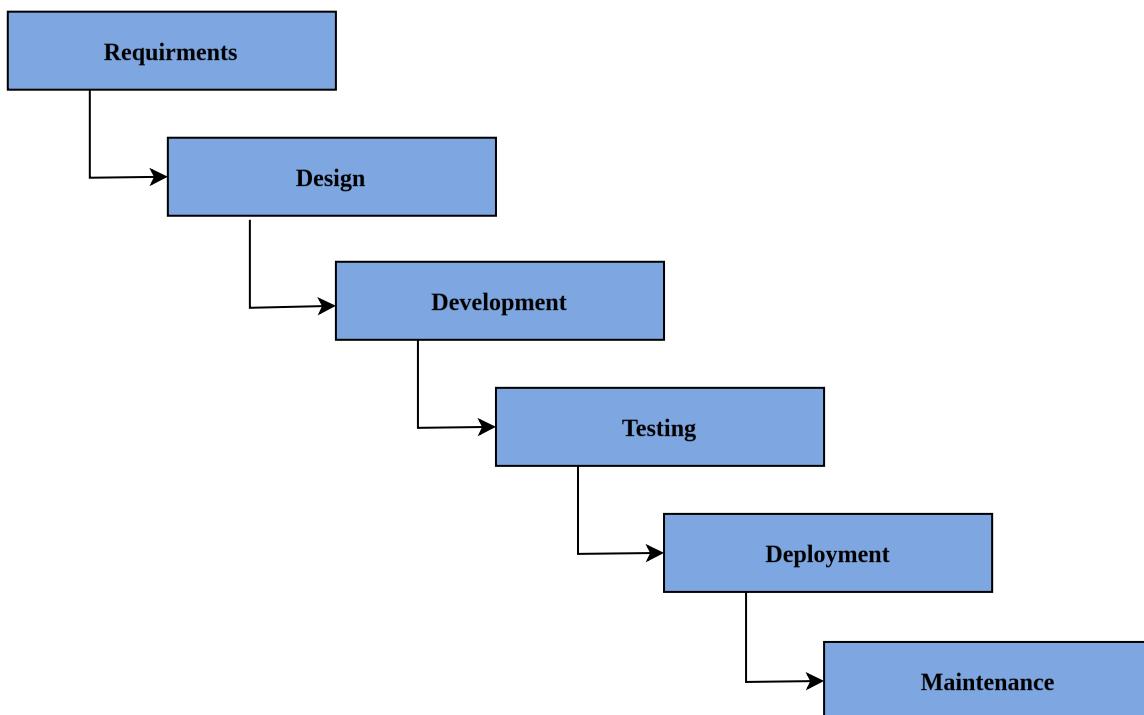


Figure 2: Waterfall Software Process Model

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.

Systems Analysis and Design Project

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 code-base.
Documentation	Typst for creating our PDF documentation.
Version Control	GitHub for sharing and tracking project progress and managing team collaboration.
Design & Prototyping	Figma for creating 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

Systems Analysis and Design Project

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	Estimated Time
<ul style="list-style-type: none"> • Resource & Schedule Planning (T1) • Requirements Gathering (T2) 	<ul style="list-style-type: none"> • Develop Work Breakdown Structure for web and mobile app development • Create risk register and define quality standards for the application 	<ul style="list-style-type: none"> • 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
	<ul style="list-style-type: none"> • N/A 	<ul style="list-style-type: none"> • Dependencies <ul style="list-style-type: none"> ▸ Approved project charter ▸ Availability of key stakeholders for interviews such as the requesters and the facility managers • Constraints <ul style="list-style-type: none"> ▸ Budget constraints 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ▸ Work breakdown structure diagram ▸ Table with risk, impact, probability, priority, and mitigation ▸ Quality standards document (acceptance criteria checklist) • Milestones <ul style="list-style-type: none"> ▸ M1.1: Charter Approved and communicated to the team. ▸ M1.2: Work Breakdown Structure completed and reviewed by all members.

Systems Analysis and Design Project

			<ul style="list-style-type: none"> ▸ M1.3: Resource plan and schedule baseline approved. ▸ M1.4: Risk register and quality standards finalized.
Phase	Task	Detailed task	Estimated Time
Analysis	<ul style="list-style-type: none"> • Requirements Elicitation (T3) • Requirements Modeling & Documentation (T4) • Requirements Validation (T5) 	<ul style="list-style-type: none"> • 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 	2 weeks

Systems Analysis and Design Project

		<ul style="list-style-type: none"> • Obtain formal sign-off on SRS 	
	Resources Needed	Dependencies and Constraints	Deliverables & Milestones
	<ul style="list-style-type: none"> • N/A 	<ul style="list-style-type: none"> • Dependencies <ul style="list-style-type: none"> ‣ Access to collaboration tools so the informations can be shared and reviewed. • Constraints <ul style="list-style-type: none"> ‣ Stakeholder time; limited availability may restrict depth of interviews ‣ Data privacy when handling the facility data 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ‣ 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 <ul style="list-style-type: none"> ‣ M2.1: Software Requirements Specification (SRS) reviewed and approved by stakeholders. ‣ M2.2: Requirements validation and traceability matrix completed.
Phase	Task	Detailed task	Estimated Time

Systems Analysis and Design Project

Design	<ul style="list-style-type: none"> • Architectural Design (T6) • High-Level (Logical) Design (T7) • Detailed (Physical) Design (T8) • Design Review & Approval (T9) 	<ul style="list-style-type: none"> • Choose overall system architecture for web and mobile • Define network topology and cloud infrastructure • Break system into modules (frontend, backend, database) • 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 	2 weeks
Resources Needed	Dependencies and Constraints	Deliverables & Milestones	
<ul style="list-style-type: none"> • Information Security Engineer 	<ul style="list-style-type: none"> • Dependencies <ul style="list-style-type: none"> ▸ Completion and approval of Software Requirements Specification 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ▸ Web and Mobile System Architecture Document 	

Systems Analysis and Design Project

		<ul style="list-style-type: none"> ▸ UI design depends on confirmed user workflows and functional requirements because they show what the user needs to do and how the system should respond • Constraints <ul style="list-style-type: none"> ▸ Technology constraints ▸ Design must comply with quality standards and security requirements 	<ul style="list-style-type: none"> ▸ High-Level Design including system modules and interaction diagrams ▸ Detailed Design including Class diagrams, Sequence diagrams, Database schema and API documentation ▸ UI/UX prototypes for web and mobile • Milestones <ul style="list-style-type: none"> ▸ 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	Estimated Time
Development	<ul style="list-style-type: none"> • Development Setup (T10) • Front-end Code (T11) • Back-end Code (T12) • Database Physical Design (T13) 	<ul style="list-style-type: none"> • Configure Git repository and version control • Set up web development environment • Set up Flutter development 	4 weeks

Systems Analysis and Design Project

		<p>environment with Android/iOS SDKs</p> <ul style="list-style-type: none"> • Initialize backend framework and dependencies • Configure linters, formatters, and testing frameworks • Implement web frontend components for maintenance 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
	<ul style="list-style-type: none"> • N/A 	<ul style="list-style-type: none"> • Dependencies 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ▸ Git repository

Systems Analysis and Design Project

		<ul style="list-style-type: none"> ▸ Depends on approved design phase deliverables such as architecture, and database schema • Constraints <ul style="list-style-type: none"> ▸ Follow the coding standards, security rules, and framework versions that were set in the earlier phases ▸ Internet or cloud service interruptions may slow down integration and testing activities 	<ul style="list-style-type: none"> ▸ Working web and mobile front-end modules (React and Flutter) ▸ Implemented database schema ▸ Unit testing and integration tests • Milestones <ul style="list-style-type: none"> ▸ M4.1: Development environment configured and 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	Estimated Time
Testing	<ul style="list-style-type: none"> • Test Planning (T14) • Integration Testing (T15) 	<ul style="list-style-type: none"> • Develop comprehensive test plan for web and mobile platforms • Define test environments 	2 weeks

Systems Analysis and Design Project

	<ul style="list-style-type: none"> • System & Acceptance Testing (T16) • Regression & Release Testing (T17) 	<p>(development, staging, production)</p> <ul style="list-style-type: none"> • 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 • 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	
	<ul style="list-style-type: none"> • External QA engineers 	<ul style="list-style-type: none"> • Dependencies <ul style="list-style-type: none"> ▸ Test data and environment setup depends on finalized database 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ▸ Completed integration and system test reports

Systems Analysis and Design Project

		<ul style="list-style-type: none"> ▶ Test plan and test cases finalized • Constraints <ul style="list-style-type: none"> ▶ 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 • Milestones <ul style="list-style-type: none"> ▶ 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	Estimated Time
Deployment	<ul style="list-style-type: none"> • Release Planning (T18) • Environment Provisioning (T19) • Go-Live Execution (T20) • Transition & Support (T21) • Post-Deployment Review (T22) 	<ul style="list-style-type: none"> • 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 	1 weeks

Systems Analysis and Design Project

		<p>automated deployment</p> <ul style="list-style-type: none"> • 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 • Monitor system performance and user feedback • Review deployment metrics and KPIs • Document lessons learned and improvement areas 	
	Resources Needed	Dependencies and Constraints	Deliverables & Milestones
	<ul style="list-style-type: none"> • N/A 	<ul style="list-style-type: none"> • Dependencies <ul style="list-style-type: none"> ▸ Dependent on successful completion of testing • Constraints <ul style="list-style-type: none"> ▸ It must be scheduled 	<ul style="list-style-type: none"> • Deliverables <ul style="list-style-type: none"> ▸ Successfully deployed web and mobile applications ▸ User training and support materials completed

Systems Analysis and Design Project

		<ul style="list-style-type: none"> ▶ Risk of configuration errors ▶ Security and data compliance must be checked and confirmed before the system goes live 	<ul style="list-style-type: none"> ▶ Documented post-deployment review and lessons learned <ul style="list-style-type: none"> • 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

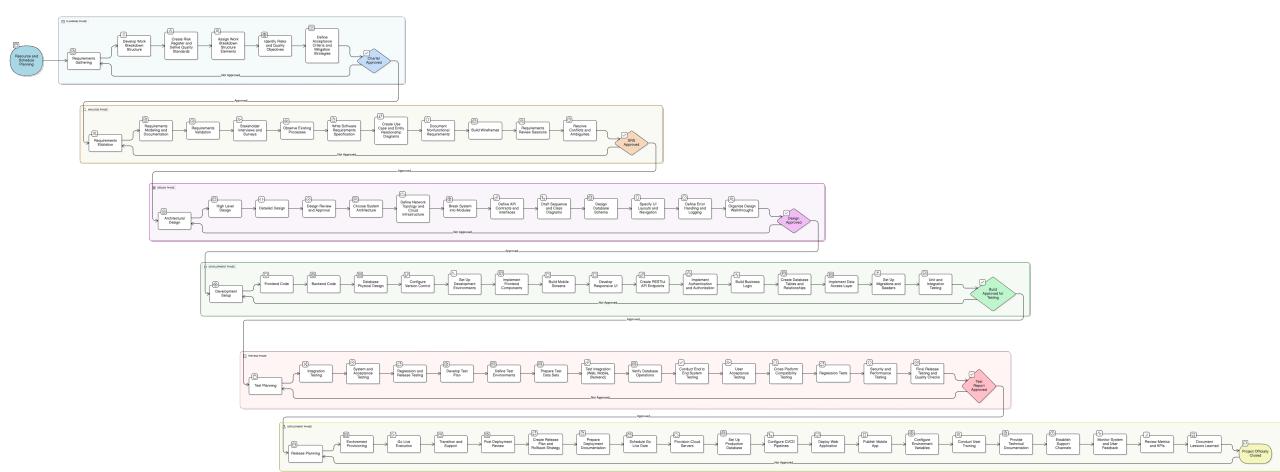


Figure 3: Activity Network Diagram

Systems Analysis and Design Project

2.6.2. Gantt Chart

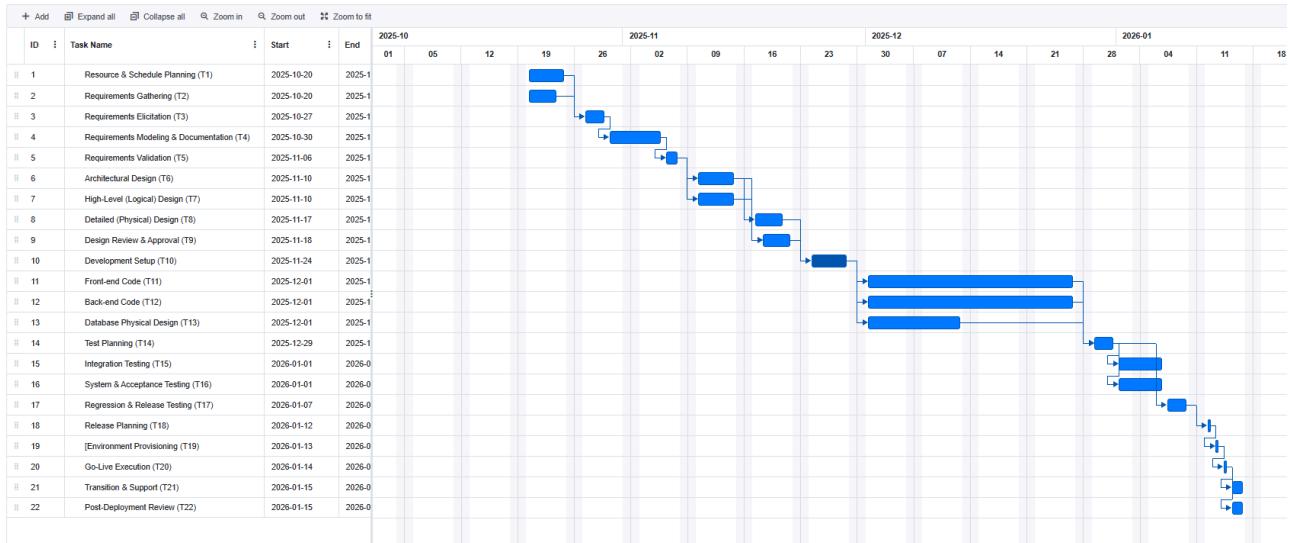


Figure 4: Gantt Chart

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
Implementation	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**

Systems Analysis and Design Project

- 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

Systems Analysis and Design Project

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

Systems Analysis and Design Project

2.9. Risk Analysis

2.9.1. Effects and Causes

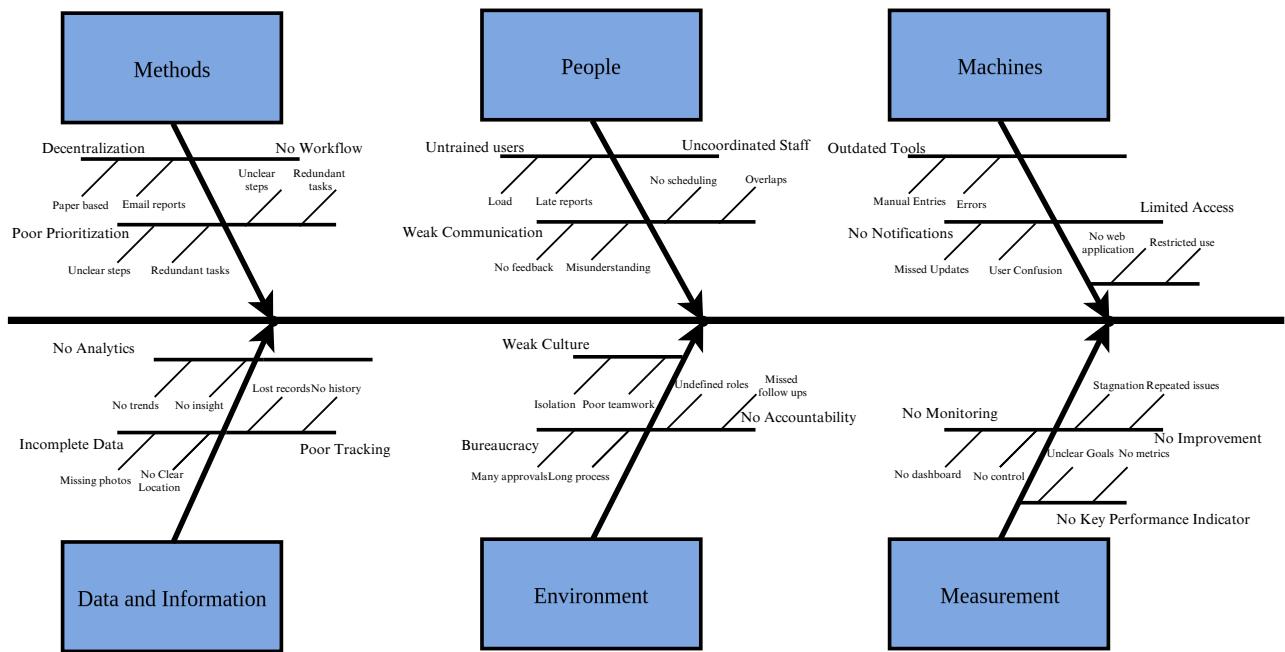


Figure 5: Fishbone Diagram

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.
 - Individuals who report maintenance issues through the application.

Systems Analysis and Design Project

- Their main concerns are ease of use, privacy and clear updates with minimum delay.
- Techinicians
 - Individials 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
 - Individials who oversee the whole maintenance workflow.
 - Their main concerns are efficiency, resource optimization and centralization.
- Project Manager
 - Responsible for overall planning and coordination and project execution.
 - Main concerns are great communication, maintaining quality standards and ensuring project completion.
- System Administrators
 - Individials 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. Executive 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 understood and defined, while also having the culture in mind, we will perform a set of techniques in our search for information.

Systems Analysis and Design Project

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	
Question	Question Type
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

Systems Analysis and Design Project

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

Systems Analysis and Design Project

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)

Systems Analysis and Design Project

B. Current Process (Ordinal)

3. How effective is the current maintenance management process? *

- Very ineffective
- Ineffective
- Neutral
- Effective
- Very effective

4. How easy is it to monitor technician performance? *

- Very difficult
- Difficult
- Neutral
- Easy
- Very easy

Figure 7: Admin Questionnaire (Ordinal)

Systems Analysis and Design Project

C. Strategic Evaluation (Likert Scale 1–5)

5. Rate your agreement: "Analytics dashboards are essential for decision-making." *

1 2 3 4 5

Strongly disagree Strongly agree

6. Rate your agreement: "Automated ticket assignment would improve efficiency." *

1 2 3 4 5

Strongly disagree Strongly agree

7. Rate your agreement: "Maintenance data helps identify recurring problems." *

1 2 3 4 5

Strongly disagree Strongly agree

Figure 8: Admin Questionnaire (Interval)

D. Quantitative Insight (Ratio)

8. Approximately how many maintenance requests are handled per month? *

Your answer _____

9. What is the average response time (in hours) for maintenance requests? *

Enter number of hours _____

Your answer _____

Figure 9: Admin Questionnaire (Ratio)

Systems Analysis and Design Project

E. Open Feedback

10. What reports or analytics would be most useful for you?

Your answer

Figure 10: Admin Questionnaire (Open-ended)

Systems Analysis and Design Project

• 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)

Systems Analysis and Design Project

B. Usage & Experience (Ordinal)

3. How often do you face maintenance issues? *

- Very rarely
- Rarely
- Sometimes
- Often
- Very often

4. How easy is it to report a maintenance issue using current methods? *

- Very difficult
- Difficult
- Neutral
- Easy
- Very easy

5. How satisfied are you with the response time of maintenance services? *

- Very dissatisfied
- Dissatisfied
- Neutral
- Satisfied
- Very satisfied

Figure 12: Resident Questionnaire (Ordinal)

Systems Analysis and Design Project

C. Perception & Evaluation (Likert Scale 1–5)

6. Rate your agreement: "I want real-time updates on my maintenance requests." *



7. Rate your agreement: "Privacy and anonymity are important when reporting issues." *



D. Quantitative Insight (Ratio)

8. On average, how many maintenance issues do you report per year? *

Your answer

9. How many days does it usually take for an issue to be resolved? *

Enter number of days

Your answer

Figure 13: Resident Questionnaire (Interval and Ratio)

Systems Analysis and Design Project

D. Quantitative Insight (Ratio)

8. On average, how many maintenance issues do you report per year? *

Your answer

9. How many days does it usually take for an issue to be resolved? *

Enter number of days

Your answer

E. Open Feedback

10. What features would you most like to see in a maintenance request system?

Your answer

Figure 14: Resident Questionnaire (Open-ended)

Systems Analysis and Design Project

- **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)

Systems Analysis and Design Project

B. Workload & Process (Ordinal)

3. How clear are the maintenance requests you receive? *

- Very unclear
- Unclear
- Neutral
- Clear
- Very clear

4. How well are tasks currently prioritized? *

- Very poorly
- Poorly
- Acceptable
- Well
- Very well

5. How easy is it to track the status of assigned tasks? *

- Very difficult
- Difficult
- Neutral
- Easy
- Very easy

Figure 16: Technician Questionnaire (Ordinal)

Systems Analysis and Design Project

C. System Evaluation (Likert Scale 1–5)

6. Rate your agreement: "A centralized system would improve my productivity." *



7. Rate your agreement: "Photo and location attachments would help resolve issues faster." *



Figure 17: Technician Questionnaire (Interval)

D. Quantitative Insight (Ratio)

8. How many maintenance tasks do you handle per week on average? *

Your answer

9. On average, how many minutes does one task take to complete? *

Enter number of minutes

Your answer

E. Open Feedback

10. What is the biggest challenge you face in maintenance work?

Your answer

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 mainte-

Systems Analysis and Design Project

nance 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**

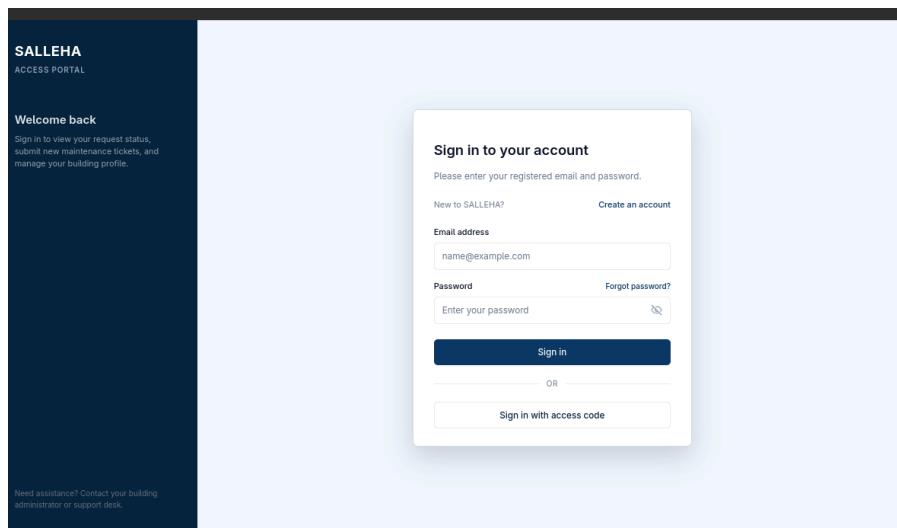
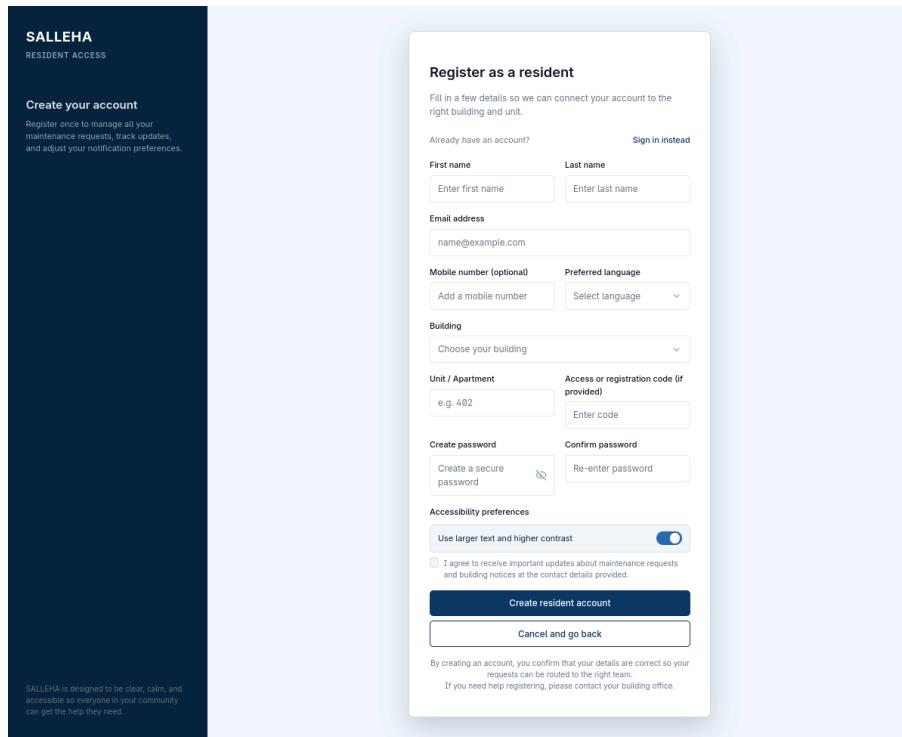


Figure 19: Login Screen

Systems Analysis and Design Project



The screenshot shows the 'Register as a resident' form. It includes fields for First name, Last name, Email address, Mobile number (optional), Preferred language, Building, Unit / Apartment, Access or registration code (if provided), Create password, Confirm password, Accessibility preferences (checkbox for larger text and higher contrast), and a checkbox for agreeing to receive updates. A 'Create resident account' button is at the bottom, along with a 'Cancel and go back' link.

Figure 20: Registration Screen

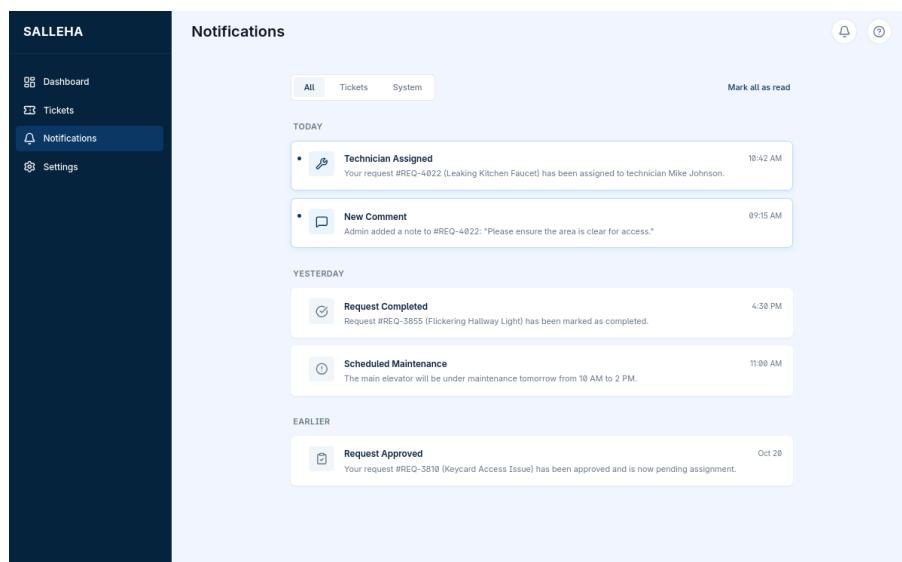


Figure 21: Notifications Screen

Systems Analysis and Design Project

• Resident Prototype Screens

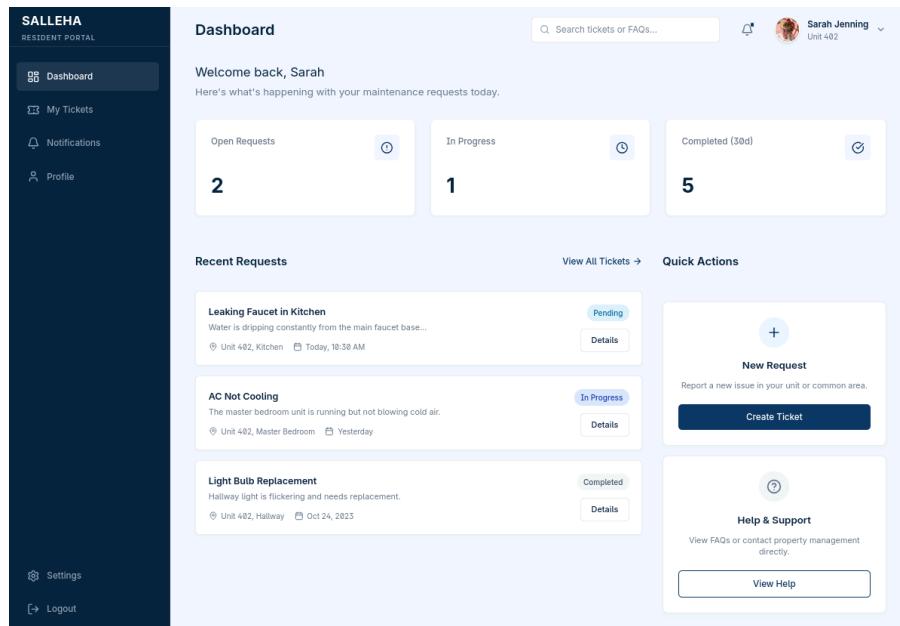


Figure 22: Resident – Dashboard

The form allows residents to create a new ticket. It includes fields for:

- Issue Details**: Ticket Title (Brief summary of the issue e.g., Leaking faucet), Category (Select category...), Urgency (Normal).
- Description**: Please describe the problem in detail. Include any specific noises, error codes, or circumstances.
- Location**: Building (Building A - North Tower) and Unit / Apartment (Unit 402).
- Photos & Attachments**: A placeholder for uploading files with instructions: Click to upload or drag and drop, SVG, PNG, JPG or GIF (max. 5MB).
- Privacy**: A toggle switch for "Submit Anonymous". A note states: Your name and contact details will be hidden from the technician. You will still receive status updates via the app.

Figure 23: Resident – Ticket Request

Systems Analysis and Design Project

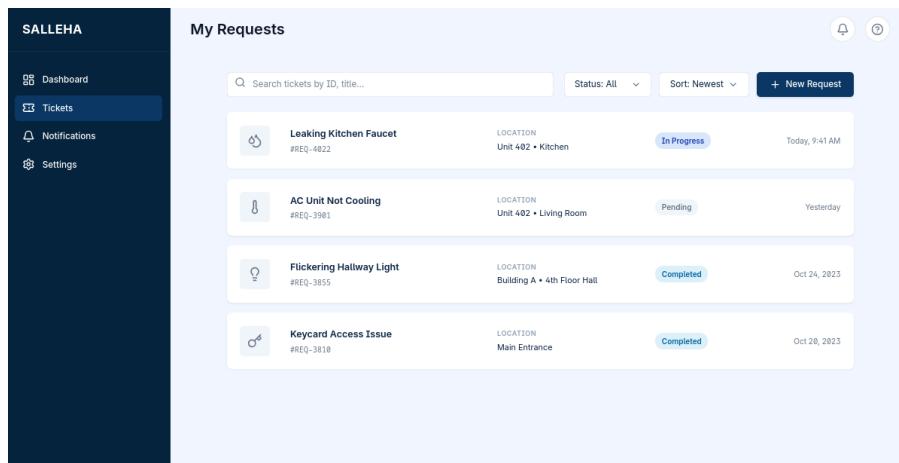


Figure 24: Resident – Tickets Screen

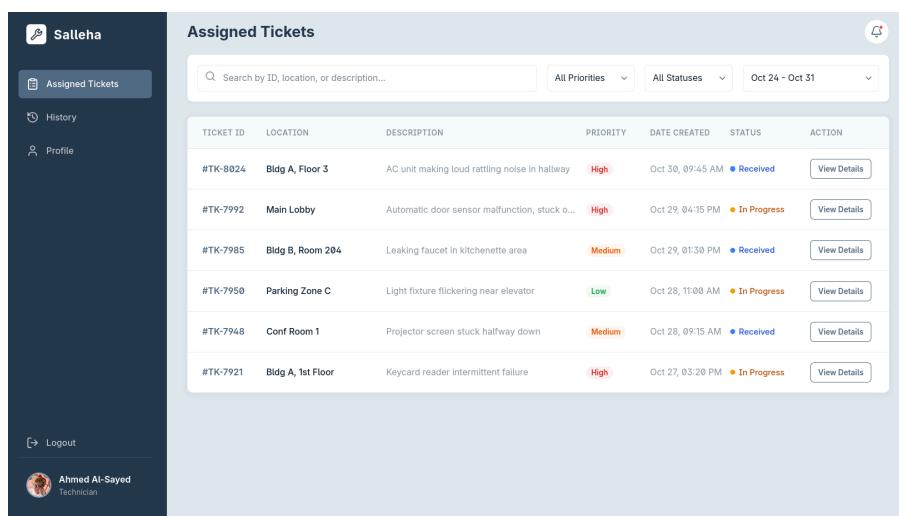


Figure 25: Technician – Dashboard

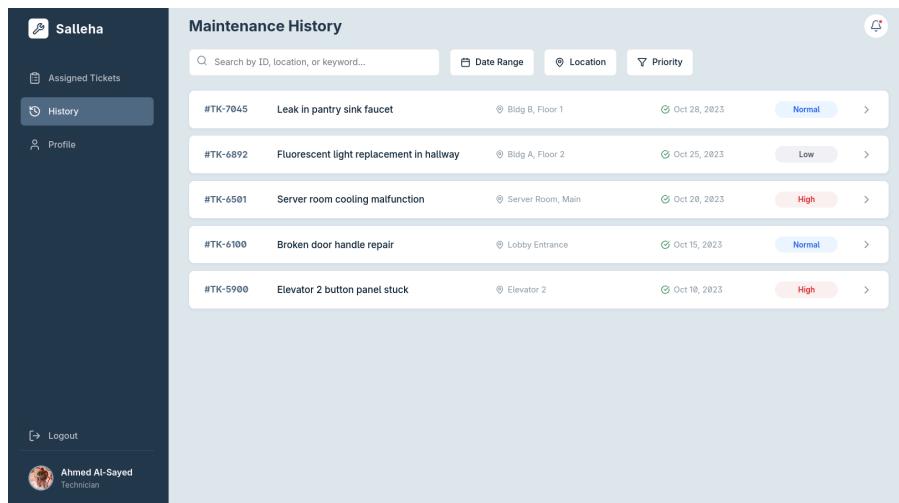


Figure 26: Technician – Maintenance History

Systems Analysis and Design Project

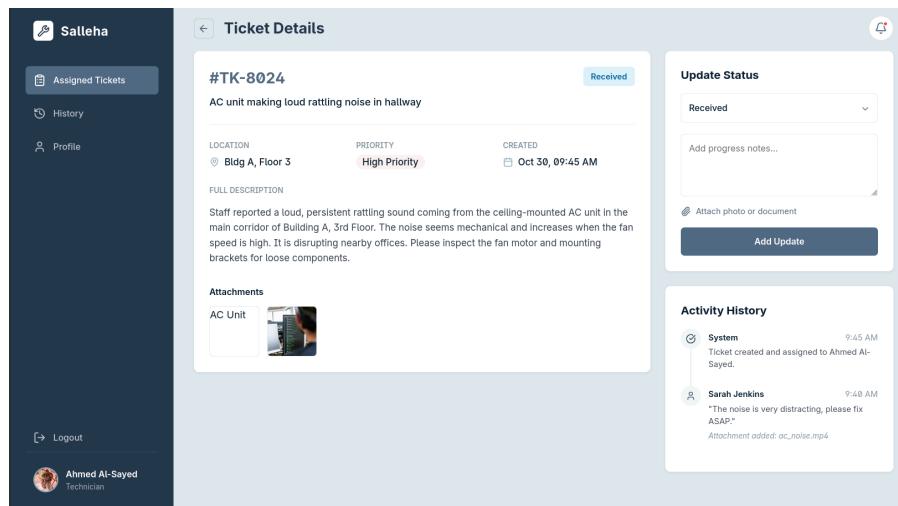


Figure 27: Technician – Ticket Details

Systems Analysis and Design Project

• Administrator Prototype Screens

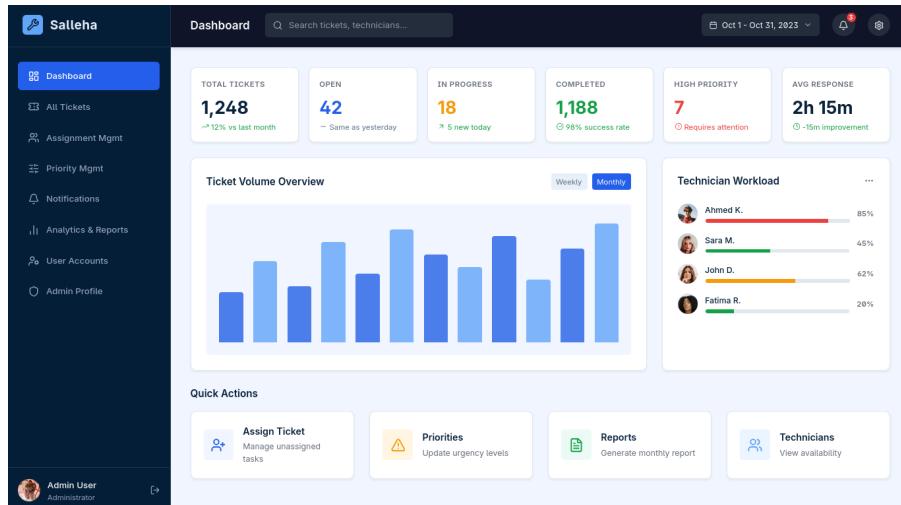


Figure 28: Administrator – Dashboard

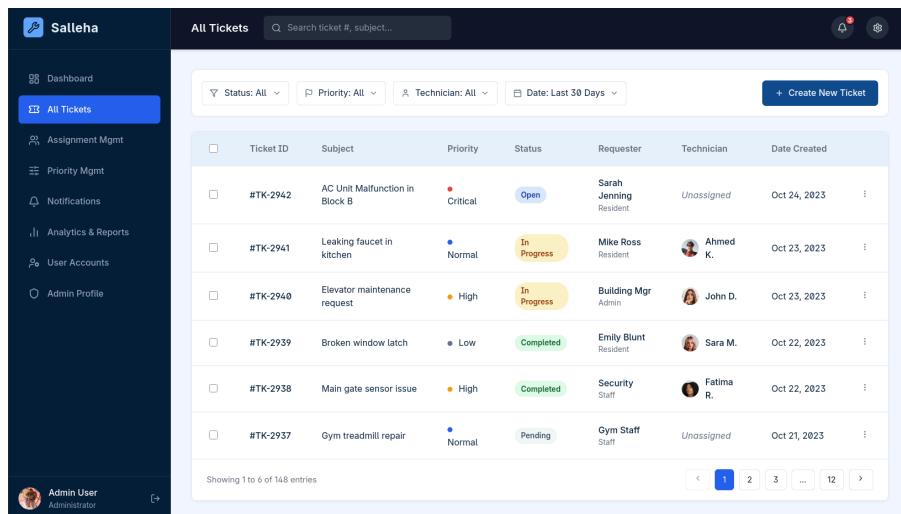


Figure 29: Administrator – All Tickets

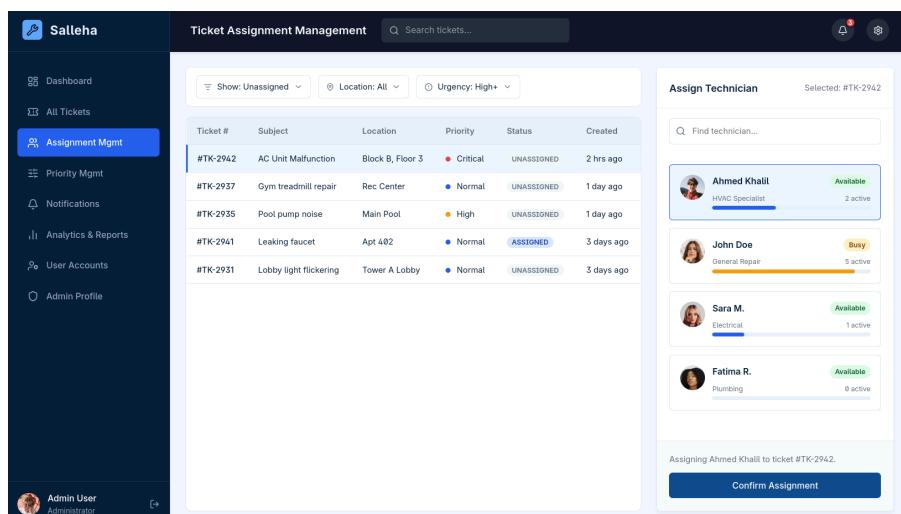


Figure 30: Administrator – Assignment Management

Systems Analysis and Design Project

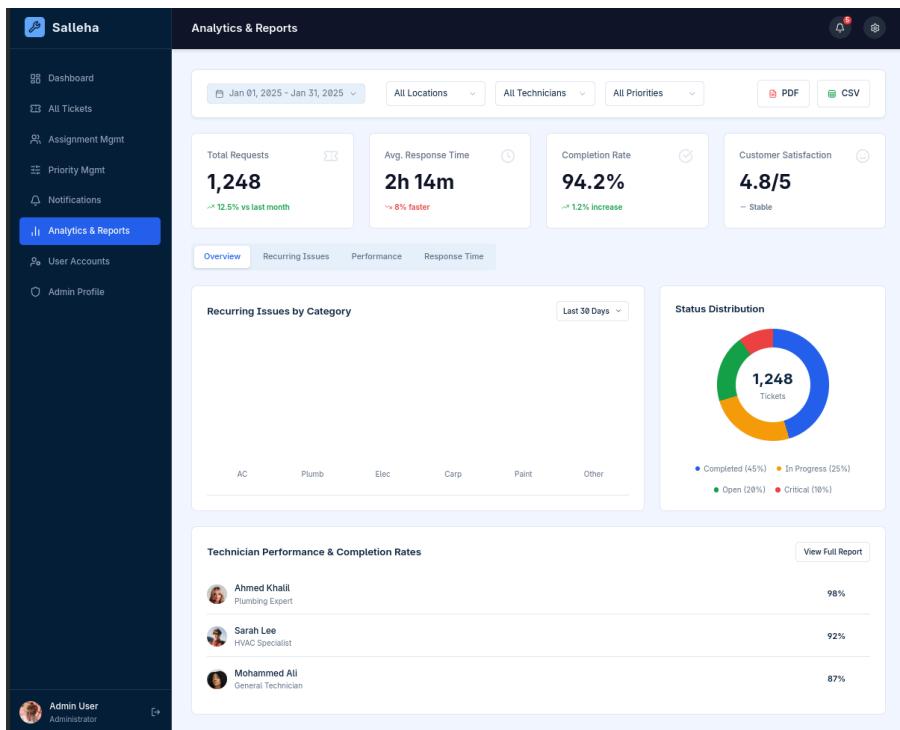


Figure 31: Administrator – Analytics & Reports

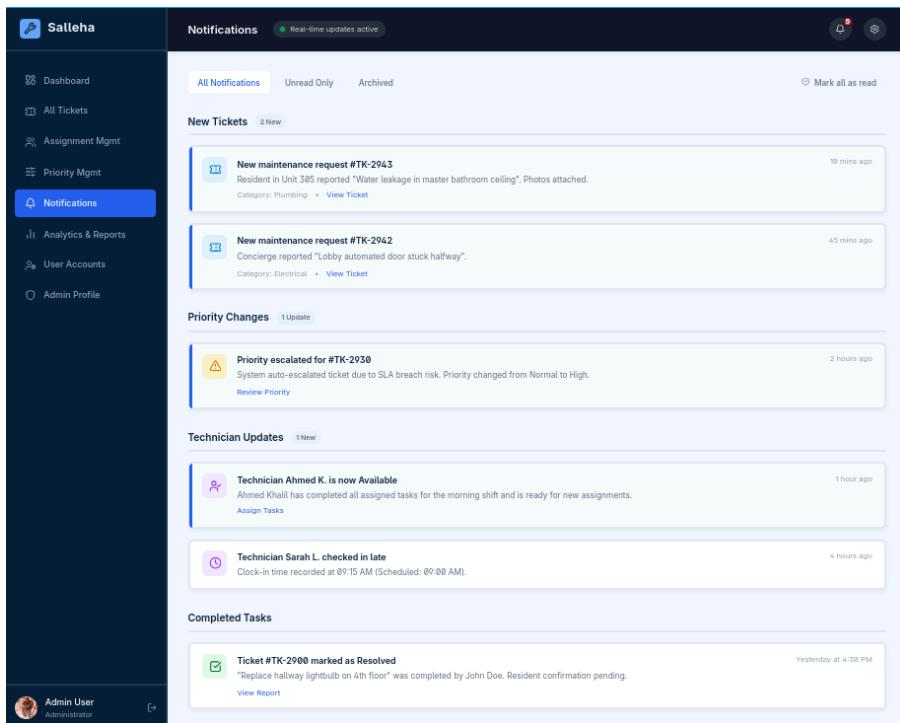


Figure 32: Administrator – Notifications

Systems Analysis and Design Project

The screenshot shows the 'Priority Management' section of the Salleha application. On the left is a dark sidebar with navigation links: Dashboard, All Tickets, Assignment Mgmt, Priority Mgmt (which is selected and highlighted in blue), Notifications, Analytics & Reports, User Accounts, and Admin Profile. The main content area has a header 'Priority Management' with a bell icon and settings gear. Below it is a table titled 'Priority Levels & Definitions' with columns: Order, Priority Name, Description & SLA, SLA Target, and Actions. It lists four priority levels: Critical (red), High (orange), Normal (blue), and Low (light blue). Each row includes a edit icon and a delete icon. Below this is a section titled 'Active Tickets by Priority' with a table showing five recent tickets. The columns are Ticket Details, Subject, Current Priority, Assigned To, and Status. The tickets are: #TK-2942 (AC Unit Malfunction - Block B, Critical, Ahmed Khalil, In Progress); #TK-2935 (Pool pump noise, High, Unassigned, Open); #TK-2941 (Leaking faucet - Apt 402, Normal, Fatima R., Completed); and #TK-2931 (Lobby light flickering, Normal, John Doe, In Progress).

Figure 33: Administrator – Priority Management

The screenshot shows the 'User Account Management' section of the Salleha application. The sidebar is identical to Figure 33. The main content area has a header 'User Account Management' with a bell icon and settings gear. Below it is a table titled 'Technicians' with columns: Technician, Contact Info, Skills, Current Load, Status, and Actions. It lists four technicians: Ahmed Khalil (Plumbing, General, 2 Active, Available), Sarah Lee (Electrical, HVAC, 5 Active, Busy), John Doe (Carpentry, 0 Active, Offline), and Mohammed Ali (HVAC, Plumbing, 3 Active, Available). Each technician row includes a edit icon and a delete icon. A search bar at the top allows searching by name, email or ID. Buttons for Filter, Export, and Add New User are also present.

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	All Users

Systems Analysis and Design Project

		from the database and redirect to the appropriate dashboard.	
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	Technician

Systems Analysis and Design Project

		Tasks, Maintenance History, Notifications, Profile.	
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,	Administrator

Systems Analysis and Design Project

		equipment performance, and task completion times.	
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

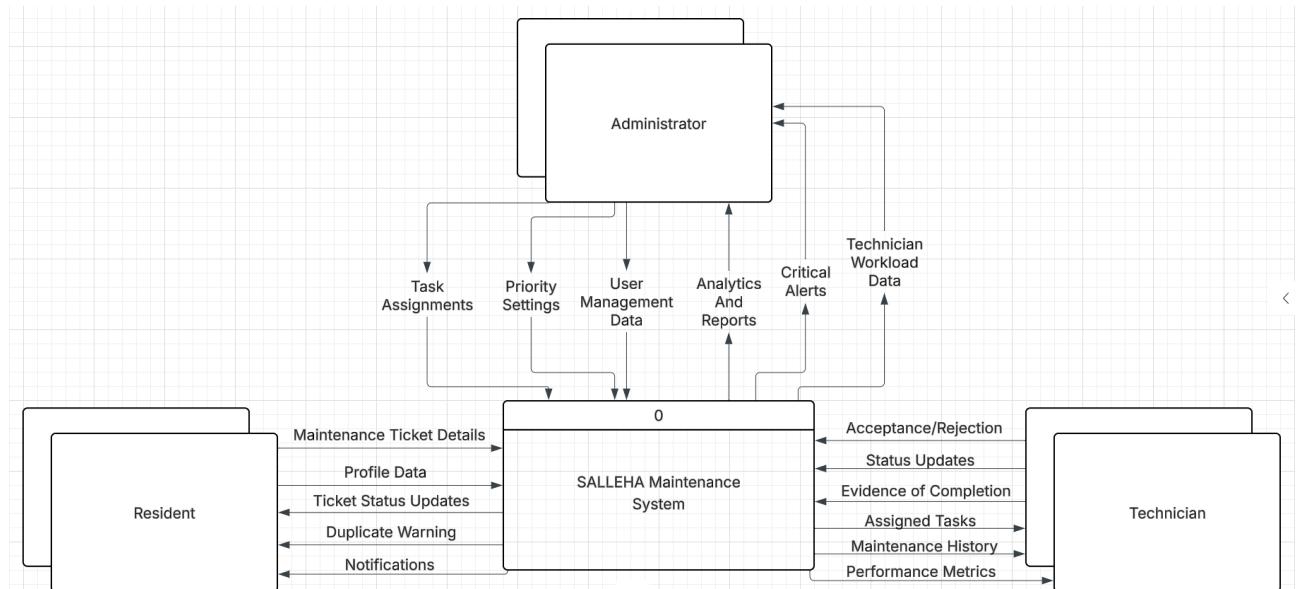


Figure 35: Context Diagram

Systems Analysis and Design Project

3.4. Use Case Model

3.4.1. Use Case diagram

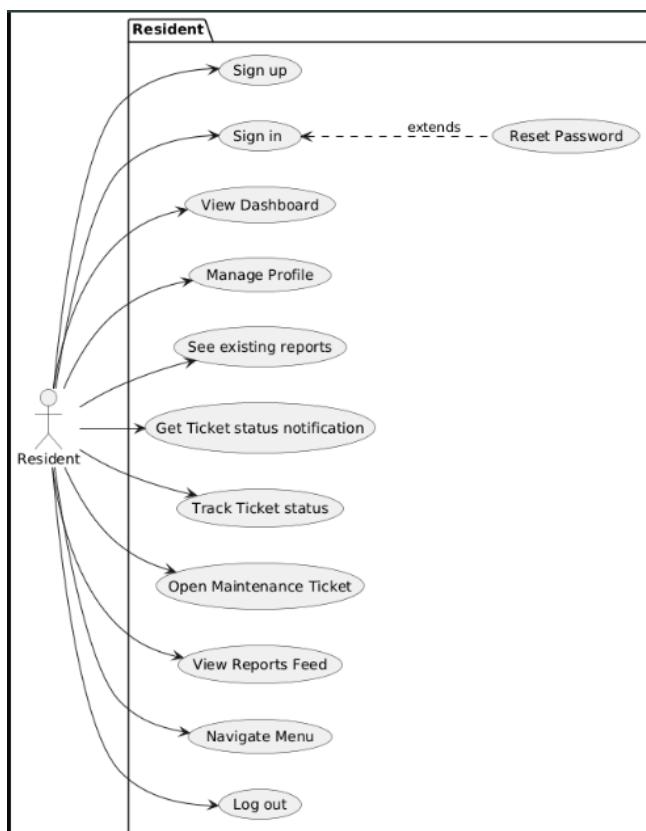


Figure 36: Use Case Diagram - Resident

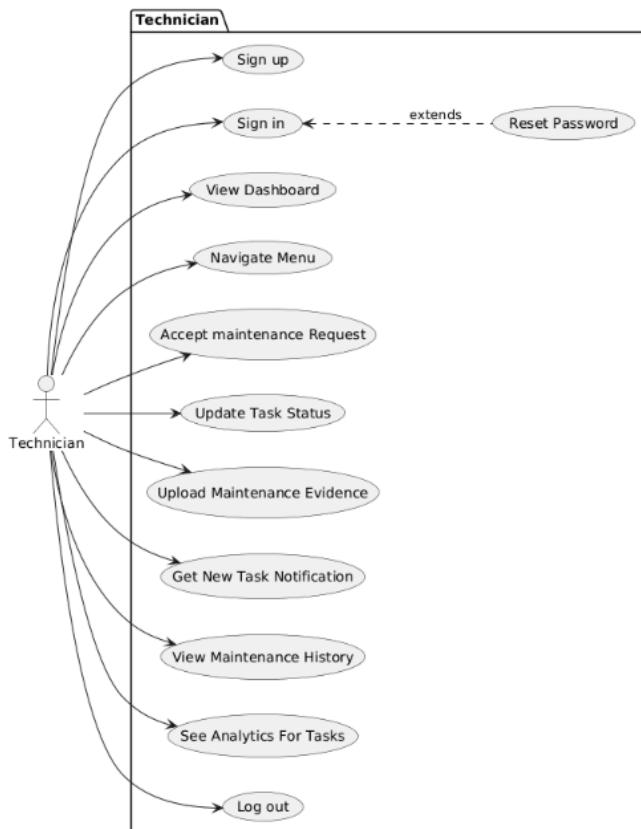


Figure 37: Use Case Diagram - Technician

Systems Analysis and Design Project

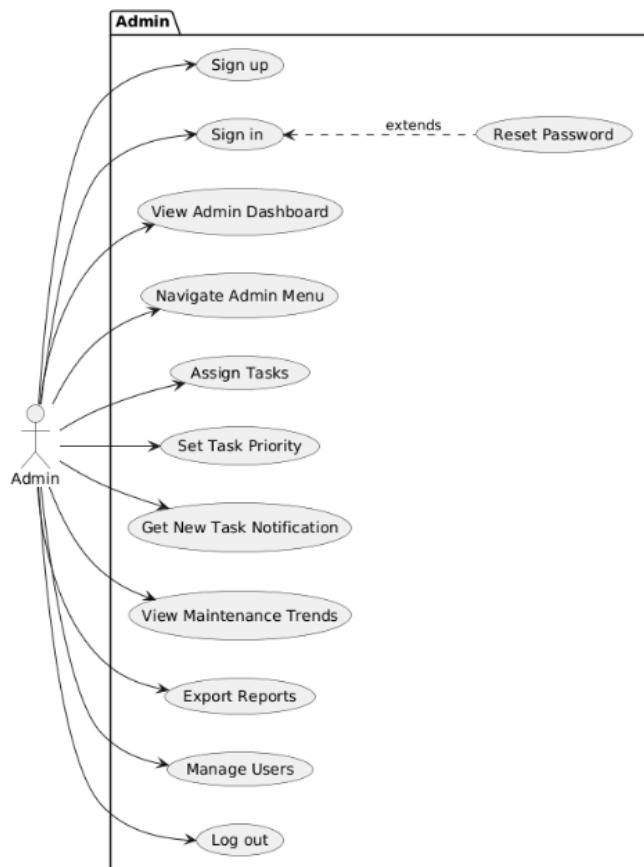


Figure 38: Use Case Diagram - Administrator

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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).

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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).

Systems Analysis and Design Project

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

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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 reassessments.
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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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 be User, Technician, Admin, or other staff), registration	UserID is the Primary Key, Email and NationalID have a uniqueness constraint, role can be User, Technician, Admin, or other staff.	<ul style="list-style-type: none"> One user can submit many requests (one to many). One user receives multiple notifications (one to many). A user can be linked to zero or one technician record

Systems Analysis and Design Project

		<p>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).</p>		<p>depending on their role (one to one or one to zero).</p>
Technician	<p>The Technician entity represents technicians or teams responsible for carrying out maintenance tasks.</p>	<p>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 tracks their</p>	<p>TechnicianID is the Primary Key, UserID is a Foreign Key referring to the User table, specialization is required.</p>	<ul style="list-style-type: none"> • One technician may handle many requests. • One technician can have many scheduled tasks.

Systems Analysis and Design Project

		<p>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.</p>		
Maintenance Request	The Maintenance Request entity documents the actual problem submitted by users or a need for service.	<p>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</p>	<p>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</p>	<ul style="list-style-type: none"> • Each maintenance request is assigned to one location. • One location can have many requests.

Systems Analysis and Design Project

		<p>for escalate tasks that remain “In Progress” beyond the estimated 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.</p>		
Location	The Location entity represents the building or place where maintenance is required, facilitating easy navigation and assignment of work to the correct place.	<p>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.</p>	<p>LocationID is the Primary Key, BuildingName is required</p>	<p>Each request is associated with exactly one location (one to many).</p>

Systems Analysis and Design Project

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

Systems Analysis and Design Project

		(String), IsRead (Boolean), CreatedAt (DateTime).		
--	--	--	--	--

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.

Systems Analysis and Design Project

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

Systems Analysis and Design Project

4. System Analysis and Design

4.1. DFDs

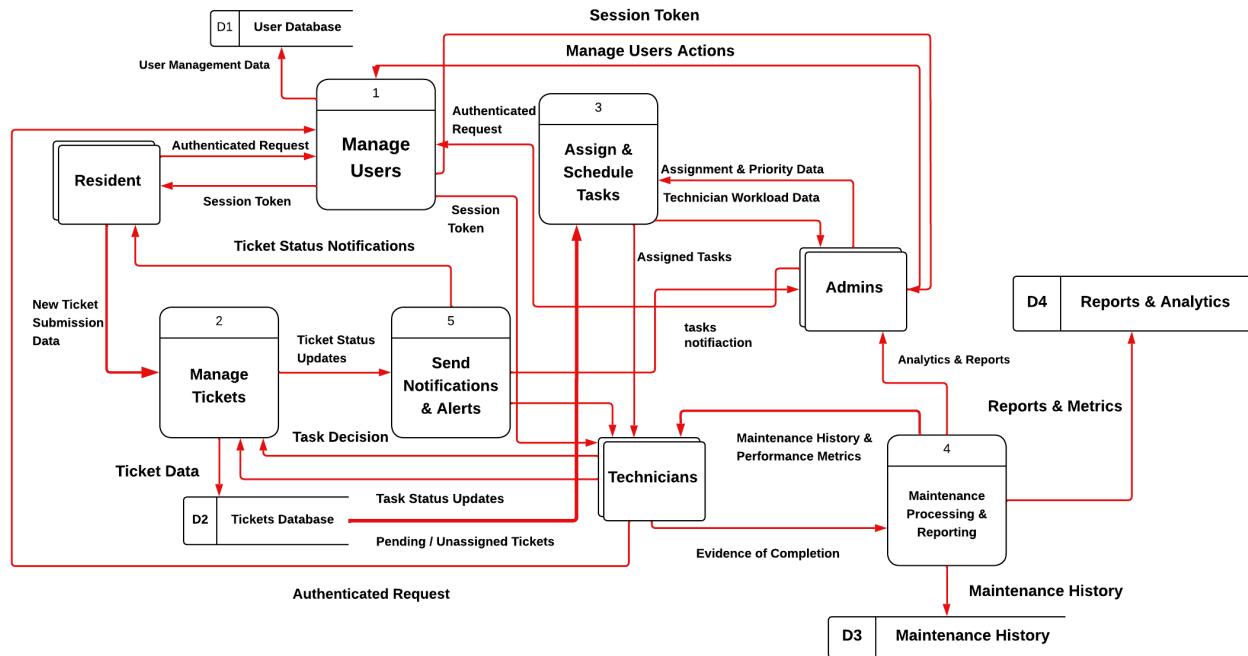


Figure 39: DFD Level 0

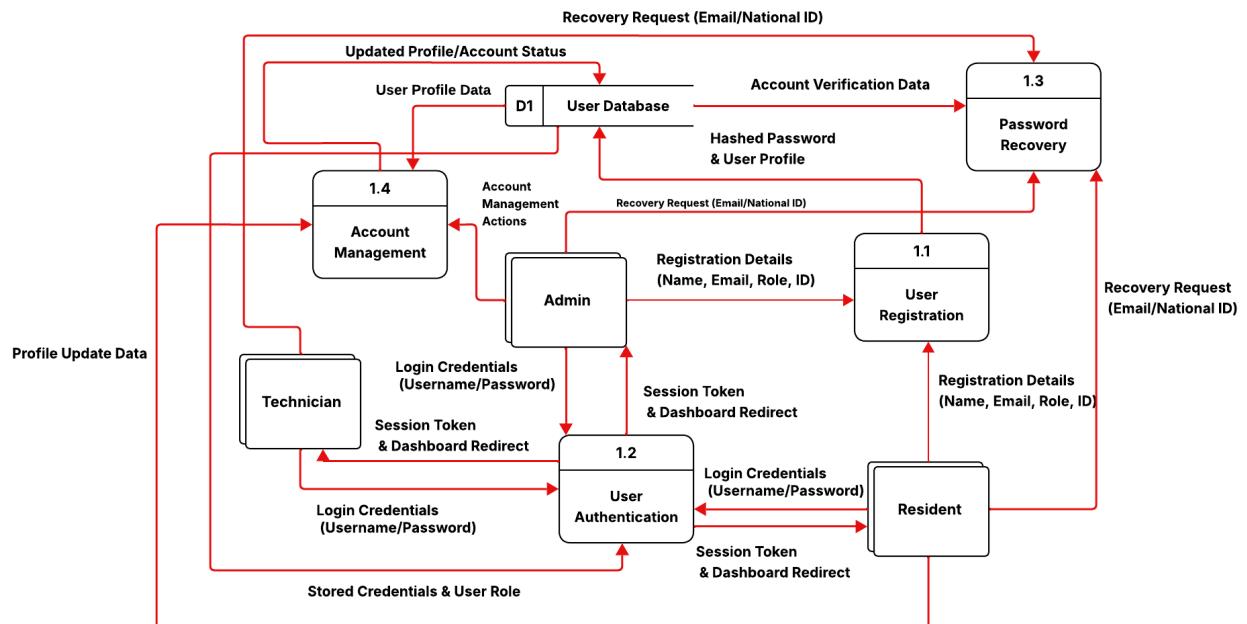


Figure 40: DFD Level 1-Fragment 1

Systems Analysis and Design Project

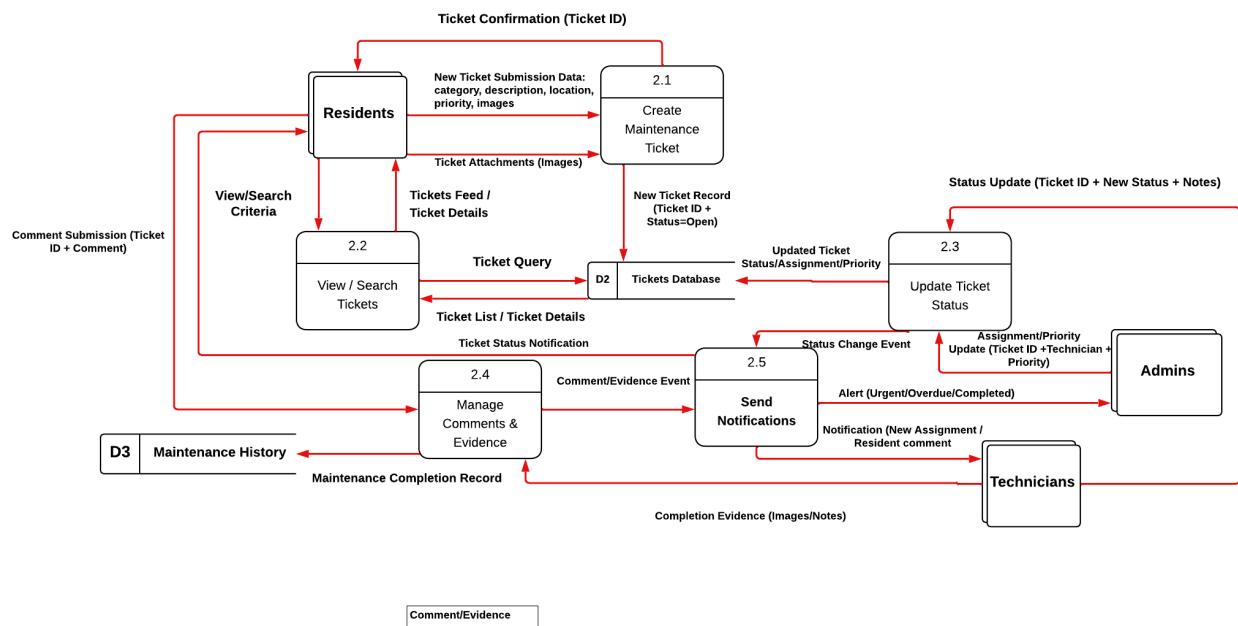


Figure 41: DFD Level 1-Fragment 2

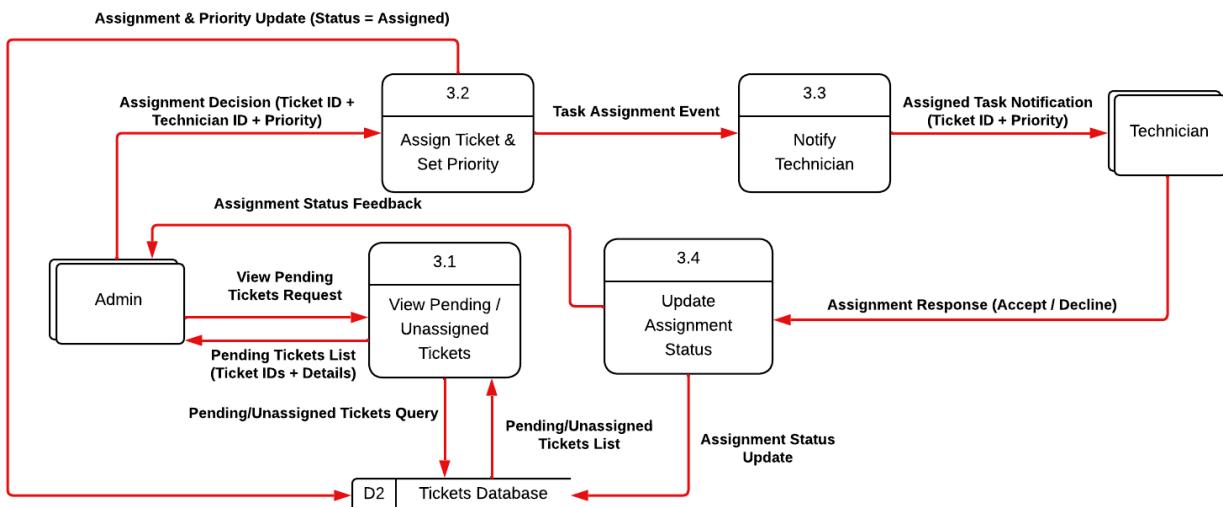


Figure 42: DFD Level 1-Fragment 3

Systems Analysis and Design Project

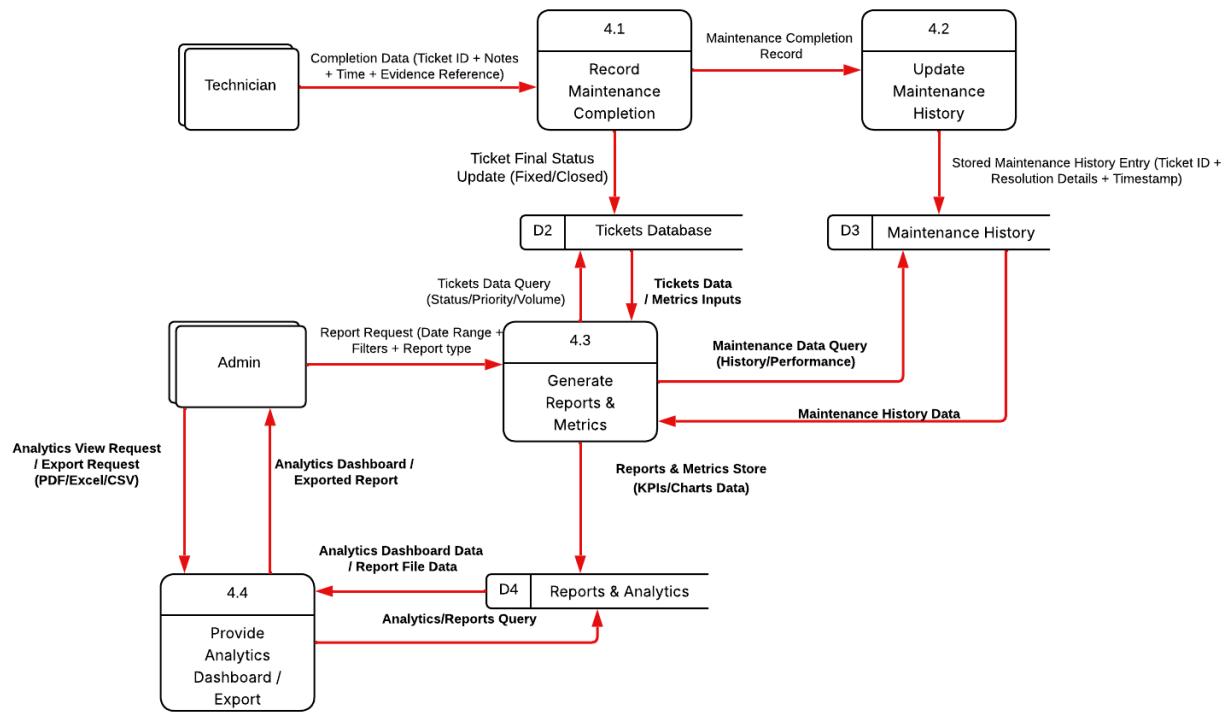


Figure 43: DFD Level 1-Fragment 4

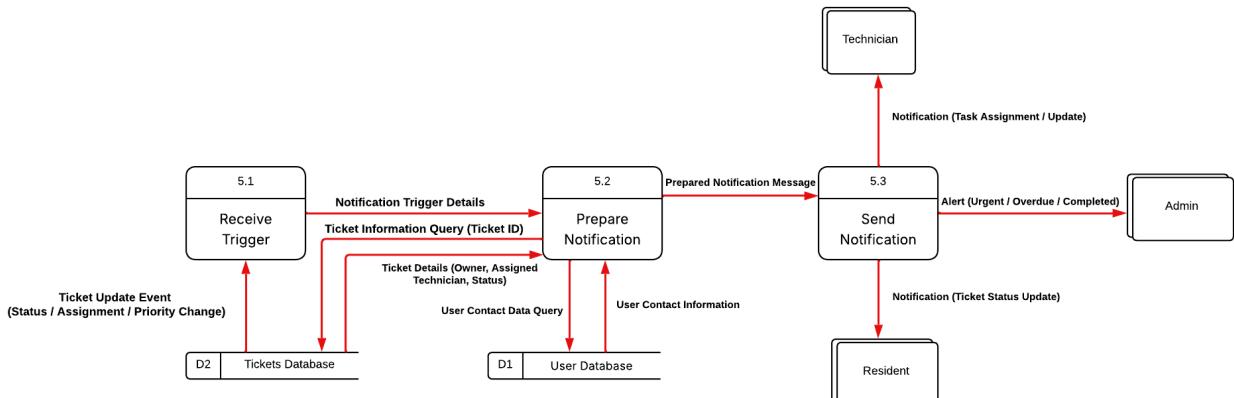


Figure 44: DFD Level 1-Fragment 5

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

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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.

Systems Analysis and Design Project

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, Time-stamp
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. Process Specifications

4.3.1. Fragment 1: User Management Processes

Number: 1.1 **Name:** User Registration **Description:** Creates a new user account by validating input data, hashing password, storing user information, and sending email verification.

Input Data Flow:

- Registration Details (from Resident/Admin/Technician)
 - Full Name
 - Email
 - National ID
 - Username
 - Password
 - Role

Output Data Flow:

- User Profile Data (to User Database D1)
- Email Verification Request (to Email Service)
- Registration Confirmation (to User)

Systems Analysis and Design Project

Type of Process: Online Batch Manual

Subprogram/Function Name: registerUser()

Process Logic (Structured English):

BEGIN Process 1.1: User Registration

READ Registration Details from User

VALIDATE Email Format

IF Email Format Invalid THEN

 DISPLAY "Invalid email format"

 EXIT Process

ENDIF

VALIDATE Password Strength

IF Password Length < 8 OR

 Missing Uppercase OR

 Missing Lowercase OR

 Missing Number OR

 Missing Special Character THEN

 DISPLAY "Password does not meet strength requirements"

 EXIT Process

ENDIF

CHECK Email Uniqueness in User Database

IF Email Already Exists THEN

 DISPLAY "Email already registered"

 EXIT Process

ENDIF

CHECK National ID Uniqueness in User Database

IF National ID Already Exists THEN

 DISPLAY "National ID already registered"

 EXIT Process

ENDIF

HASH Password using bcrypt

GENERATE UserID

SET IsEmailVerified = FALSE

SET AccountStatus = "Pending Verification"

SET RegistrationDate = CURRENT_DATETIME

SET FailedLoginAttempts = 0

WRITE User Record to User Database (D1)

GENERATE Email Verification Token

SEND Verification Email with Activation Link

DISPLAY "Registration successful. Please check your email to verify your account."

Systems Analysis and Design Project

END Process

Refer to:

- FR-1.1 through FR-1.8
- Table 52: Data Flow 1 Details

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Email Format Valid	Y	Y	Y	Y	N
Password Strength Valid	Y	Y	Y	N	-
Email Unique	Y	Y	N	-	-
National ID Unique	Y	N	-	-	-
Create User Account	X	-	-	-	-
Send Verification Email	X	-	-	-	-
Display “National ID exists”	-	X	-	-	-
Display “Email exists”	-	-	X	-	-
Display “Weak password”	-	-	-	X	-
Display “Invalid email”	-	-	-	-	X

Unresolved Issues: None

Number: 1.2 **Name:** User Authentication **Description:** Authenticates user credentials, manages failed login attempts, enforces CAPTCHA and account locking, creates session tokens, and redirects to role-specific dashboards.

Input Data Flow:

- Login Credentials (from Resident/Technician/Admin)
 - Username
 - Password
 - CAPTCHA (if required)

Output Data Flow:

- Session Token & Dashboard Redirect (to User)
- Updated Failed Login Attempts (to User Database D1)

Type of Process: Online Batch Manual

Subprogram/Function Name: authenticateUser()

Process Logic (Structured English):

Systems Analysis and Design Project

```
BEGIN Process 1.2: User Authentication

READ Username and Password from User

RETRIEVE User Record from User Database (D1) by Username
IF User Record Not Found THEN
    DISPLAY "Invalid credentials"
    EXIT Process
ENDIF

CHECK AccountStatus
IF AccountStatus = "Locked" THEN
    DISPLAY "Account is locked. Contact administrator or verify email to unlock."
    EXIT Process
ENDIF

CHECK FailedLoginAttempts
IF FailedLoginAttempts >= 3 AND FailedLoginAttempts < 5 THEN
    REQUIRE CAPTCHA Validation
    IF CAPTCHA Invalid THEN
        DISPLAY "Invalid CAPTCHA"
        EXIT Process
    ENDIF
ENDIF

VERIFY Password against PasswordHash using bcrypt
IF Password Incorrect THEN
    INCREMENT FailedLoginAttempts

    IF FailedLoginAttempts >= 5 THEN
        SET AccountStatus = "Locked"
        SEND Notification "Account Locked" to User Email
        DISPLAY "Account locked due to multiple failed attempts"
    ELSE
        DISPLAY "Invalid credentials"
   ENDIF

    UPDATE User Record in User Database (D1)
    EXIT Process
ENDIF

CHECK IsEmailVerified
IF IsEmailVerified = FALSE THEN
    DISPLAY "Please verify your email before logging in"
    EXIT Process
ENDIF

// Successful Authentication
SET FailedLoginAttempts = 0
SET LastLoginAt = CURRENT_DATETIME
GENERATE Session Token (JWT)

DETERMINE Session Timeout based on Role
IF Role = "Resident" THEN
```

Systems Analysis and Design Project

```
    SET SessionTimeout = 30 minutes
ELSIF Role = "Technician" THEN
    SET SessionTimeout = 15 minutes
ELSIF Role = "Administrator" THEN
    SET SessionTimeout = 60 minutes
ENDIF

STORE Session in Redis Cache
UPDATE User Record in User Database (D1)

REDIRECT to Role-Specific Dashboard
IF Role = "Resident" THEN REDIRECT to Resident Dashboard
ELSIF Role = "Technician" THEN REDIRECT to Technician Dashboard
ELSIF Role = "Administrator" THEN REDIRECT to Administrator Dashboard
ENDIF

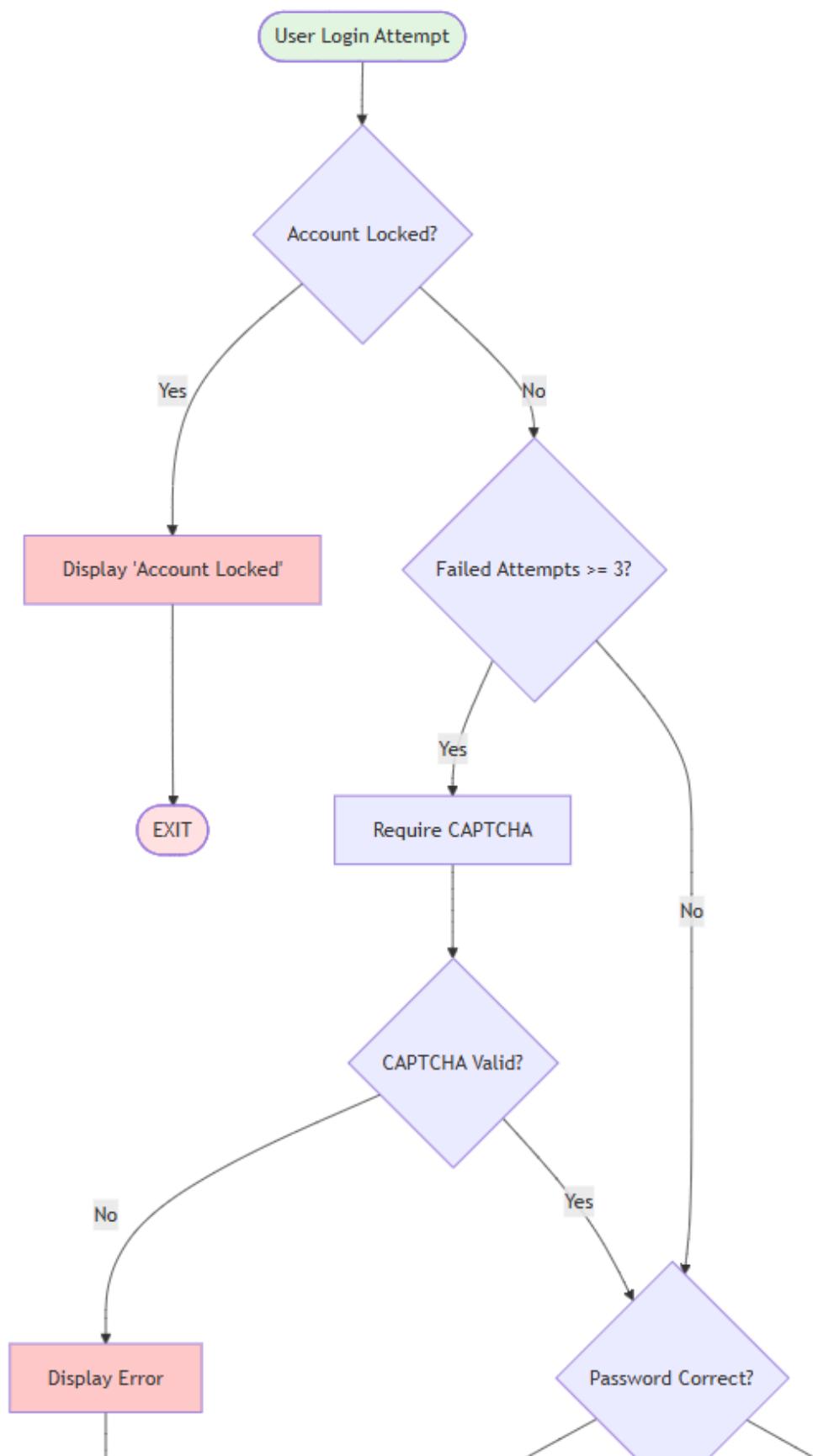
END Process
```

Refer to:

- FR-2.1 through FR-2.8
- Table 55: Data Flow 4 Details
- Table 56: Data Flow 5 Details

Systems Analysis and Design Project

Decision Tree:



Systems Analysis and Design Project

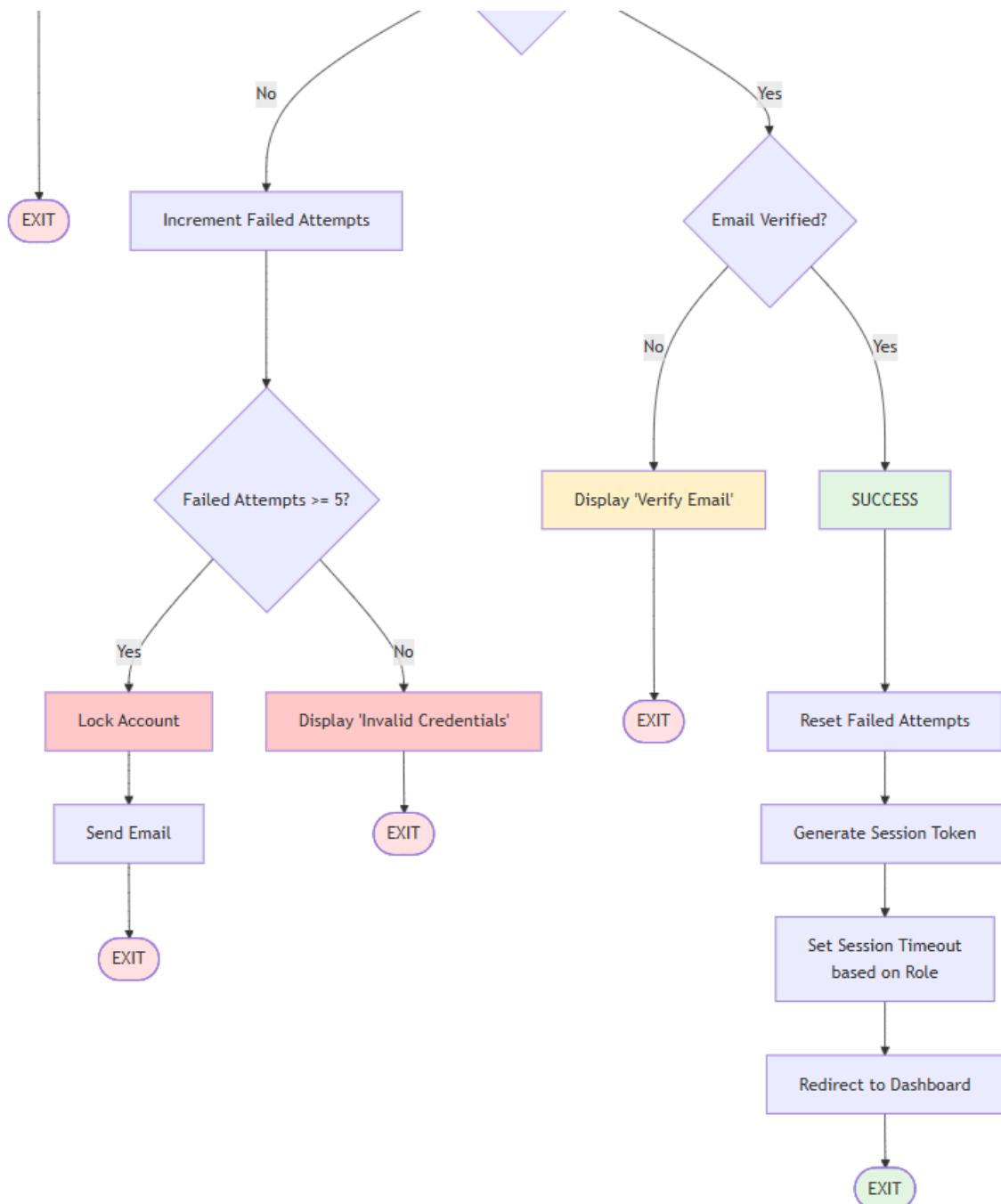


Figure 46: Decision Tree 1.2

Number: 1.3 **Name:** Password Recovery **Description:** Allows users to reset forgotten passwords by verifying email and national ID, sending one-time verification code, and updating password after validation.

Input Data Flow:

- Recovery Request (from Resident/Technician/Admin)
 - Email OR National ID
 - Verification Code (after email sent)
 - New Password
 - Password Confirmation

Systems Analysis and Design Project

Output Data Flow:

- Account Verification Data (from User Database D1)
- Updated Password (to User Database D1)
- Verification Email (to Email Service)

Type of Process: Online Batch Manual

Subprogram/Function Name: recoverPassword()

Process Logic (Structured English):

```
BEGIN Process 1.3: Password Recovery

// Step 1: Initiate Recovery
READ Email and National ID from User

RETRIEVE User Record from User Database (D1)
IF User Record Not Found OR
    Email Not Matching OR
    National ID Not Matching THEN
        DISPLAY "Invalid information provided"
        EXIT Process
ENDIF

GENERATE One-Time Verification Code (6-digit random number)
SET Code Expiration = CURRENT_DATETIME + 15 minutes
STORE Verification Code with UserID and Expiration in Redis Cache

SEND Verification Code to User Email
DISPLAY "Verification code sent to your email"

// Step 2: Verify Code
READ Verification Code from User

RETRIEVE Stored Code from Redis Cache by UserID
IF Code Not Found THEN
    DISPLAY "Verification code expired or invalid"
    EXIT Process
ENDIF

IF Current Time > Code Expiration THEN
    DELETE Code from Redis Cache
    DISPLAY "Verification code expired. Please request a new one."
    EXIT Process
ENDIF

IF Entered Code ≠ Stored Code THEN
    DISPLAY "Invalid verification code"
    EXIT Process
ENDIF
```

Systems Analysis and Design Project

```
// Step 3: Reset Password
READ New Password and Password Confirmation from User

IF New Password ≠ Password Confirmation THEN
    DISPLAY "Passwords do not match"
    EXIT Process
ENDIF

VALIDATE Password Strength
IF Password Length < 8 OR
    Missing Uppercase OR
    Missing Lowercase OR
    Missing Number OR
    Missing Special Character THEN
    DISPLAY "Password does not meet strength requirements"
    EXIT Process
ENDIF

RETRIEVE Last 3 Passwords from User Database (D1)
FOR EACH Previous Password DO
    IF bcrypt Compare(New Password, Previous Password Hash) = TRUE THEN
        DISPLAY "Cannot reuse recent passwords"
        EXIT Process
    ENDIF
ENDFOR

HASH New Password using bcrypt
UPDATE PasswordHash in User Database (D1)
ARCHIVE Old Password Hash to Password History
DELETE Verification Code from Redis Cache

SEND Confirmation Email "Password Changed Successfully"
DISPLAY "Password reset successful. Please login with your new password."

END Process
```

Refer to:

- FR-4.1 through FR-4.10
- Table 57: Data Flow 6 Details
- Table 58: Data Flow 7 Details

Decision Table:

Systems Analysis and Design Project

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
User Found with Email+ID	Y	Y	Y	Y	Y	N
Code Valid & Not Expired	Y	Y	Y	Y	N	-
Passwords Match	Y	Y	Y	N	-	-
Password Strength Valid	Y	Y	N	-	-	-
Not in Last 3 Passwords	Y	N	-	-	-	-
Update Password	X	-	-	-	-	-
Send Confirmation Email	X	-	-	-	-	-
Display "Cannot reuse password"	-	X	-	-	-	-
Display "Weak password"	-	-	X	-	-	-
Display "Passwords don't match"	-	-	-	X	-	-
Display "Code expired/invalid"	-	-	-	-	X	-
Display "Invalid information"	-	-	-	-	-	X

Unresolved Issues: None

Number: 1.4 **Name:** Account Management **Description:** Allows users to view and update profile information, and allows administrators to manage user accounts (activate, deactivate, update roles).

Input Data Flow:

- Profile Update Data (from Resident)
- User Management Commands (from Administrator)
- User Profile Data (from User Database D1)

Output Data Flow:

- Updated Profile/Account Status (to User Database D1)

Type of Process: Online Batch Manual

Subprogram/Function Name: manageUserAccount()

Process Logic (Structured English):

BEGIN Process 1.4: Account Management

READ Action Type (View, Update Profile, Admin Action)

// Resident Profile Update

IF Action = "Update Profile" THEN

 READ UserID and Updated Fields from Resident

 RETRIEVE Current Profile from User Database (D1)

 IF Updating Sensitive Field (Email, Password) THEN

Systems Analysis and Design Project

```
REQUIRE Password Verification
IF Password Verification Fails THEN
    DISPLAY "Password verification required"
    EXIT Process
ENDIF
ENDIF

IF Updating Email THEN
    CHECK Email Uniqueness
    IF Email Already Exists THEN
        DISPLAY "Email already in use"
        EXIT Process
    ENDIF

    GENERATE Email Verification Token
    SEND Verification Email to New Email Address
    SET PendingEmail = New Email
    DISPLAY "Verification email sent. Email will be updated after
verification."
ENDIF

IF Updating Password THEN
    VALIDATE Password Strength
    IF Password Strength Invalid THEN
        DISPLAY "Password does not meet requirements"
        EXIT Process
    ENDIF

    CHECK Last 3 Passwords
    IF Password Used Recently THEN
        DISPLAY "Cannot reuse recent passwords"
        EXIT Process
    ENDIF

    HASH New Password
    UPDATE PasswordHash
ENDIF

IF Updating Profile Picture THEN
    VALIDATE File Type (JPEG, PNG)
    VALIDATE File Size (< 5MB)
    IF Validation Fails THEN
        DISPLAY "Invalid file format or size"
        EXIT Process
    ENDIF
    UPLOAD to AWS S3
    UPDATE ProfilePictureURL
ENDIF

IF Updating Full Name THEN
    UPDATE FullName
ENDIF

LOG Profile Change with Timestamp and IP Address
```

Systems Analysis and Design Project

```
UPDATE User Record in User Database (D1)
DISPLAY "Profile updated successfully"
ENDIF

// Administrator User Management
IF Action = "Admin Manage User" THEN
    VERIFY Admin Privileges
    IF Not Administrator THEN
        DISPLAY "Unauthorized action"
        EXIT Process
    ENDIF

READ TargetUserID and Admin Action

IF Admin Action = "Activate Account" THEN
    SET AccountStatus = "Active"
    SET IsEmailVerified = TRUE
    SEND Notification to User
ELSIF Admin Action = "Deactivate Account" THEN
    SET AccountStatus = "Suspended"
    INVALIDATE All Active Sessions
    SEND Notification to User
ELSIF Admin Action = "Delete Account" THEN
    SOFT DELETE User Record (mark as deleted)
    INVALIDATE All Active Sessions
    ARCHIVE User Data
ELSIF Admin Action = "Unlock Account" THEN
    SET AccountStatus = "Active"
    SET FailedLoginAttempts = 0
    SEND Notification to User
ELSIF Admin Action = "Reset Password" THEN
    GENERATE Temporary Password
    HASH Temporary Password
    UPDATE PasswordHash
    SET ForcePasswordChange = TRUE
    SEND Temporary Password to User Email
ELSIF Admin Action = "Update Role" THEN
    REQUIRE Approval Workflow
    IF Approval Granted THEN
        UPDATE Role
        UPDATE Permissions
        SEND Notification to User
    ENDIF
ELSIF Admin Action = "Assign Technician Specialization" THEN
    IF User Role = "Technician" THEN
        UPDATE Specialization
        UPDATE SkillLevel
        UPDATE WorkZone
    ELSE
        DISPLAY "User is not a technician"
    ENDIF
ENDIF

LOG Admin Action with Timestamp and Admin ID
```

Systems Analysis and Design Project

```
    UPDATE User Record in User Database (D1)
    DISPLAY "User account updated successfully"
ENDIF

END Process
```

Refer to:

- FR-12 (Resident Profile Management)
- FR-28 (User Account Management)
- Table 59: Data Flow 8 Details
- Table 60: Data Flow 9 Details

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Actor = Resident	Y	Y	Y	N	N
Actor = Administrator	N	N	N	Y	Y
Updating Sensitive Field	Y	Y	N	-	-
Password Verified	Y	N	-	-	-
Admin Privileges Valid	-	-	-	Y	N
Allow Profile Update	X	-	X	-	-
Require Password Verification	X	X	-	-	-
Deny Update	-	X	-	-	X
Allow Admin Action	-	-	-	X	-
Log Change	X	-	X	X	-

Unresolved Issues: None

4.3.2. Fragment 2: Maintenance Request Management Processes

Number: 2.1 **Name:** Create Maintenance Ticket **Description:** Validates and creates a new maintenance ticket, checks for duplicates, uploads images, generates unique ticket ID, and stores ticket information.

Input Data Flow:

- New Ticket Submission Data (from Resident)
 - Category
 - Description
 - Location
 - Priority
- Ticket Attachments/Images (from Resident)

Output Data Flow:

- New Ticket Record (to Tickets Database D2)
- Ticket Confirmation (to Resident)

Systems Analysis and Design Project

- Notification Trigger (to Process 2.5)

Type of Process: Online Batch Manual

Subprogram/Function Name: createMaintenanceTicket()

Process Logic (Structured English):

```
BEGIN Process 2.1: Create Maintenance Ticket

READ Ticket Details from Resident
    Category, Description, Location, Priority, Images

VALIDATE Mandatory Fields
IF Category = NULL OR
    Description = NULL OR
    Location = NULL OR
    Priority = NULL THEN
    DISPLAY "All mandatory fields must be filled"
    EXIT Process
ENDIF

VALIDATE Description Length
IF Description Length < 20 characters THEN
    DISPLAY "Description must be at least 20 characters"
    EXIT Process
ENDIF

VALIDATE Images (if provided)
IF Images Attached THEN
    IF Number of Images > 5 THEN
        DISPLAY "Maximum 5 images allowed"
        EXIT Process
    ENDIF

    FOR EACH Image DO
        VALIDATE File Type (JPEG, PNG only)
        VALIDATE File Size (< 5MB per image)
        IF Validation Fails THEN
            DISPLAY "Invalid image format or size"
            EXIT Process
        ENDIF
    ENDFOR
ENDIF

// Duplicate Detection
CHECK for Similar Tickets in Last 7 Days
QUERY Tickets Database (D2) WHERE:
    Category = Input Category AND
    Location = Input Location AND
    ReportedAt >= (CURRENT_DATE - 7 days)
```

Systems Analysis and Design Project

```
CALCULATE Similarity Score for Description
FOR EACH Retrieved Ticket DO
    COMPUTE Text Similarity (Cosine Similarity, Levenshtein Distance)
    IF Similarity Score > 80% THEN
        SET SimilarityFlag = TRUE
        ADD Ticket to Similar Tickets List
    ENDIF
ENDFOR

IF SimilarityFlag = TRUE THEN
    DISPLAY Warning "Similar tickets found:"
    DISPLAY Similar Tickets List with Links
    PROMPT User "Proceed with submission?" OR "Add comment to existing ticket" OR
    "Cancel"

    IF User Selects "Cancel" THEN
        EXIT Process
    ELSIF User Selects "Add comment to existing ticket" THEN
        REDIRECT to Process 2.4 (Add Comment)
        EXIT Process
    ENDIF
    // If Proceed, continue below
ENDIF

// Upload Images to AWS S3
IF Images Attached THEN
    FOR EACH Image DO
        COMPRESS Image (if size > 1MB)
        GENERATE Unique Image Name
        UPLOAD to AWS S3 Bucket
        STORE Image URL
    ENDFOR
ENDIF

// Create Ticket
GENERATE Unique TicketID (Format: TKT-YYYYMMDD-XXXXX)
SET Status = "Open"
SET ReportedAt = CURRENT_DATETIME
SET RequesterID = Current User ID
SET TechnicianID = NULL (not yet assigned)
SET AssignedAt = NULL
SET ResolvedAt = NULL
SET IsUrgentFlag = FALSE
SET EstimatedCompletionTime = CALCULATE based on Priority

WRITE Ticket Record to Tickets Database (D2)

// Calculate Estimated Response Time
IF Priority = "Emergency" THEN
    EstimatedResponse = "Within 1 hour"
ELSIF Priority = "Urgent" THEN
    EstimatedResponse = "Within 4 hours"
ELSIF Priority = "High" THEN
    EstimatedResponse = "Within 24 hours"
```

Systems Analysis and Design Project

```

ELSIF Priority = "Medium" THEN
    EstimatedResponse = "Within 3 days"
ELSE // Low
    EstimatedResponse = "Within 7 days"
ENDIF

SEND Notification to Administrators (Process 2.5)
DISPLAY Confirmation Page
    "Ticket Created Successfully"
    "Ticket ID: " + TicketID
    "Estimated Response Time: " + EstimatedResponse

REDIRECT to Ticket Details View

END Process

```

Refer to:

- FR-8 (Open Maintenance Ticket)
- FR-11 (Duplicate Ticket Prevention)
- Table 61: Data Flow 10 Details
- Table 62: Data Flow 11 Details
- Table 63: Data Flow 12 Details
- Table 64: Data Flow 13 Details

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
All Mandatory Fields Filled	Y	Y	Y	Y	Y	N
Description >= 20 chars	Y	Y	Y	Y	N	-
Images Valid (if provided)	Y	Y	Y	N	-	-
Duplicate Detected	Y	Y	N	-	-	-
User Proceeds Anyway	Y	N	-	-	-	-
Create Ticket	X	-	X	-	-	-
Generate Ticket ID	X	-	X	-	-	-
Send Admin Notification	X	-	X	-	-	-
Show Confirmation	X	-	X	-	-	-
Show Duplicates Warning	X	X	-	-	-	-
User Chooses Action	X	X	-	-	-	-
Display "Invalid images"	-	-	-	X	-	-
Display "Description too short"	-	-	-	-	X	-
Display "Missing fields"	-	-	-	-	-	X

Unresolved Issues: Should the expected arrival date of goods on order be considered when calculating estimated completion time for tickets requiring parts?

Systems Analysis and Design Project

Number: 2.2 **Name:** View/Search Tickets **Description:** Retrieves and displays tickets based on search criteria, applies filters, paginates results, and shows ticket details.

Input Data Flow:

- View/Search Criteria (from Resident/Technician/Admin)
 - Ticket ID (optional)
 - Status
 - Category
 - Date Range
 - Keyword

Output Data Flow:

- Ticket List/Ticket Details (from Tickets Database D2 to User)

Type of Process: Online Batch Manual

Subprogram/Function Name: viewSearchTickets()

Process Logic (Structured English):

```
BEGIN Process 2.2: View/Search Tickets

READ Search Criteria from User
    Optional: TicketID, Status, Category, DateRange, Keyword

GET Current User Role and UserID

// Build Query
INITIALIZE Query = "SELECT * FROM Tickets WHERE 1=1"

// Role-based filtering
IF User Role = "Resident" THEN
    ADD to Query "AND RequesterID = Current UserID"
ELSIF User Role = "Technician" THEN
    ADD to Query "AND (TechnicianID = Current UserID OR Status = 'Open')"
ELSIF User Role = "Administrator" THEN
    // No restriction - can view all tickets
ENDIF

// Apply Search Filters
IF TicketID Provided THEN
    ADD to Query "AND TicketID = Input TicketID"
ENDIF

IF Status Provided THEN
    IF Status is Multi-Select THEN
        ADD to Query "AND Status IN (Selected Statuses)"
    ELSE
        ADD to Query "AND Status = Input Status"
```

Systems Analysis and Design Project

```
ENDIF
ENDIF

IF Category Provided THEN
    ADD to Query "AND Category = Input Category"
ENDIF

IF Date Range Provided THEN
    ADD to Query "AND ReportedAt BETWEEN StartDate AND EndDate"
ENDIF

IF Keyword Provided THEN
    ADD to Query "AND (Title LIKE '%Keyword%' OR Description LIKE '%Keyword%')"
ENDIF

// Execute Query
EXECUTE Query against Tickets Database (D2)
SET Tickets = Query Results

// Check if results found
IF Tickets is Empty THEN
    DISPLAY "No tickets found matching your criteria"
    EXIT Process
ENDIF

// Sort Results
IF User Specifies Sort Order THEN
    SORT Tickets by Specified Field (Priority DESC, ReportedAt DESC, etc.)
ELSE
    // Default sorting
    IF User Role = "Administrator" THEN
        SORT by Priority DESC, then ReportedAt DESC
    ELSIF User Role = "Technician" THEN
        SORT by AssignedAt DESC, then Priority DESC
    ELSE // Resident
        SORT by ReportedAt DESC
    ENDIF
ENDIF

// Pagination
SET ItemsPerPage = 20
CALCULATE TotalPages = CEILING(Tickets Count / ItemsPerPage)
GET Current Page from User (default = 1)
SET StartIndex = (Current Page - 1) * ItemsPerPage
SET EndIndex = StartIndex + ItemsPerPage
EXTRACT Page Items = Tickets[StartIndex to EndIndex]

// Prepare Display Data
FOR EACH Ticket in Page Items DO
    // Mark user's own tickets for residents
    IF User Role = "Resident" AND Ticket.RequesterID = Current UserID THEN
        SET Ticket.IsOwned = TRUE
    ENDIF
```

Systems Analysis and Design Project

```
// Hide sensitive information
IF User Role ≠ "Administrator" THEN
    REMOVE Ticket.RequesterPersonalInfo
    REMOVE Ticket.InternalNotes
ENDIF

// Format display fields
FORMAT Ticket.ReportedAt as "MMM DD, YYYY HH:MM"
FORMAT Ticket.Status with Status Badge/Color
FORMAT Ticket.Priority with Priority Icon
ENDFOR

// Display Results
DISPLAY Ticket List with:
- Ticket ID
- Title
- Category
- Location
- Status
- Priority
- Reported Date
- Assigned Technician (if applicable)
- "Your Ticket" badge (if applicable)

DISPLAY Pagination Controls
DISPLAY Search/Filter Panel
DISPLAY "Total Results: " + Tickets Count

// Real-time Updates
IF Real-time Mode Enabled THEN
    ESTABLISH WebSocket Connection
    LISTEN for Ticket Updates
    ON Ticket Update Received DO
        IF Updated Ticket matches Current Filters THEN
            UPDATE Ticket in Display List
            SHOW Notification "Ticket Updated"
        ENDIF
    ENDDO
ENDIF

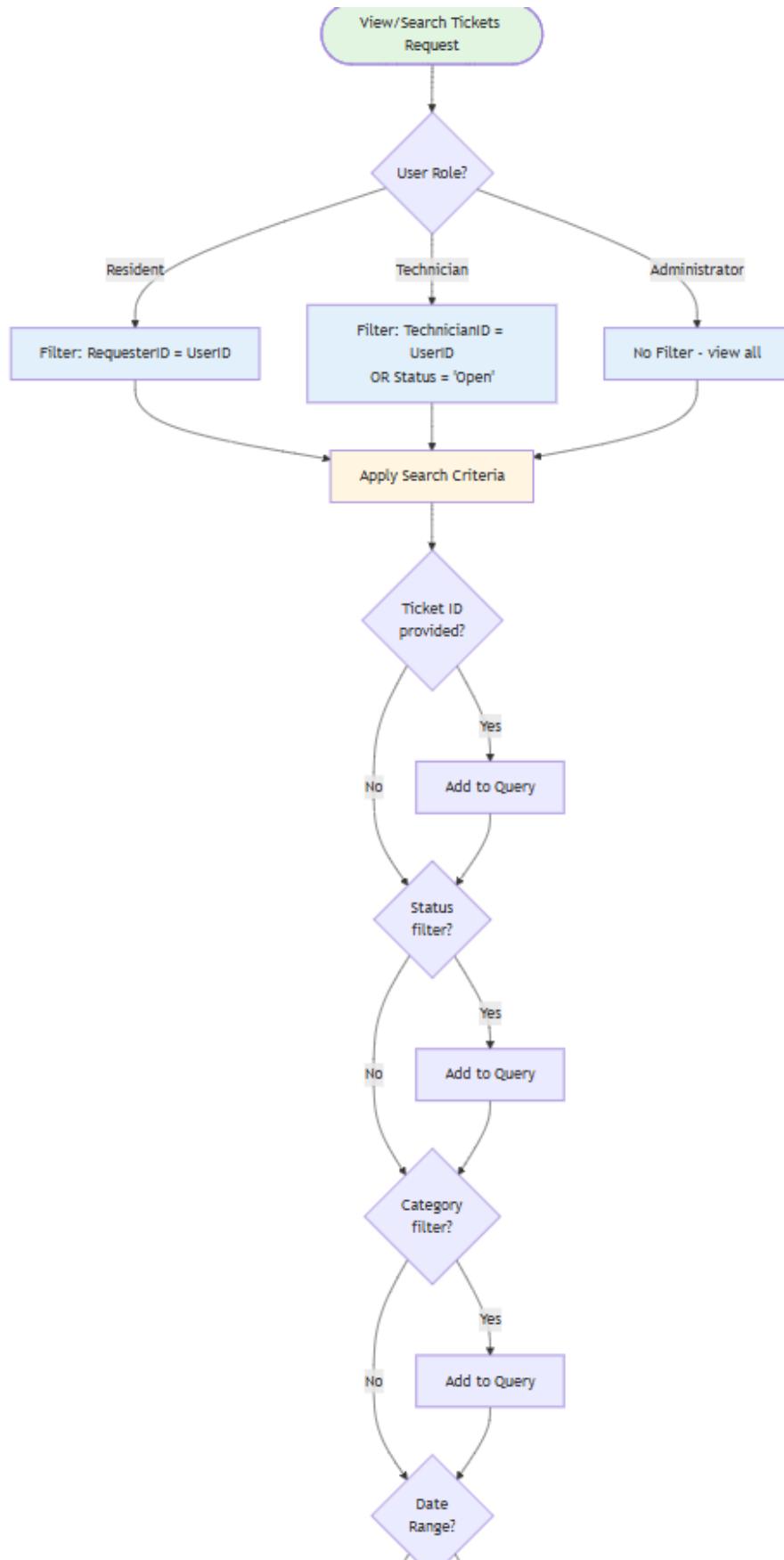
END Process
```

Refer to:

- FR-7 (View Reports Feed)
- FR-9 (Track Ticket Status)
- Table 65: Data Flow 14 Details
- Table 66: Data Flow 15 Details

Systems Analysis and Design Project

Decision Tree:



Systems Analysis and Design Project

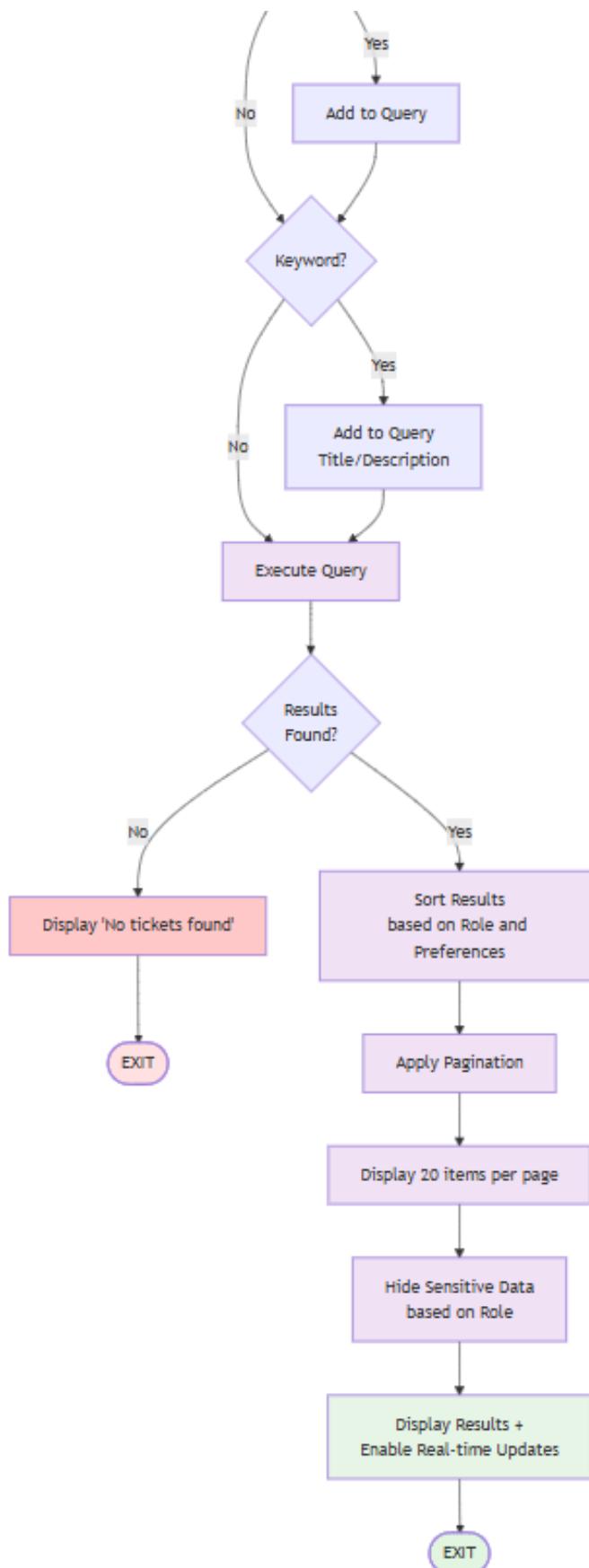


Figure 48: Decision Tree 2.3

Unresolved Issues: None

Systems Analysis and Design Project

Number: 2.3 **Name:** Update Ticket Status **Description:** Updates the status of maintenance tickets based on technician progress or administrator actions, validates status transitions, and triggers notifications.

Input Data Flow:

- Assignment/Priority Update (from Administrator)
- Status Update (from Technician)

Output Data Flow:

- Updated Ticket Status/Assignment/Priority (to Tickets Database D2)
- Status Change Notification (to Process 2.5)

Type of Process: Online Batch Manual

Subprogram/Function Name: updateTicketStatus()

Process Logic (Structured English):

```
BEGIN Process 2.3: Update Ticket Status

READ Update Request from User (Technician or Administrator)
    TicketID, New Status/Assignment/Priority, Optional Notes

RETRIEVE Current Ticket from Tickets Database (D2)
IF Ticket Not Found THEN
    DISPLAY "Ticket not found"
    EXIT Process
ENDIF

GET Current User Role and UserID
SET Current Status = Ticket.Status

// Authorization Check
IF User Role = "Technician" THEN
    IF Ticket.TechicianID ≠ Current UserID THEN
        DISPLAY "You are not authorized to update this ticket"
        EXIT Process
    ENDIF
ELSIF User Role = "Administrator" THEN
    // Administrators can update any ticket
ELSE
    DISPLAY "Unauthorized action"
    EXIT Process
ENDIF

// Process Administrator Updates
IF User Role = "Administrator" THEN
    IF Updating Assignment THEN
        READ New TechnicianID
```

Systems Analysis and Design Project

```
VALIDATE Technician Exists and Is Active
IF Technician Invalid THEN
    DISPLAY "Invalid technician selected"
    EXIT Process
ENDIF

// Check technician availability and workload
GET Technician Current Workload
GET Technician Availability Status
GET Technician Specialization

IF Availability Status = "On Break" OR Availability Status = "Busy" THEN
    WARN "Technician is currently " + Availability Status
    PROMPT "Proceed anyway?"
    IF User Declines THEN
        EXIT Process
    ENDIF
ENDIF

// Check expertise match
IF Ticket.Category NOT IN Technician.Specialization THEN
    WARN "Technician specialization doesn't match ticket category"
    PROMPT "Proceed anyway?"
    IF User Declines THEN
        EXIT Process
    ENDIF
ENDIF

SET Ticket.TechnicianID = New TechnicianID
SET Ticket.AssignedAt = CURRENT_DATETIME
SET Ticket.Status = "Assigned"

SEND Notification to New Technician (Process 2.5)
LOG "Ticket assigned to " + Technician.Name + " by " + Admin.Name
ENDIF

IF Updating Priority THEN
    READ New Priority Level

    VALIDATE Priority Level IN (Low, Medium, High, Urgent, Emergency)
    IF Priority Invalid THEN
        DISPLAY "Invalid priority level"
        EXIT Process
    ENDIF

    SET Old Priority = Ticket.Priority
    SET Ticket.Priority = New Priority

    // Recalculate estimated completion time
    CALCULATE Estimated Completion Time based on New Priority
    SET Ticket.EstimatedCompletionTime = Calculated Time

    IF New Priority > Old Priority THEN
        SET Ticket.IsUrgentFlag = TRUE
```

Systems Analysis and Design Project

```
    SEND Notification to Assigned Technician (Process 2.5)
ENDIF

LOG "Priority changed from " + Old Priority + " to " + New Priority
ENDIF

IF Manually Closing Ticket THEN
    SET Ticket.Status = "Closed"
    SET Ticket.ResolvedAt = CURRENT_DATETIME
    LOG "Ticket manually closed by administrator"
    SEND Notification to Requester (Process 2.5)
ENDIF
ENDIF

// Process Technician Status Updates
IF User Role = "Technician" THEN
    READ New Status, Optional Notes

    // Validate Status Transition
    CALL ValidateStatusTransition(Current Status, New Status)
    IF Transition Invalid THEN
        DISPLAY "Invalid status transition from " + Current Status + " to " + New
Status
        EXIT Process
    ENDIF

    IF New Status = "In Progress" THEN
        IF Current Status ≠ "Assigned" AND Current Status ≠ "On Hold" THEN
            DISPLAY "Cannot set to In Progress from current status"
            EXIT Process
        ENDIF

        SET Ticket.Status = "In Progress"
        SET Ticket.ActualStartTime = CURRENT_DATETIME

        CALCULATE Expected Completion = ActualStartTime + EstimatedDuration
        SET Ticket.ExpectedCompletionTime = Expected Completion

        LOG "Work started by " + Technician.Name
        SEND Notification to Requester (Process 2.5)
    ENDIF

    IF New Status = "On Hold" THEN
        IF Current Status ≠ "In Progress" THEN
            DISPLAY "Can only put In Progress tickets on hold"
            EXIT Process
        ENDIF

        REQUIRE On Hold Reason
        READ Hold Reason (Waiting for Parts, Requires Specialist, Resident Not
Available)
        IF Hold Reason Not Provided THEN
            DISPLAY "Please provide a reason for putting ticket on hold"
            EXIT Process
        ENDIF
    ENDIF
ENDIF
```

Systems Analysis and Design Project

```
ENDIF

SET Ticket.Status = "On Hold"
SET Ticket.OnHoldReason = Hold Reason
SET Ticket.OnHoldDate = CURRENT_DATETIME

LOG "Ticket placed on hold: " + Hold Reason
SEND Notification to Administrator and Requester (Process 2.5)
ENDIF

IF New Status = "Fixed" THEN
    IF Current Status ≠ "In Progress" THEN
        DISPLAY "Ticket must be In Progress before marking as Fixed"
        EXIT Process
    ENDIF

    REQUIRE Completion Evidence (handled by Process 2.4)
    // Technician must submit evidence before fixing
    CHECK if Evidence Submitted for This Ticket
    IF No Evidence THEN
        DISPLAY "Please upload completion evidence before marking as Fixed"
        REDIRECT to Evidence Upload (Process 2.4)
        EXIT Process
    ENDIF

    SET Ticket.Status = "Fixed"
    SET Ticket.ResolvedAt = CURRENT_DATETIME

    CALCULATE Actual Resolution Time = ResolvedAt - ActualStartTime
    SET Ticket.ActualResolutionTime = Actual Resolution Time

    // Update technician performance metrics
    UPDATE Technician.AverageResolutionTime
    UPDATE Technician.CompletionRate

    LOG "Ticket marked as Fixed by " + Technician.Name
    SEND Notification to Administrator and Requester for Review (Process 2.5)
ENDIF

IF New Status = "Closed" THEN
    IF Current Status ≠ "Fixed" THEN
        DISPLAY "Only Fixed tickets can be closed"
        EXIT Process
    ENDIF

    // Usually closed by Administrator after review
    DISPLAY "Ticket will be closed after administrator review"
    EXIT Process
ENDIF

IF Notes Provided THEN
    SET Ticket.UpdateNotes = Notes
ENDIF
ENDIF
```

Systems Analysis and Design Project

```
// Check for Overdue Escalation
IF Current Status = "In Progress" THEN
    IF CURRENT_DATETIME > Ticket.EstimatedCompletionTime THEN
        SET Ticket.IsUrgentFlag = TRUE
        INCREMENT Ticket.EscalationLevel
        SEND Escalation Notification to Administrator (Process 2.5)
        LOG "Ticket escalated due to overdue status"
    ENDIF
ENDIF

// Update Database
UPDATE Ticket Record in Tickets Database (D2)
SET Ticket.LastUpdated = CURRENT_DATETIME

// Trigger Notifications
SEND Status Change Notification (Process 2.5)

DISPLAY "Ticket updated successfully"

END Process
```

Subfunction: ValidateStatusTransition

```
FUNCTION ValidateStatusTransition(Current Status, New Status) RETURNS Boolean

// Valid Transitions Matrix
SET Valid Transitions = {
    "Open": ["Assigned"],
    "Assigned": ["In Progress", "Open"],
    "In Progress": ["On Hold", "Fixed"],
    "On Hold": ["In Progress"],
    "Fixed": ["Closed"],
    "Closed": []
}

IF New Status IN Valid Transitions[Current Status] THEN
    RETURN TRUE
ELSE
    RETURN FALSE
ENDIF

END FUNCTION
```

Refer to:

- FR-16 (Task Status Updates)
- FR-23 (Ticket Assignment Management)
- FR-24 (Ticket Priority Management)
- Table 70: Data Flow 19 Details
- Table 71: Data Flow 20 Details
- Table 72: Data Flow 21 Details

Systems Analysis and Design Project

Decision Table (Technician Status Updates):

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Current Status = Assigned	Y	N	N	N	N	N
Current Status = In Progress	N	Y	Y	Y	N	N
Current Status = On Hold	N	N	N	N	Y	N
Current Status = Fixed	N	N	N	N	N	Y
New Status = In Progress	Y	N	N	N	Y	N
New Status = On Hold	N	Y	N	N	N	N
New Status = Fixed	N	N	Y	N	N	N
Evidence Submitted	-	-	Y	N	-	-
Set Status = In Progress	X	-	-	-	X	-
Record Start Time	X	-	-	-	-	-
Set Status = On Hold	-	X	-	-	-	-
Require Hold Reason	-	X	-	-	-	-
Set Status = Fixed	-	-	X	-	-	-
Update Metrics	-	-	X	-	-	-
Display "Need Evidence"	-	-	-	X	-	-
Display "Invalid Transition"	-	-	-	-	-	X
Send Notifications	X	X	X	-	X	-

Unresolved Issues: None

Number: 2.4 **Name:** Manage Comments & Evidence **Description:** Allows residents to add comments to tickets and technicians to upload maintenance completion evidence (images, notes, documents).

Input Data Flow:

- Comment Submission (from Resident)
- Completion Evidence (from Technician)

Output Data Flow:

- Maintenance Completion Record (to Maintenance History D3)
- Updated Ticket with Evidence (to Tickets Database D2)

Type of Process: Online Batch Manual

Subprogram/Function Name: manageCommentsEvidence()

Process Logic (Structured English):

Systems Analysis and Design Project

BEGIN Process 2.4: Manage Comments & Evidence

READ Action Type (Add Comment OR Upload Evidence)

// =====

// RESIDENT: Add Comment to Ticket

// =====

IF Action = "Add Comment" THEN

 READ TicketID and Comment Text from Resident

 RETRIEVE Ticket from Tickets Database (D2)

 IF Ticket Not Found THEN

 DISPLAY "Ticket not found"

 EXIT Process

 ENDIF

 VERIFY Resident is Ticket Owner

 IF Ticket.RequesterID ≠ Current UserID THEN

 DISPLAY "You can only comment on your own tickets"

 EXIT Process

 ENDIF

 CHECK Ticket Status

 IF Ticket.Status = "Closed" THEN

 IF Ticket.ResidentEditableFlag = FALSE THEN

 DISPLAY "Cannot add comments to closed tickets"

 EXIT Process

 ENDIF

 ENDIF

 VALIDATE Comment Text

 IF Comment Text is Empty OR Length < 5 characters THEN

 DISPLAY "Comment must be at least 5 characters"

 EXIT Process

 ENDIF

 GENERATE CommentID

 CREATE Comment Record

 CommentID = Generated ID

 TicketID = Input TicketID

 UserID = Current UserID

 Content = Comment Text

 CreatedAt = CURRENT_DATETIME

 IsInternal = FALSE

 WRITE Comment to Database

 // Send notification to assigned technician (if any)

 IF Ticket.TechnicianID ≠ NULL THEN

 SEND Notification to Technician (Process 2.5)

 "New comment added to ticket " + TicketID

 ENDIF

 // Send notification to administrators

Systems Analysis and Design Project

```
SEND Notification to Administrators (Process 2.5)

DISPLAY "Comment added successfully"
ENDIF

// =====
// TECHNICIAN: Upload Maintenance Evidence
// =====
IF Action = "Upload Evidence" THEN
    READ TicketID, Images, Notes, Documents from Technician

    RETRIEVE Ticket from Tickets Database (D2)
    IF Ticket Not Found THEN
        DISPLAY "Ticket not found"
        EXIT Process
    ENDIF

    VERIFY Technician is Assigned to Ticket
    IF Ticket.TechnicianID ≠ Current UserID THEN
        DISPLAY "You can only upload evidence for your assigned tickets"
        EXIT Process
    ENDIF

    CHECK Ticket Status
    IF Ticket.Status ≠ "In Progress" AND Ticket.Status ≠ "Fixed" THEN
        DISPLAY "Evidence can only be uploaded for In Progress or Fixed tickets"
        EXIT Process
    ENDIF

    // Validate Images
    IF Images Provided THEN
        IF Number of Images > 10 THEN
            DISPLAY "Maximum 10 evidence images allowed"
            EXIT Process
        ENDIF

        FOR EACH Image DO
            VALIDATE File Type (JPEG, PNG)
            VALIDATE File Size (< 10MB per image)
            IF Validation Fails THEN
                DISPLAY "Invalid image format or size. Allowed: JPEG, PNG up to
10MB"
                EXIT Process
            ENDIF
        ENDFOR
    ENDIF

    // Validate Documents
    IF Documents Provided THEN
        FOR EACH Document DO
            VALIDATE File Type (PDF only)
            VALIDATE File Size (< 20MB)
            IF Validation Fails THEN
                DISPLAY "Invalid document. Only PDF files up to 20MB allowed"
            ENDIF
        ENDFOR
    ENDIF
```

Systems Analysis and Design Project

```
        EXIT Process
    ENDIF
    ENDFOR
ENDIF

// Validate that at least one piece of evidence is provided
IF Images is Empty AND Notes is Empty AND Documents is Empty THEN
    DISPLAY "Please provide at least one piece of evidence (image or notes)"
    EXIT Process
ENDIF

// Upload Images to AWS S3
IF Images Provided THEN
    FOR EACH Image DO
        COMPRESS Image (if size > 2MB, compress to maintain quality)
        GENERATE Unique Filename
            Format: evidence_<TicketID>_<Timestamp>_<Index>.<ext>

        UPLOAD to AWS S3
            Bucket: maintenance-evidence
            Path: /<Year>/<Month>/<TicketID>/

        STORE Image URL

        // Add caption if provided
        IF Caption Provided for Image THEN
            STORE Caption with Image URL
        ENDIF
    ENDFOR
ENDIF

// Upload Documents to AWS S3
IF Documents Provided THEN
    FOR EACH Document DO
        GENERATE Unique Filename
        UPLOAD to AWS S3
            Bucket: maintenance-documents
            Path: /<Year>/<Month>/<TicketID>/
        STORE Document URL
    ENDFOR
ENDIF

// Get Standard Completion Checklist for Category
RETRIEVE Completion Checklist Template
    based on Ticket.Category

IF Checklist Items Provided THEN
    FOR EACH Checklist Item DO
        VALIDATE Item is Checked or Has Note
        STORE Checklist Response
    ENDFOR
ENDIF

// Create Evidence Record
```

Systems Analysis and Design Project

```
GENERATE EvidenceID
CREATE Evidence Record
    EvidenceID = Generated ID
    TicketID = Input TicketID
    TechnicianID = Current UserID
    CompletionNotes = Input Notes
    ImagesURL = Uploaded Image URLs Array
    DocumentsURL = Uploaded Document URLs Array
    ChecklistData = Completed Checklist
    WorkPerformed = Notes.WorkPerformed
    PartsUsed = Notes.PartsUsed
    TimeSpent = Notes.TimeSpent
    FollowUpRequired = Notes.FollowUpRequired
    UploadedAt = CURRENT_DATETIME

    WRITE Evidence Record to Database

    // Update Ticket with Evidence Reference
    UPDATE Ticket in Tickets Database (D2)
        SET EvidenceID = Generated EvidenceID
        SET HasEvidence = TRUE

    // Store in Maintenance History
    IF Ticket.Status = "Fixed" THEN
        CREATE Maintenance History Record
            TicketID = Input TicketID
            TechnicianID = Current UserID
            LocationID = Ticket.LocationID
            Category = Ticket.Category
            ResolutionSummary = Notes
            CompletionDate = CURRENT_DATETIME
            EvidenceID = Generated EvidenceID
            ActualResolutionTime = Ticket.ActualResolutionTime

            WRITE to Maintenance History (D3)
    ENDIF

    // Send Notification
    SEND Notification to Requester (Process 2.5)
        "Maintenance work completed with evidence"
    SEND Notification to Administrators (Process 2.5)
        "Evidence uploaded for ticket " + TicketID

    DISPLAY "Evidence uploaded successfully"
    DISPLAY "Ticket ID: " + TicketID
    DISPLAY "Evidence ID: " + EvidenceID
ENDIF

END Process
```

Refer to:

- FR-17 (Maintenance Evidence Submission)
- FR-9.6 (Resident Comments)

Systems Analysis and Design Project

- Table 67: Data Flow 16 Details
- Table 68: Data Flow 17 Details
- Table 69: Data Flow 18 Details

Decision Table (Evidence Upload):

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Assigned to Technician	Y	Y	Y	Y	Y	N
Status = In Progress/Fixed	Y	Y	Y	Y	N	-
Images Provided	Y	Y	Y	N	-	-
Images Valid	Y	Y	N	-	-	-
Notes Provided	Y	N	-	Y	-	-
Upload Images	X	X	-	-	-	-
Store Evidence Record	X	X	-	X	-	-
Update Ticket	X	X	-	X	-	-
Update History (if Fixed)	X	X	-	X	-	-
Send Notifications	X	X	-	X	-	-
Display "Invalid images"	-	-	X	-	-	-
Display "Wrong status"	-	-	-	-	X	-
Display "Not authorized"	-	-	-	-	-	X

Unresolved Issues: None

Number: 2.5 **Name:** Send Notifications **Description:** Central notification dispatcher that sends real-time notifications via in-app, email, and SMS channels based on ticket events and status changes.

Input Data Flow:

- Notification Trigger (from Processes 2.1, 2.3, 2.4, 3.2, 4.1)

Output Data Flow:

- Notification (to Resident/Technician/Administrator)

Type of Process: Online Batch Manual

Subprogram/Function Name: sendNotification()

Process Logic (Structured English):

BEGIN Process 2.5: Send Notifications

READ Notification Trigger Data
TicketID, Event Type, RecipientRole, RecipientUserID

Systems Analysis and Design Project

```
RETRIEVE Ticket Details from Tickets Database (D2)
RETRIEVE Recipient User Data from User Database (D1)

// Determine Notification Type
SET Notification Type based on Event
IF Event = "Ticket Created" THEN
    SET Type = "Alert"
ELSIF Event = "Status Changed" THEN
    SET Type = "StatusChange"
ELSIF Event = "Ticket Assigned" THEN
    SET Type = "Assignment"
ELSIF Event = "Priority Changed" THEN
    SET Type = "PriorityChange"
ELSIF Event = "Comment Added" THEN
    SET Type = "StatusChange"
ELSIF Event = "Evidence Uploaded" THEN
    SET Type = "StatusChange"
ELSIF Event = "Ticket Overdue" THEN
    SET Type = "Alert"
ELSE
    SET Type = "System"
ENDIF

// Determine Priority Level
IF Event = "Ticket Overdue" OR
    Event = "Emergency Ticket" OR
    Event = "Account Locked" THEN
    SET Priority Level = "High" // Immediate delivery
ELSIF Event = "Status Changed" OR
    Event = "Priority Changed" OR
    Event = "Ticket Assigned" THEN
    SET Priority Level = "Medium" // Within 15 minutes
ELSE
    SET Priority Level = "Low" // Within 1 hour
ENDIF

// Check User Notification Preferences
RETRIEVE User Notification Preferences from User Database (D1)
IF Recipient.DoNotDisturb = TRUE THEN
    IF Priority Level ≠ "High" THEN
        QUEUE Notification for Later Delivery
        EXIT Process
    ENDIF
ENDIF

// Check if Notification Should Be Grouped
CHECK for Recent Similar Notifications (within last 10 minutes)
IF Similar Notifications Exist THEN
    GROUP Notifications Together
    UPDATE Existing Notification with Latest Status
ELSE
    CREATE New Notification
ENDIF
```

Systems Analysis and Design Project

```
// Build Notification Message
SET Message = ""
IF Event = "Ticket Created" THEN
    SET Message = "New maintenance ticket created: " + Ticket.Title
ELSIF Event = "Status Changed" THEN
    SET Message = "Ticket " + TicketID + " status changed to " + Ticket.Status
ELSIF Event = "Ticket Assigned" THEN
    SET Message = "New ticket assigned to you: " + Ticket.Title + " (Priority: " +
Ticket.Priority + ")"
ELSIF Event = "Priority Changed" THEN
    SET Message = "Ticket " + TicketID + " priority changed to " + Ticket.Priority
ELSIF Event = "Comment Added" THEN
    SET Message = "New comment added to your ticket " + TicketID
ELSIF Event = "Evidence Uploaded" THEN
    SET Message = "Maintenance evidence uploaded for ticket " + TicketID
ELSIF Event = "Ticket Overdue" THEN
    SET Message = "ALERT: Ticket " + TicketID + " is overdue"
ENDIF

GENERATE NotificationID
CREATE Notification Record
    NotificationID = Generated ID
    UserID = RecipientUserID
    RelatedTicketID = TicketID
    NotificationType = Type
    Message = Message
    Priority = Priority Level
    IsRead = FALSE
    CreatedAt = CURRENT_DATETIME
    ExpirationDate = CURRENT_DATETIME + 90 days

WRITE Notification to Database

// Send via Multiple Channels
// 1. In-App Notification
IF Recipient.Preferences.InAppEnabled = TRUE THEN
    SEND via WebSocket to Active User Sessions
    UPDATE Notification Badge Count
ENDIF

// 2. Email Notification
IF Recipient.Preferences.EmailEnabled = TRUE THEN
    IF Priority Level = "High" OR
        Recipient.Preferences.EmailForAllUpdates = TRUE THEN

        COMPOSE Email
            Subject = "SALLEHA: " + Message
            Body = Build HTML Email Template with:
                - Ticket Details
                - Status/Update Information
                - Direct Link to View Ticket
                - User Preferences Link
    ENDIF
ENDIF
```

Systems Analysis and Design Project

```
SEND Email via Nodemailer SMTP
LOG Email Sent
ENDIF
ENDIF

// 3. SMS Notification (Optional - Only for Urgent)
IF Recipient.Preferences.SMSEnabled = TRUE THEN
    IF Priority Level = "High" OR Ticket.Priority = "Emergency" THEN
        IF Recipient.PhoneNumber ≠ NULL THEN
            COMPOSE SMS Message (max 160 chars)
            "SALLEHA ALERT: " + Message + " View: " + Short URL

            SEND SMS via Twilio API
            LOG SMS Sent
        ENDIF
    ENDIF
ENDIF

// 4. Push Notification (Mobile App)
IF Recipient Has Active Mobile Session THEN
    COMPOSE Push Notification
    Title = "SALLEHA Maintenance"
    Body = Message
    Data = {TicketID, Type, DeepLink}

    SEND via Firebase Cloud Messaging (FCM)
    LOG Push Notification Sent
ENDIF

DISPLAY "Notification sent successfully"

END Process
```

Refer to:

- FR-10 (Status Change Notifications)
- FR-18 (Technician Notifications)
- FR-25 (Administrator Notifications)
- Table 73: Data Flow 22 Details

Decision Table:

Systems Analysis and Design Project

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Priority Level = High	Y	Y	N	N	N
Do Not Disturb Active	Y	N	Y	N	N
In-App Enabled	-	Y	-	Y	Y
Email Enabled	-	Y	-	Y	N
SMS Enabled	-	Y	-	N	-
Send Immediately	X	X	-	-	-
Queue for Later	-	-	X	-	-
Send In-App	X	X	-	X	X
Send Email	X	X	-	X	-
Send SMS	X	X	-	-	-
Send Push Notification	X	X	-	X	X
Group Notifications	-	X	-	X	X

Unresolved Issues: None

4.3.3. Fragment 3: Assignment & Task Management Processes

Number: 3.1 **Name:** View Pending/Unassigned Tickets **Description:** Displays all tickets that haven't been assigned to technicians, sorted by priority and submission time for administrator review.

Input Data Flow:

- View Pending Tickets Request (from Administrator)

Output Data Flow:

- Pending Tickets List (from Tickets Database D2 to Administrator)

Type of Process: Online Batch Manual

Subprogram/Function Name: viewPendingTickets()

Process Logic (Structured English):

BEGIN Process 3.1: View Pending/Unassigned Tickets

```

VERIFY User is Administrator
IF User Role ≠ "Administrator" THEN
    DISPLAY "Unauthorized access"
    EXIT Process
ENDIF

```

```

READ Filter Options from Administrator (Optional)
    Priority Filter, Category Filter, Date Range, Location Filter

```

```
// Query Pending Tickets
```

Systems Analysis and Design Project

```
QUERY Tickets Database (D2) WHERE:
    Status IN ("Open", "Assigned") AND
    (TechnicianID IS NULL OR Status = "Open")

// Apply Optional Filters
IF Priority Filter Provided THEN
    FILTER Results by Priority IN Selected Priorities
ENDIF

IF Category Filter Provided THEN
    FILTER Results by Category = Selected Category
ENDIF

IF Date Range Provided THEN
    FILTER Results by ReportedAt BETWEEN StartDate AND EndDate
ENDIF

IF Location Filter Provided THEN
    FILTER Results by LocationID = Selected Location
ENDIF

SET Pending Tickets = Query Results

IF Pending Tickets is Empty THEN
    DISPLAY "No pending tickets found"
    EXIT Process
ENDIF

// Sort by Priority and Time
SORT Pending Tickets by:
    1. Priority DESC (Emergency > Urgent > High > Medium > Low)
    2. IsUrgentFlag DESC
    3. ReportedAt ASC (oldest first)

// Calculate Wait Times
FOR EACH Ticket in Pending Tickets DO
    CALCULATE Wait Time = CURRENT_DATETIME - Ticket.ReportedAt
    SET Ticket.WaitTime = Wait Time

    // Highlight overdue tickets
    IF Wait Time > Expected Response Time for Priority THEN
        SET Ticket.IsOverdue = TRUE
    ENDIF

    // Get available technicians for this ticket
    QUERY Available Technicians WHERE:
        AvailabilityStatus = "Available" AND
        Specialization MATCHES Ticket.Category

    SET Ticket.AvailableTechnicians = Query Results Count
ENDFOR

// Display Statistics
CALCULATE Total Pending = Pending Tickets Count
```

Systems Analysis and Design Project

```
CALCULATE Total Overdue = Count of Tickets WHERE IsOverdue = TRUE
CALCULATE Avg Wait Time = AVERAGE(All Tickets WaitTime)

DISPLAY Dashboard Header
    "Total Pending: " + Total Pending
    "Overdue: " + Total Overdue
    "Average Wait Time: " + Avg Wait Time

// Display Ticket List
FOR EACH Ticket in Pending Tickets DO
    DISPLAY Ticket Row
        TicketID (Clickable)
        Title
        Category
        Location
        Priority (Color-coded badge)
        Reported Time (Relative: "2 hours ago")
        Wait Time
        Available Technicians Count
        "Assign" Button
        "View Details" Link

    IF Ticket.IsOverdue THEN
        HIGHLIGHT Row in Red
        SHOW "OVERDUE" Badge
    ENDIF
ENDFOR

// Provide Quick Actions
DISPLAY Quick Action Buttons
    "Auto-Assign All"
    "Bulk Assign by Category"
    "Refresh List"
    "Export to Excel"

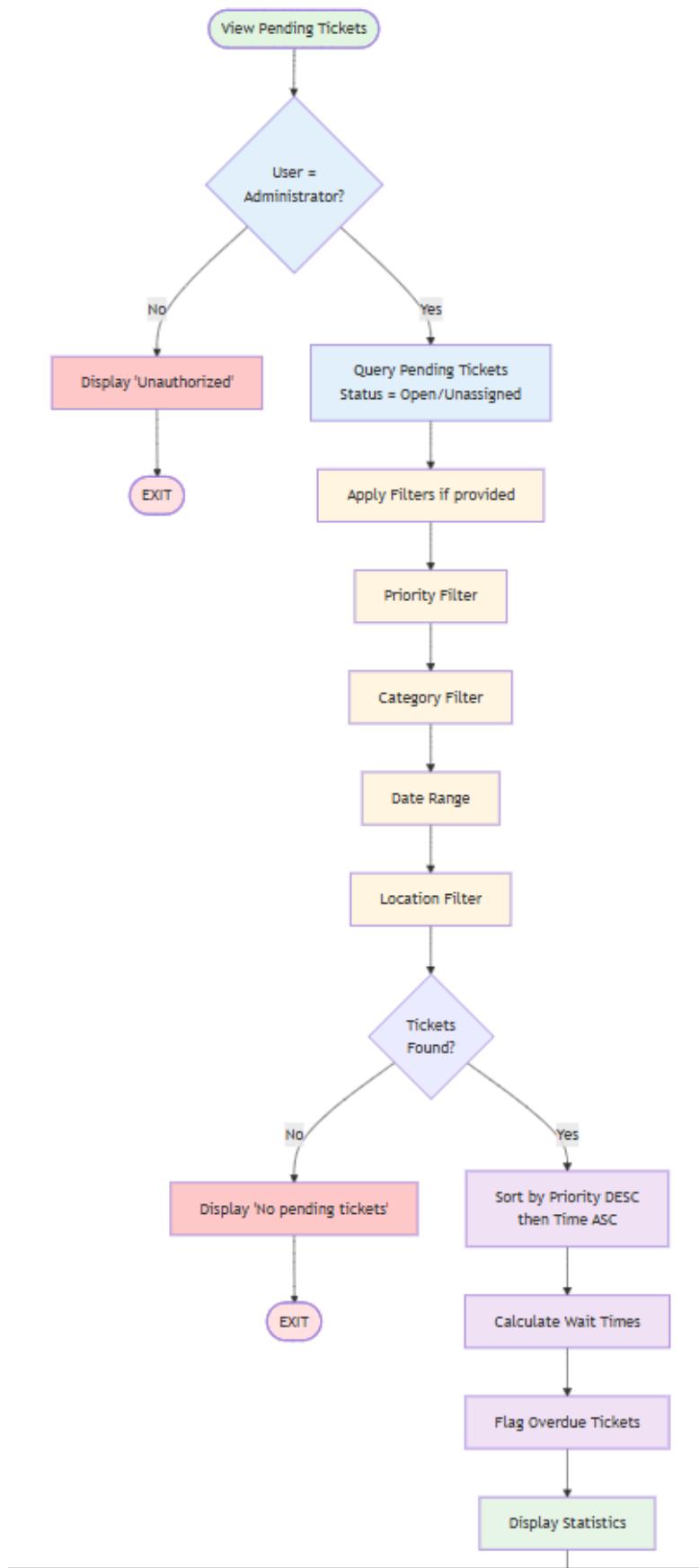
END Process
```

Refer to:

- FR-23 (Ticket Assignment Management)
- Table 74: Data Flow 23 Details
- Table 75: Data Flow 24 Details

Systems Analysis and Design Project

Decision Tree:



Systems Analysis and Design Project

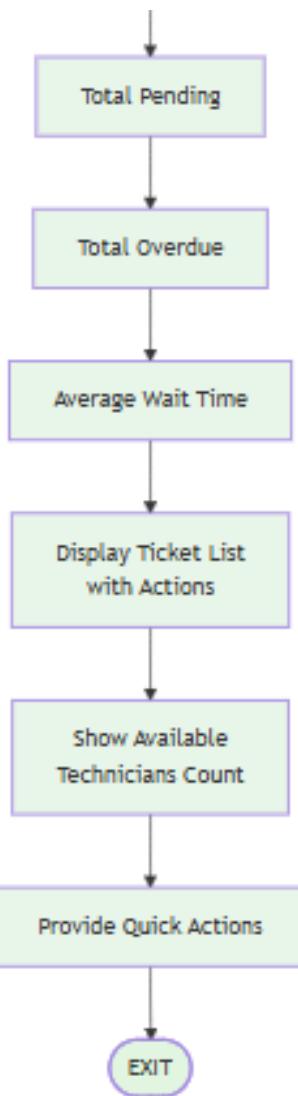


Figure 50: Decision Tree 3.2

Unresolved Issues: None

Number: 3.2 **Name:** Assign Ticket & Set Priority **Description:** Assigns maintenance tickets to technicians based on availability, expertise, and workload. Sets or updates ticket priority levels.

Input Data Flow:

- Assignment Decision (from Administrator)
 - TicketID
 - TechnicianID
 - Priority Level

Output Data Flow:

- Assignment & Priority Update (to Tickets Database D2)
- Assigned Task Notification (to Process 3.3 → Technician)

Systems Analysis and Design Project

Type of Process: Online Batch Manual

Subprogram/Function Name: assignTicketAndSetPriority()

Process Logic (Structured English):

```
BEGIN Process 3.2: Assign Ticket & Set Priority

VERIFY User is Administrator
IF User Role ≠ "Administrator" THEN
    DISPLAY "Unauthorized access"
    EXIT Process
ENDIF

READ Assignment Data from Administrator
    TicketID, TechnicianID, Priority Level (Optional)

RETRIEVE Ticket from Tickets Database (D2)
IF Ticket Not Found THEN
    DISPLAY "Ticket not found"
    EXIT Process
ENDIF

RETRIEVE Technician from User Database (D1)
IF Technician Not Found OR Technician.Role ≠ "Technician" THEN
    DISPLAY "Invalid technician selected"
    EXIT Process
ENDIF

// Check Ticket is Assignable
IF Ticket.Status = "Closed" THEN
    DISPLAY "Cannot assign closed tickets"
    EXIT Process
ENDIF

// Validate Technician Status
IF Technician.AccountStatus ≠ "Active" THEN
    DISPLAY "Selected technician account is not active"
    EXIT Process
ENDIF

// Check Availability
GET Technician Current Availability
IF Technician.AvailabilityStatus = "On Break" THEN
    WARN "Technician is currently on break"
    PROMPT "Assign anyway?"
    IF Administrator Declines THEN
        EXIT Process
    ENDIF
ENDIF

// Check Workload
```

Systems Analysis and Design Project

```
GET Technician Current Assigned Tasks Count
CALCULATE Current Workload = Count of Tickets WHERE:
    TechnicianID = Selected Technician AND
    Status IN ("Assigned", "In Progress")

IF Current Workload >= 10 THEN
    WARN "Technician already has " + Current Workload + " active tasks"
    DISPLAY "Average workload: 5 tasks"
    PROMPT "Assign anyway?"
    IF Administrator Declines THEN
        SUGGEST Alternative Technicians with Lower Workload
        EXIT Process
    ENDIF
ENDIF

// Check Expertise Match
GET Technician Specialization
IF Ticket.Category NOT IN Technician.Specialization THEN
    WARN "Technician specialization: " + Technician.Specialization
    WARN "Ticket category: " + Ticket.Category
    WARN "Specializations don't match"
    PROMPT "Assign anyway?"
    IF Administrator Declines THEN
        // Suggest better matched technicians
        QUERY Technicians WHERE:
            Specialization CONTAINS Ticket.Category AND
            AvailabilityStatus = "Available"
        DISPLAY "Suggested Technicians: " + Query Results
        EXIT Process
    ENDIF
ENDIF

// Check Location Proximity (if WorkZone defined)
IF Technician.WorkZone ≠ NULL THEN
    GET Ticket Location Zone
    IF Technician.WorkZone ≠ Ticket.Location.Zone THEN
        INFO "Technician's work zone: " + Technician.WorkZone
        INFO "Ticket location zone: " + Ticket.Location.Zone
        PROMPT "Proceed with cross-zone assignment?"
        IF Administrator Declines THEN
            EXIT Process
        ENDIF
    ENDIF
ENDIF

// Update Priority (if provided)
IF Priority Level Provided THEN
    VALIDATE Priority IN (Low, Medium, High, Urgent, Emergency)
    IF Priority Invalid THEN
        DISPLAY "Invalid priority level"
        EXIT Process
    ENDIF

    SET Old Priority = Ticket.Priority
```

Systems Analysis and Design Project

```
SET Ticket.Priority = New Priority Level

// Recalculate estimated times
CALCULATE Estimated Response Time based on New Priority
CALCULATE Estimated Completion Time based on New Priority

IF New Priority > Old Priority THEN
    SET Ticket.IsUrgentFlag = TRUE
ENDIF

LOG "Priority changed from " + Old Priority + " to " + New Priority + " by " +
Admin.Name
ENDIF

// Set Expected Completion Instructions (Optional)
IF Administrator Provides Special Instructions THEN
    SET Ticket.AdminNotes = Special Instructions
    SET Ticket.RequiredTools = Specified Tools
    SET Ticket.RequiredParts = Specified Parts
ENDIF

// Perform Assignment
IF Ticket Already Assigned to Different Technician THEN
    SET Old TechnicianID = Ticket.TechnicianID
    LOG "Ticket reassigned from Technician " + Old TechnicianID + " to " +
TechnicianID

    // Notify old technician of reassignment
    SEND Notification to Old Technician (Process 2.5)
    "Ticket " + TicketID + " has been reassigned"
ENDIF

SET Ticket.TechnicianID = Selected TechnicianID
SET Ticket.AssignedAt = CURRENT_DATETIME
SET Ticket.Status = "Assigned"

// Calculate Expected Completion Based on Priority and Historical Data
QUERY Historical Average Resolution Time WHERE:
    Category = Ticket.Category AND
    Priority = Ticket.Priority

IF Historical Data Available THEN
    SET Estimated Duration = Historical Average
ELSE
    // Use default estimates
    IF Priority = "Emergency" THEN
        SET Estimated Duration = 1 hour
    ELSIF Priority = "Urgent" THEN
        SET Estimated Duration = 4 hours
    ELSIF Priority = "High" THEN
        SET Estimated Duration = 1 day
    ELSIF Priority = "Medium" THEN
        SET Estimated Duration = 3 days
    ELSE // Low
```

Systems Analysis and Design Project

```
SET Estimated Duration = 7 days
ENDIF
ENDIF

SET Ticket.ExpectedCompletionTime = CURRENT_DATETIME + Estimated Duration

// Update Database
UPDATE Ticket in Tickets Database (D2)
SET Ticket.LastUpdated = CURRENT_DATETIME

// Maintain Assignment History
CREATE Assignment History Record
    TicketID = TicketID
    TechnicianID = TechnicianID
    AssignedBy = Administrator.UserID
    AssignedAt = CURRENT_DATETIME
    Priority = Ticket.Priority
    Reasoning = Assignment Notes

WRITE to Assignment History Table

// Send Notification to Technician
TRIGGER Process 3.3 (Notify Technician)
    TicketID, TechnicianID, Priority

// Send Notification to Requester
SEND Notification to Ticket Requester (Process 2.5)
    "Your ticket has been assigned to a technician"

DISPLAY "Ticket " + TicketID + " successfully assigned to " + Technician.FullName

END Process
```

Refer to:

- FR-23 (Ticket Assignment Management)
- FR-24 (Ticket Priority Management)
- Table 76: Data Flow 25 Details
- Table 77: Data Flow 26 Details

Decision Table:

Systems Analysis and Design Project

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Technician Active	Y	Y	Y	Y	Y	N
Ticket Assignable	Y	Y	Y	Y	N	-
Workload < 10 Tasks	Y	Y	Y	N	-	-
Expertise Matches	Y	Y	N	-	-	-
Admin Confirms Mismatch	-	-	Y	-	-	-
Zone Matches (if defined)	Y	N	-	-	-	-
Actions	Assign Ticket	X	X	X	X	-
-	Update Priority	X	X	X	X	-
-	Send Notification	X	X	X	X	-
-	Log Assignment	X	X	X	X	-
-	Warn Zone Mismatch	-	X	-	-	-
-	Warn Expertise Mismatch	-	-	X	-	-
-	Warn High Workload	-	-	-	X	-
-	Display "Not Assignable"	-	-	-	-	X
-	Display "Invalid Technician"	-	-	-	-	-
X						

Unresolved Issues: None

Number: 3.3 **Name:** Notify Technician **Description:** Sends notification to technician when a new task is assigned, including ticket details and priority information.

Input Data Flow:

- Assignment Notification Trigger (from Process 3.2)

Output Data Flow:

- Assigned Task Notification (to Technician)

Type of Process: Online Batch Manual

Subprogram/Function Name: notifyTechnicianAssignment()

Process Logic (Structured English):

BEGIN Process 3.3: Notify Technician

Systems Analysis and Design Project

```
READ Notification Trigger Data
    TicketID, TechnicianID, Priority

RETRIEVE Ticket Details from Tickets Database (D2)
RETRIEVE Technician Data from User Database (D1)

// Build Notification Message
SET Message = "New ticket assigned: " + Ticket.Title
SET Details = {
    "Ticket ID": TicketID,
    "Category": Ticket.Category,
    "Location": Ticket.Location.BuildingName + " - " + Ticket.Location.RoomNumber,
    "Priority": Ticket.Priority,
    "Reported": Ticket.ReportedAt (Relative Time),
    "Expected Completion": Ticket.ExpectedCompletionTime
}

IF Ticket.AdminNotes ≠ NULL THEN
    ADD "Special Instructions: " + Ticket.AdminNotes to Details
ENDIF

IF Ticket.RequiredTools ≠ NULL THEN
    ADD "Required Tools: " + Ticket.RequiredTools to Details
ENDIF

// Determine Notification Priority
IF Ticket.Priority IN ("Emergency", "Urgent") THEN
    SET Notification Priority = "High" // Immediate
ELSIF Ticket.Priority = "High" THEN
    SET Notification Priority = "Medium" // Within 15 min
ELSE
    SET Notification Priority = "Low" // Within 1 hour
ENDIF

// Create Notification Record
GENERATE NotificationID
CREATE Notification
    NotificationID = Generated ID
    UserID = TechnicianID
    RelatedTicketID = TicketID
    NotificationType = "Assignment"
    Message = Message
    Details = Details JSON
    Priority = Notification Priority
    IsRead = FALSE
    CreatedAt = CURRENT_DATETIME
    ExpirationDate = CURRENT_DATETIME + 90 days

WRITE to Notifications Database

// Send Multi-Channel Notifications
// 1. In-App Push Notification
SEND Real-time Notification via WebSocket
    IF Technician Has Active Session THEN
```

Systems Analysis and Design Project

```
PUSH Notification to Dashboard
UPDATE Notification Badge
PLAY Notification Sound (if Priority = High)
ENDIF

// 2. Email Notification
IF Technician.Preferences.EmailNotifications = TRUE THEN
    COMPOSE Email
        To: Technician.Email
        Subject: "[SALLEHA] New Task Assigned - Priority: " + Priority
        Body: HTML Template with:
            - Ticket Summary
            - Location and Category
            - Priority Badge
            - Expected Completion Time
            - "View Ticket" Button (Deep Link)
            - "Accept Task" Quick Action Link

    SEND via Email Service
ENDIF

// 3. SMS (Only for Urgent/Emergency)
IF Ticket.Priority IN ("Emergency", "Urgent") THEN
    IF Technician.Preferences.SMSNotifications = TRUE THEN
        IF Technician.PhoneNumber ≠ NULL THEN
            COMPOSE SMS (max 160 chars)
                "URGENT: New task " + TicketID + " at " + Location + ". Priority: "
+ Priority + ". View: " + Short URL

            SEND via SMS Service (Twilio)
        ENDIF
    ENDIF
ENDIF

// 4. Mobile Push Notification
IF Technician Has Mobile App Installed THEN
    COMPOSE Push Notification
        Title: "New Task Assigned"
        Body: Ticket.Title + " (" + Priority + ")"
        Badge: Increment Unread Count
        Sound: (Priority = High ? "urgent.mp3" : "default.mp3")
        Data: {
            type: "assignment",
            ticketID: TicketID,
            priority: Priority,
            deepLink: "salleha://tickets/" + TicketID
        }

    SEND via Firebase Cloud Messaging (FCM)
ENDIF

LOG "Assignment notification sent to Technician " + TechnicianID + " for Ticket " +
TicketID
```

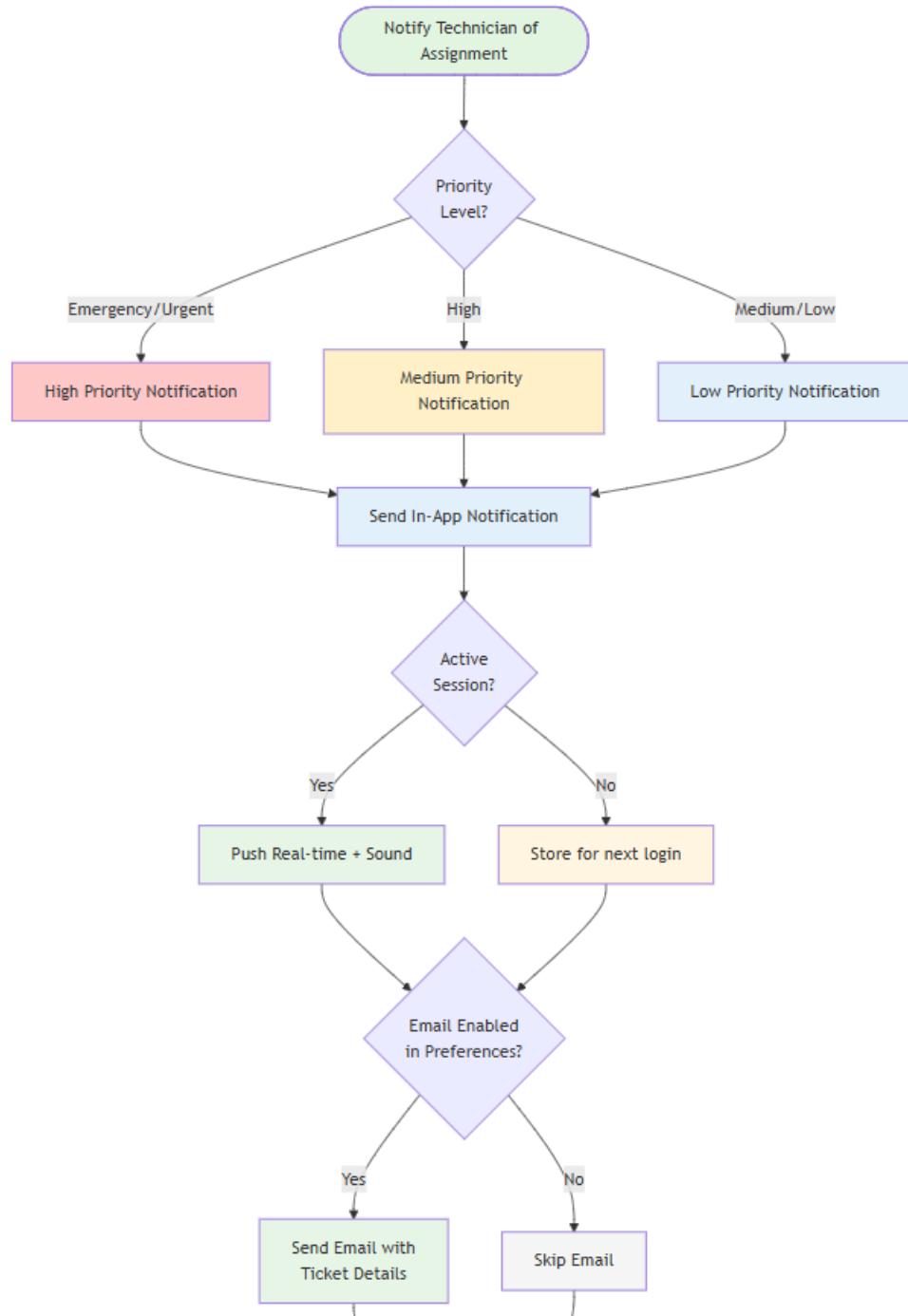
Systems Analysis and Design Project

END Process

Refer to:

- FR-18 (Technician Notifications)
- Table 78: Data Flow 27 Details

Decision Tree:



Systems Analysis and Design Project

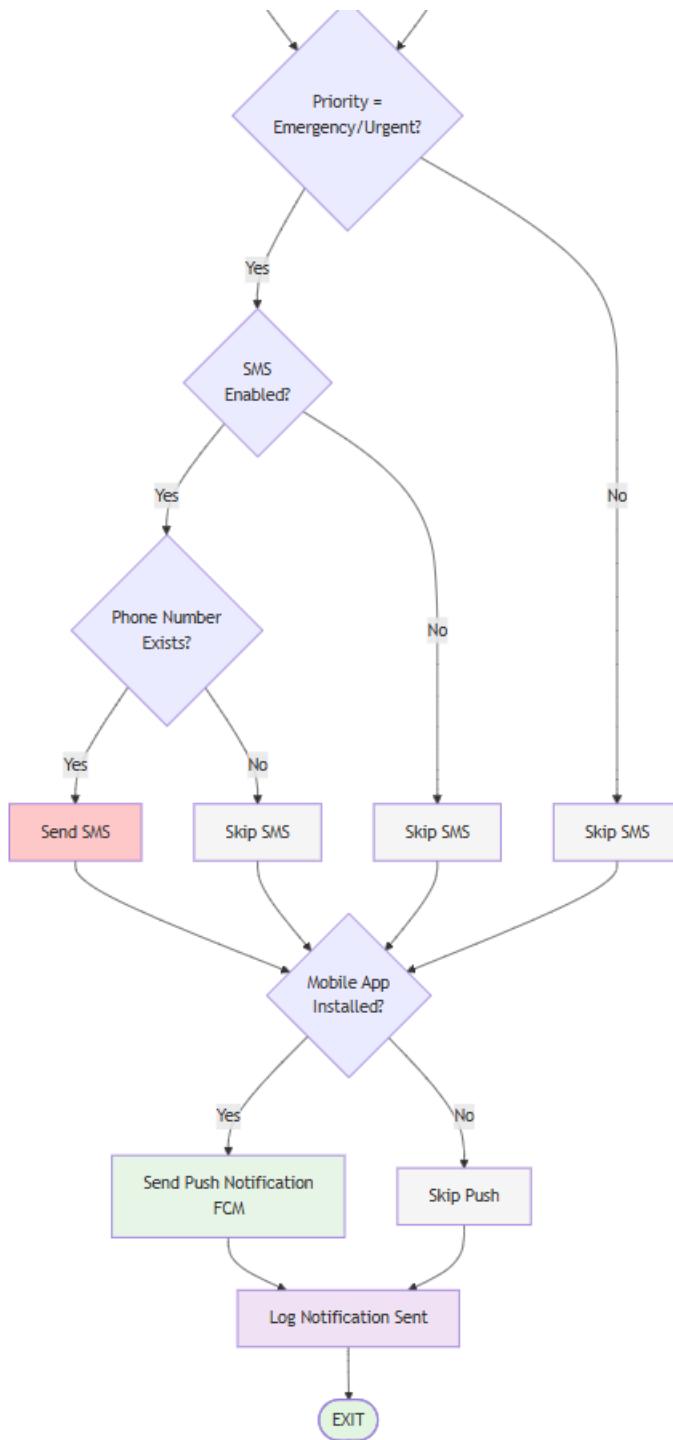


Figure 52: Decision Tree 3.3

Unresolved Issues: None

Number: 3.4 **Name:** Update Assignment Status **Description:** Updates ticket assignment status based on technician response (accept or decline), handles reassignment if declined.

Input Data Flow:

- Assignment Response (from Technician)
 - TicketID
 - Response (Accept/Decline)

Systems Analysis and Design Project

- Decline Reason (if declining)

Output Data Flow:

- Assignment Status Update (to Tickets Database D2)
- Reassignment Trigger (if declined)

Type of Process: Online Batch Manual

Subprogram/Function Name: updateAssignmentStatus()

Process Logic (Structured English):

```
BEGIN Process 3.4: Update Assignment Status

READ Assignment Response from Technician
    TicketID, Response Status, Optional Decline Reason

RETRIEVE Ticket from Tickets Database (D2)
IF Ticket Not Found THEN
    DISPLAY "Ticket not found"
    EXIT Process
ENDIF

VERIFY Technician is Assigned to Ticket
IF Ticket.TechnicianID ≠ Current UserID THEN
    DISPLAY "You are not assigned to this ticket"
    EXIT Process
ENDIF

CHECK Ticket Current Status
IF Ticket.Status ≠ "Assigned" THEN
    DISPLAY "Ticket cannot be accepted/declined in current status"
    EXIT Process
ENDIF

// =====
// TECHNICIAN ACCEPTS TASK
// =====
IF Response Status = "Accept" THEN
    SET Ticket.Status = "Accepted"
    SET Ticket.AcceptedAt = CURRENT_DATETIME

    UPDATE Ticket in Tickets Database (D2)

    LOG "Ticket " + TicketID + " accepted by Technician " + Technician.Name

    // Send Notification to Requester
    SEND Notification to Ticket Requester (Process 2.5)
        "Your ticket has been accepted by a technician"

    // Send Notification to Administrator
```

Systems Analysis and Design Project

```
SEND Notification to Administrators (Process 2.5)
    "Ticket " + TicketID + " accepted by " + Technician.Name

DISPLAY "Task accepted successfully"
REDIRECT to Ticket Details Page
ENDIF

// =====
// TECHNICIAN DECLINES TASK
// =====

IF Response Status = "Decline" THEN
    // Require Decline Reason
    IF Decline Reason is Empty THEN
        DISPLAY "Please provide a reason for declining"
        PROMPT Reason Selection:
            - "Outside my specialization"
            - "Current workload too high"
            - "Lack required equipment/parts"
            - "Urgent personal matter"
            - "Other (specify)"
    READ Decline Reason

    IF Decline Reason is Still Empty THEN
        DISPLAY "Reason is required to decline task"
        EXIT Process
    ENDIF
ENDIF

// Log Decline
CREATE Decline Record
    TicketID = TicketID
    TechnicianID = Current UserID
    DeclinedAt = CURRENT_DATETIME
    DeclineReason = Decline Reason

WRITE to Decline History

// Clear Assignment
SET Old TechnicianID = Ticket.TechnicianID
SET Ticket.TechnicianID = NULL
SET Ticket.AssignedAt = NULL
SET Ticket.Status = "Open"
SET Ticket.DeclineCount = Ticket.DeclineCount + 1

UPDATE Ticket in Tickets Database (D2)

LOG "Ticket " + TicketID + " declined by Technician " + Old TechnicianID + ".
Reason: " + Decline Reason

// If ticket declined multiple times, escalate
IF Ticket.DeclineCount >= 3 THEN
    SET Ticket.IsUrgentFlag = TRUE
    INCREMENT Ticket.EscalationLevel
    SEND Urgent Notification to Administrator (Process 2.5)
```

Systems Analysis and Design Project

```
"ALERT: Ticket " + TicketID + " declined " + Ticket.DeclineCount + "
times"
ENDIF

// Auto-Reassign to Next Available Technician
QUERY Available Technicians WHERE:
    AvailabilityStatus = "Available" AND
    Specialization MATCHES Ticket.Category AND
    UserID ≠ Old TechnicianID // Exclude technician who declined

ORDER BY:
    1. SkillLevel DESC
    2. Current Workload ASC
    3. CustomerSatisfactionRating DESC

LIMIT 1

IF Next Technician Found THEN
    // Auto-assign
    SET Ticket.TechnicianID = Next Technician.UserID
    SET Ticket.AssignedAt = CURRENT_DATETIME
    SET Ticket.Status = "Assigned"
    SET Ticket.AutoAssigned = TRUE

    UPDATE Ticket in Tickets Database (D2)

    // Notify new technician
    TRIGGER Process 3.3 (Notify Technician)
        TicketID, Next Technician.UserID, Ticket.Priority

    // Notify administrator
    SEND Notification to Administrator (Process 2.5)
        "Ticket " + TicketID + " auto-reassigned to " + Next Technician.Name +
    " after decline"

    DISPLAY "Task declined. Ticket has been reassigned to another technician."
ELSE
    // No available technician found
    SEND Urgent Notification to Administrator (Process 2.5)
        "ALERT: Ticket " + TicketID + " declined but no available technician
found for manual reassignment"

    DISPLAY "Task declined. Administrator will manually reassign this ticket."
ENDIF
ENDIF

END Process
```

Refer to:

- FR-15 (Accept Maintenance Requests)
- Table 79: Data Flow 28 Details
- Table 80: Data Flow 29 Details

Systems Analysis and Design Project

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Response = Accept	Y	N	N	N	N
Response = Decline	N	Y	Y	Y	N
Decline Reason Provided	-	Y	Y	Y	-
Next Technician Available	-	Y	Y	N	-
Decline Count >= 3	-	N	Y	-	-
Set Status = Accepted	X	-	-	-	-
Notify Requester + Admin	X	-	-	-	-
Record Decline	-	X	X	X	-
Clear Assignment	-	X	X	X	-
Auto-Reassign	-	X	X	-	-
Escalate (Urgent Flag)	-	-	X	-	-
Alert Admin (No Technician)	-	-	-	X	-
Display "Reason Required"	-	-	-	-	X

Unresolved Issues: None

4.3.4. Fragment 4: Analytics & Reporting Processes

Number: 4.1 **Name:** Record Maintenance Completion **Description:** Records the completion of maintenance work, validates evidence submission, updates ticket status, and calculates resolution metrics.

Input Data Flow:

- Completion Data (from Technician)
 - TicketID
 - Completion Notes
 - Completion Time
 - Evidence Reference

Output Data Flow:

- Maintenance Completion Record (to Process 4.2 → Maintenance History D3)
- Ticket Final Status Update (to Tickets Database D2)

Type of Process: Online Batch Manual

Subprogram/Function Name: recordMaintenanceCompletion()

Process Logic (Structured English):

BEGIN Process 4.1: Record Maintenance Completion

Systems Analysis and Design Project

```
READ Completion Data from Technician
    TicketID, Completion Notes, Work Summary, Parts Used, Time Spent

RETRIEVE Ticket from Tickets Database (D2)
IF Ticket Not Found THEN
    DISPLAY "Ticket not found"
    EXIT Process
ENDIF

VERIFY Technician Ownership
IF Ticket.TechnicianID ≠ Current UserID THEN
    DISPLAY "You are not authorized to complete this ticket"
    EXIT Process
ENDIF

CHECK Ticket Status
IF Ticket.Status ≠ "In Progress" THEN
    DISPLAY "Ticket must be In Progress to mark as completed"
    EXIT Process
ENDIF

// Verify Evidence Was Submitted
CHECK if Evidence Exists for Ticket
IF Ticket.EvidenceID IS NULL OR Ticket.HasEvidence = FALSE THEN
    DISPLAY "Please upload completion evidence before marking as complete"
    REDIRECT to Evidence Upload (Process 2.4)
    EXIT Process
ENDIF

// Validate Completion Notes
IF Completion Notes is Empty OR Length < 20 characters THEN
    DISPLAY "Please provide detailed completion notes (minimum 20 characters)"
    EXIT Process
ENDIF

// Retrieve Evidence Record
RETRIEVE Evidence from Evidence Table WHERE EvidenceID = Ticket.EvidenceID

// Update Ticket Status
SET Ticket.Status = "Fixed"
SET Ticket.ResolvedAt = CURRENT_DATETIME

// Calculate Resolution Metrics
IF Ticket.ActualStartTime ≠ NULL THEN
    CALCULATE Actual Resolution Time = ResolvedAt - ActualStartTime
    SET Ticket.ActualResolutionTime = Actual Resolution Time
ELSE
    // If start time wasn't recorded, use assigned time
    CALCULATE Actual Resolution Time = ResolvedAt - AssignedAt
    SET Ticket.ActualResolutionTime = Actual Resolution Time
ENDIF

// Compare with Estimated Time
IF Ticket.EstimatedCompletionTime ≠ NULL THEN
```

Systems Analysis and Design Project

```
IF ResolvedAt > EstimatedCompletionTime THEN
    SET Ticket.CompletedLate = TRUE
    CALCULATE Delay = ResolvedAt - EstimatedCompletionTime
    SET Ticket.DelayDuration = Delay
ELSE
    SET Ticket.CompletedOnTime = TRUE
ENDIF
ENDIF

// Update Ticket Final Details
SET Ticket.CompletionNotes = Completion Notes
SET Ticket.ResolutionSummary = Work Summary
SET Ticket.PartsUsed = Parts Used
SET Ticket.LaborHours = Time Spent

UPDATE Ticket in Tickets Database (D2)

// Update Technician Performance Metrics
GET Technician Current Metrics
CALCULATE New Average Resolution Time
    = (Current Avg * Completed Count + Actual Resolution Time) / (Completed Count +
1)

INCREMENT Technician.CompletedTasksCount
UPDATE Technician.AverageResolutionTime = New Average Resolution Time
CALCULATE Technician.CompletionRate
    = (CompletedTasksCount / TotalAssignedTasks) * 100

UPDATE Technician Record in User Database (D1)

// Prepare Completion Record for History
CREATE Completion Data Object
    TicketID = TicketID
    TechnicianID = Ticket.TechicianID
    LocationID = Ticket.LocationID
    Category = Ticket.Category
    Priority = Ticket.Priority
    ResolutionSummary = Work Summary
    CompletionNotes = Completion Notes
    PartsUsed = Parts Used
    LaborHours = Time Spent
    CompletionDate = ResolvedAt
    ActualResolutionTime = Actual Resolution Time
    EvidenceID = Ticket.EvidenceID
    CompletedOnTime = Ticket.CompletedOnTime

// Send to Maintenance History Process
TRIGGER Process 4.2 (Update Maintenance History)
    Pass Completion Data Object

// Send Notifications
// 1. Notify Administrator for Review
SEND Notification to Administrators (Process 2.5)
    "Ticket " + TicketID + " marked as Fixed by " + Technician.Name + ". Awaiting
```

Systems Analysis and Design Project

review."

```
// 2. Notify Requester
SEND Notification to Ticket Requester (Process 2.5)
    "Your ticket " + TicketID + " has been completed. Please verify the work."

LOG "Ticket " + TicketID + " completed by Technician " + Technician.Name
DISPLAY "Maintenance work completed successfully"
DISPLAY "Ticket ID: " + TicketID
DISPLAY "Resolution Time: " + Actual Resolution Time

END Process
```

Refer to:

- FR-16.3 (Task Status Updates - Fixed)
- Table 81: Data Flow 30 Details
- Table 82: Data Flow 31 Details
- Table 83: Data Flow 32 Details

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Status = In Progress	Y	Y	Y	Y	N
Evidence Submitted	Y	Y	Y	N	-
Completion Notes Valid	Y	Y	N	-	-
Completed On Time	Y	N	-	-	-
Set Status = Fixed	X	X	-	-	-
Calculate Metrics	X	X	-	-	-
Update Technician Performance	X	X	-	-	-
Send to History	X	X	-	-	-
Send Notifications	X	X	-	-	-
Flag as Late	-	X	-	-	-
Display "Notes too short"	-	-	X	-	-
Display "Need evidence"	-	-	-	X	-
Display "Wrong status"	-	-	-	-	X

Unresolved Issues: None

Number: 4.2 **Name:** Update Maintenance History **Description:** Archives completed maintenance records for historical tracking, trend analysis, and performance reporting.

Input Data Flow:

- Maintenance Completion Record (from Process 4.1)

Systems Analysis and Design Project

Output Data Flow:

- Stored Maintenance History Entry (to Maintenance History D3)

Type of Process: Online Batch Manual

Subprogram/Function Name: updateMaintenanceHistory()

Process Logic (Structured English):

```
BEGIN Process 4.2: Update Maintenance History

READ Completion Data from Process 4.1
    TicketID, TechnicianID, LocationID, Category, Priority,
    ResolutionSummary, CompletionNotes, PartsUsed, LaborHours,
    CompletionDate, ActualResolutionTime, EvidenceID, CompletedOnTime

GENERATE History Record ID
CREATE Maintenance History Record
    HistoryID = Generated ID
    TicketID = Input TicketID
    TechnicianID = Input TechnicianID
    TechnicianName = Get from User Database
    LocationID = Input LocationID
    LocationDetails = Get from Location Database
    Category = Input Category
    Priority = Input Priority
    IssueDescription = Get from Original Ticket
    ResolutionSummary = Input ResolutionSummary
    CompletionNotes = Input CompletionNotes
    WorkPerformed = Extract from Notes
    PartsUsed = Input PartsUsed
    PartsOrderPrice = Calculate Total (if Parts Cost Available)
    LaborHours = Input LaborHours
    LaborCost = Calculate (LaborHours * Hourly Rate)
    TotalCost = PartsOrderPrice + LaborCost
    CompletionDate = Input CompletionDate
    ActualResolutionTime = Input ActualResolutionTime
    EvidenceID = Input EvidenceID
    CompletedOnTime = Input CompletedOnTime
    RecordedAt = CURRENT_DATETIME

// Link to Equipment (if equipment ID was specified)
IF Ticket Contains Equipment Information THEN
    SET EquipmentID = Ticket.EquipmentID
    SET ManufacturerDetails = Get from Equipment Database
    SET WarrantyInfo = Get from Equipment Database

    // Update Equipment Maintenance Log
    ADD History Record ID to Equipment.MaintenanceHistory
ENDIF

// Calculate MTBF (Mean Time Between Failures) for Equipment
```

Systems Analysis and Design Project

```
IF EquipmentID ≠ NULL THEN
    QUERY Previous Maintenance for Same Equipment
    IF Previous Records Exist THEN
        GET Last Maintenance Date
        CALCULATE Time Between Failures = CompletionDate - Last Maintenance Date
        UPDATE Equipment.MTBF Calculation
    ENDIF
ENDIF

WRITE Maintenance History Record to Maintenance History (D3)

// Update Location Statistics
UPDATE Location Statistics
    INCREMENT Location.TotalMaintenanceCount
    ADD Category to Location.MaintenanceCategories
    CALCULATE Location.AverageResolutionTime

// Update Category Statistics
UPDATE Category Statistics
    INCREMENT Global Category Count
    CALCULATE Average Resolution Time for Category
    UPDATE Most Common Issues List

LOG "Maintenance history archived for Ticket " + TicketID

END Process
```

Refer to:

- FR-19 (Maintenance History Access)
- FR-26 (Analytics and Reporting)
- Table 84: Data Flow 33 Details

Structured English (Simplified):

```
BEGIN

READ completion data
GENERATE history record ID

CREATE comprehensive history record with:
    - Ticket details
    - Technician details
    - Location details
    - Work performed
    - Parts and costs
    - Time metrics
    - Evidence reference

IF equipment involved THEN
    UPDATE equipment maintenance log
    CALCULATE MTBF for equipment
ENDIF
```

Systems Analysis and Design Project

WRITE to maintenance history database

UPDATE location statistics
UPDATE category statistics

LOG archive completion

END

Unresolved Issues: None

Number: 4.3 **Name:** Generate Reports & Metrics **Description:** Compiles maintenance data, calculates KPIs, generates analytics dashboards, and produces reports for administrators.

Input Data Flow:

- Report Request (from Administrator)
 - Date Range
 - Filters
 - Report Type

Output Data Flow:

- Reports & Metrics Store (to Reports & Analytics D4)
- Analytics Dashboard Data (from D2, D3 to Administrator)

Type of Process: Online Batch Manual

Subprogram/Function Name: generateReportsAndMetrics()

Process Logic (Structured English):

BEGIN Process 4.3: Generate Reports & Metrics

VERIFY User is Administrator
IF User Role ≠ "Administrator" THEN
 DISPLAY "Unauthorized access"
 EXIT Process
ENDIF

READ Report Parameters from Administrator
 Report Type, Date Range, Filters (Category, Location, Technician, Priority)

// Default to Current Month if no date range specified
IF Date Range Not Provided THEN
 SET Start Date = First Day of Current Month
 SET End Date = CURRENT_DATE
ENDIF

VALIDATE Date Range

Systems Analysis and Design Project

```
IF Start Date > End Date THEN
    DISPLAY "Invalid date range"
    EXIT Process
ENDIF

// =====
// COLLECT DATA BASED ON REPORT TYPE
// =====

IF Report Type = "Overall Dashboard" OR Report Type = "Summary" THEN
    // === KEY PERFORMANCE INDICATORS ===

    // 1. Total Tickets
    QUERY COUNT(*) FROM Tickets WHERE
        ReportedAt BETWEEN Start Date AND End Date
    SET KPI.TotalTickets = Query Result

    // 2. Open Issues
    QUERY COUNT(*) FROM Tickets WHERE
        Status IN ("Open", "Assigned", "In Progress") AND
        ReportedAt BETWEEN Start Date AND End Date
    SET KPI.OpenIssues = Query Result

    // 3. Resolved Issues
    QUERY COUNT(*) FROM Tickets WHERE
        Status IN ("Fixed", "Closed") AND
        ResolvedAt BETWEEN Start Date AND End Date
    SET KPI.ResolvedIssues = Query Result

    // 4. Average Resolution Time
    QUERY AVG(ActualResolutionTime) FROM Tickets WHERE
        Status = "Fixed" OR Status = "Closed" AND
        ResolvedAt BETWEEN Start Date AND End Date
    SET KPI.AvgResolutionTime = Query Result
    CONVERT to Hours or Days

    // 5. Completion Rate
    CALCULATE Completion Rate = (ResolvedIssues / TotalTickets) * 100
    SET KPI.CompletionRate = Completion Rate + "%"

    // 6. On-Time Completion Rate
    QUERY COUNT(*) FROM Tickets WHERE
        CompletedOnTime = TRUE AND
        ResolvedAt BETWEEN Start Date AND End Date
    SET On Time Count = Query Result
    CALCULATE On Time Rate = (On Time Count / ResolvedIssues) * 100
    SET KPI.OnTimeCompletionRate = On Time Rate + "%"
ENDIF

IF Report Type = "Maintenance Trends" OR Report Type = "Summary" THEN
    // === TREND ANALYSIS ===

    // Tickets by Category
    QUERY Category, COUNT(*) as Count FROM Tickets
```

Systems Analysis and Design Project

```
WHERE ReportedAt BETWEEN Start Date AND End Date
GROUP BY Category
ORDER BY Count DESC
SET Trends.ByCategory = Query Results

// Tickets by Priority
QUERY Priority, COUNT(*) as Count FROM Tickets
WHERE ReportedAt BETWEEN Start Date AND End Date
GROUP BY Priority
SET Trends.ByPriority = Query Results

// Tickets Over Time (Daily/Weekly/Monthly)
DETERMINE Time Granularity based on Date Range
IF Date Range <= 7 days THEN
    SET Granularity = "Daily"
ELSIF Date Range <= 90 days THEN
    SET Granularity = "Weekly"
ELSE
    SET Granularity = "Monthly"
ENDIF

QUERY Date, COUNT(*) as Count FROM Tickets
WHERE ReportedAt BETWEEN Start Date AND End Date
GROUP BY Date (Grouped by Granularity)
ORDER BY Date ASC
SET Trends.TicketsOverTime = Query Results

// Recurring Issue Patterns
QUERY Category, LocationID, COUNT(*) as Frequency FROM Tickets
WHERE ReportedAt BETWEEN Start Date AND End Date
GROUP BY Category, LocationID
HAVING Frequency >= 3
ORDER BY Frequency DESC
SET Trends.RecurringIssues = Query Results
ENDIF

IF Report Type = "Frequently Reported Areas" OR Report Type = "Summary" THEN
    // === LOCATION ANALYSIS ===

    QUERY
        L.LocationID,
        L.BuildingName,
        L.Zone,
        COUNT(T.TicketID) as IssueCount,
        AVG(T.ActualResolutionTime) as AvgResolutionTime
    FROM Tickets T
    JOIN Locations L ON T.LocationID = L.LocationID
    WHERE T.ReportedAt BETWEEN Start Date AND End Date
    GROUP BY L.LocationID
    ORDER BY IssueCount DESC
    LIMIT 10

    SET LocationData.TopLocations = Query Results
```

Systems Analysis and Design Project

```
// Generate Heat Map Data
QUERY LocationID, GeoCoordinates, COUNT(*) as Density
FROM Tickets T
JOIN Locations L ON T.LocationID = L.LocationID
WHERE ReportedAt BETWEEN Start Date AND End Date
GROUP BY LocationID

SET LocationData.HeatMapData = Query Results (for FR-21.5)
ENDIF

IF Report Type = "Equipment Performance" OR Report Type = "Summary" THEN
// === EQUIPMENT ANALYSIS ===

    QUERY
        EquipmentID,
        EquipmentType,
        COUNT(*) as MaintenanceFrequency,
        AVG(ActualResolutionTime) as AvgRepairTime,
        SUM(TotalCost) as TotalMaintenanceCost
    FROM Maintenance_History
    WHERE CompletionDate BETWEEN Start Date AND End Date
    AND EquipmentID IS NOT NULL
    GROUP BY EquipmentID
    ORDER BY MaintenanceFrequency DESC

    SET EquipmentData = Query Results

    // Calculate MTBF for Equipment
    FOR EACH Equipment DO
        QUERY Previous Maintenance Dates for Equipment
        CALCULATE Average Time Between Failures
        SET Equipment.MTBF = Calculated MTBF
    ENDFOR
ENDIF

IF Report Type = "Technician Performance" OR Report Type = "Summary" THEN
// === TECHNICIAN ANALYSIS ===

    QUERY
        T.TechicianID,
        U.FullName,
        COUNT(CASE WHEN T.Status IN ('Fixed','Closed') THEN 1 END) as CompletedTasks,
        COUNT(CASE WHEN T.Status IN ('Assigned','In Progress') THEN 1 END) as ActiveTasks,
        AVG(T.ActualResolutionTime) as AvgResolutionTime,
        AVG(CASE WHEN T.CompletedOnTime = TRUE THEN 1.0 ELSE 0.0 END) * 100 as OnTimeRate,
        Technician.CustomerSatisfactionRating
    FROM Tickets T
    JOIN Users U ON T.TechicianID = U.UserID
    JOIN Technician ON U.UserID = Technician.UserID
    WHERE T.AssignedAt BETWEEN Start Date AND End Date
    GROUP BY T.TechicianID
```

Systems Analysis and Design Project

```
ORDER BY CompletedTasks DESC

SET TechnicianData.Performance = Query Results

// Workload Distribution
QUERY TechnicianID, COUNT(*) as CurrentWorkload
FROM Tickets
WHERE Status IN ('Assigned', 'In Progress')
GROUP BY TechnicianID

SET TechnicianData.WorkloadDistribution = Query Results
ENDIF

IF Report Type = "Cost Analysis" OR Report Type = "Summary" THEN
    // === COST ANALYSIS ===

    QUERY
        SUM(PartsOrderPrice) as TotalPartsCost,
        SUM(LaborCost) as TotalLaborCost,
        SUM(TotalCost) as TotalMaintenanceCost
    FROM Maintenance_History
    WHERE CompletionDate BETWEEN Start Date AND End Date

    SET CostData.Summary = Query Results

    // Cost by Category
    QUERY Category, SUM(TotalCost) as TotalCost
    FROM Maintenance_History
    WHERE CompletionDate BETWEEN Start Date AND End Date
    GROUP BY Category
    ORDER BY TotalCost DESC

    SET CostData.ByCategory = Query Results

    // Cost Trend Over Time
    QUERY
        DATE(CompletionDate) as Date,
        SUM(TotalCost) as DailyCost
    FROM Maintenance_History
    WHERE CompletionDate BETWEEN Start Date AND End Date
    GROUP BY DATE(CompletionDate)
    ORDER BY Date ASC

    SET CostData.Trend = Query Results
ENDIF

// =====
// GENERATE VISUALIZATIONS
// =====

CREATE Chart Data Objects

// 1. Tickets by Status Pie Chart
PREPARE Pie Chart Data
```

Systems Analysis and Design Project

```
Labels: ["Open", "In Progress", "Fixed", "Closed"]
Values: [Count for Each Status]
Colors: [Status-specific colors]

// 2. Tickets Over Time Line Chart
PREPARE Line Chart Data
  X-Axis: Dates
  Y-Axis: Ticket Count
  Data: Trends.TicketsOverTime

// 3. Category Distribution Bar Chart
PREPARE Bar Chart Data
  X-Axis: Categories
  Y-Axis: Count
  Data: Trends.ByCategory

// 4. Technician Performance Comparison
PREPARE Multi-Bar Chart Data
  X-Axis: Technician Names
  Y-Axis: Metrics
  Series 1: Completed Tasks
  Series 2: Active Tasks
  Data: TechnicianData.Performance

// 5. Location Heat Map
PREPARE Heat Map Data
  Coordinates: Location.GeoCoordinates
  Intensity: Issue Density
  Data: LocationData.HeatMapData

// =====
// STORE REPORT
// =====

GENERATE Report ID
CREATE Report Record
  ReportID = Generated ID
  ReportType = Report Type
  GeneratedBy = Administrator.UserID
  GeneratedAt = CURRENT_DATETIME
  DateRange = {Start Date, End Date}
  Filters = Applied Filters
  KPIs = KPI Data Object
  Trends = Trends Data Object
  LocationData = Location Data Object
  EquipmentData = Equipment Data Object
  TechnicianData = Technician Data Object
  CostData = Cost Data Object
  Charts = Chart Data Objects

WRITE Report to Reports & Analytics (D4)

LOG "Report " + Report ID + " generated by " + Administrator.Name
```

Systems Analysis and Design Project

```
// Return Report for Display
```

```
RETURN Report Data Object
```

```
END Process
```

Refer to:

- FR-26 (Analytics and Reporting)
- Table 85-87: Data Flow 35-37 Details

Systems Analysis and Design Project

Decision Tree:

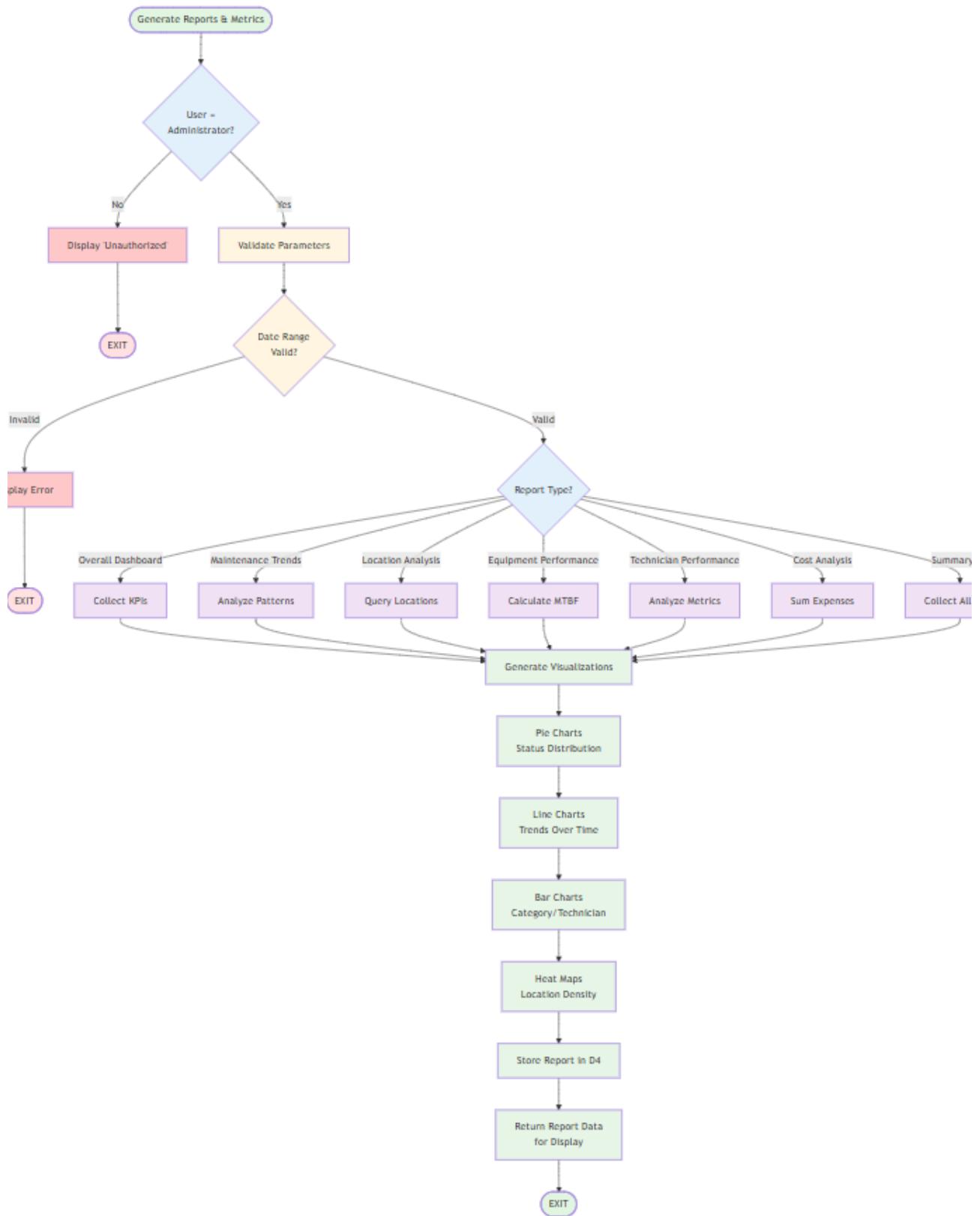


Figure 53: Decision Tree 4.3

Unresolved Issues: None

Systems Analysis and Design Project

Number: 4.4 **Name:** Provide Analytics Dashboard / Export **Description:** Displays analytics dashboards to administrators and allows exporting reports in PDF, Excel, or CSV formats.

Input Data Flow:

- Analytics View/Export Request (from Administrator)
 - Report ID (optional, for existing reports)
 - Export Format (PDF/Excel/CSV)

Output Data Flow:

- Analytics Dashboard / Exported Report (to Administrator)

Type of Process: Online Batch Manual

Subprogram/Function Name: provideAnalyticsDashboard()

Process Logic (Structured English):

BEGIN Process 4.4: Provide Analytics Dashboard / Export

```
VERIFY User is Administrator
IF User Role ≠ "Administrator" THEN
    DISPLAY "Unauthorized access"
    EXIT Process
ENDIF

READ Request Type (View Dashboard OR Export Report)

// =====
// VIEW ANALYTICS DASHBOARD
// =====

IF Request Type = "View Dashboard" THEN
    // Check for existing recent report
    QUERY Most Recent Report from Reports & Analytics (D4)
    WHERE GeneratedBy = Administrator.UserID
    AND GeneratedAt >= (CURRENT_DATE - 1 day)
```

```
    IF Recent Report Exists THEN
        RETRIEVE Report Data from D4
    ELSE
        // Generate new report
        TRIGGER Process 4.3 (Generate Reports & Metrics)
        WITH Report Type = "Summary"
        WAIT for Report Generation
        RETRIEVE Generated Report Data
    ENDIF
```

```
// Render Dashboard
DISPLAY Dashboard Header
    Title: "Maintenance Analytics Dashboard"
    Date Range: Report.DateRange
```

Systems Analysis and Design Project

Last Updated: Report.GeneratedAt

```
// Display KPIs in Cards
DISPLAY KPI Cards Grid
    Card 1: Total Tickets (KPI.TotalTickets)
    Card 2: Open Issues (KPI.OpenIssues) - Red Badge
    Card 3: Resolved Issues (KPI.ResolvedIssues) - Green Badge
    Card 4: Avg Resolution Time (KPI.AvgResolutionTime)
    Card 5: Completion Rate (KPI.CompletionRate)
    Card 6: On-Time Rate (KPI.OnTimeCompletionRate)

// Display Charts
RENDER Chart Section "Tickets Overview"
    Pie Chart: Tickets by Status
    Line Chart: Tickets Over Time
    Bar Chart: Tickets by Category

RENDER Chart Section "Location Analysis"
    Heat Map: Issue Density by Location
    Top 10 Locations Table: Most Reported Areas

RENDER Chart Section "Technician Performance"
    Multi-Bar Chart: Completed vs Active Tasks
    Table: Detailed Technician Metrics
    Workload Distribution Pie Chart

RENDER Chart Section "Cost Analysis"
    Line Chart: Cost Trend Over Time
    Pie Chart: Cost by Category
    Summary Cards: Total Costs

RENDER Chart Section "Equipment Performance"
    Table: Equipment Maintenance Frequency
    Bar Chart: MTBF by Equipment Type

// Provide Dashboard Actions
DISPLAY Action Buttons
    "Refresh Data"
    "Export Report"
    "Customize Filters"
    "Schedule Automated Report"

// Enable Drill-Down
FOR EACH Interactive Chart Element DO
    ON CLICK Event DO
        SHOW Detailed View
        ALLOW Filtering by Clicked Element
    ENDDO
ENDFOR
ENDIF

// =====
// EXPORT REPORT
// =====
```

Systems Analysis and Design Project

```
IF Request Type = "Export Report" THEN
    READ Export Format and Report ID from Administrator

    RETRIEVE Report Data from Reports & Analytics (D4)
    IF Report Not Found THEN
        DISPLAY "Report not found. Please generate a report first."
        EXIT Process
    ENDIF

    VALIDATE Export Format
    IF Export Format NOT IN ("PDF", "Excel", "CSV") THEN
        DISPLAY "Invalid export format. Supported: PDF, Excel, CSV"
        EXIT Process
    ENDIF

// === EXPORT TO PDF ===
IF Export Format = "PDF" THEN
    CREATE PDF Document using iText Library

    // Add Header
    ADD Company Logo (if available)
    ADD Title "SALLEHA Maintenance Report"
    ADD Date Range
    ADD Generated By: Administrator.Name
    ADD Generated At: Report.GeneratedAt
    ADD Page Numbers

    // Add Executive Summary Section
    ADD Section "Executive Summary"
    ADD KPIs Table
        Row: Total Tickets | KPI.TotalTickets
        Row: Open Issues | KPI.OpenIssues
        Row: Resolved Issues | KPI.ResolvedIssues
        Row: Avg Resolution Time | KPI.AvgResolutionTime
        Row: Completion Rate | KPI.CompletionRate

    // Add Charts as Images
    ADD Section "Maintenance Trends"
    CONVERT Chart to Image (PNG)
    EMBED Chart Images in PDF

    ADD Section "Location Analysis"
    ADD Top Locations Table
    EMBED Heat Map Image

    ADD Section "Technician Performance"
    ADD Technician Performance Table
    EMBED Performance Charts

    ADD Section "Equipment Analysis"
    ADD Equipment Table with MTBF

    ADD Section "Cost Analysis"
    ADD Cost Summary Table
```

Systems Analysis and Design Project

```
ADD Cost Breakdown by Category

// Add Footer
ADD Disclaimer: "Generated by SALLEHA Maintenance System"
ADD Contact Information

SET Filename = "SALLEHA_Report_" + Report.DateRange + "_" + TIMESTAMP +
".pdf"
SAVE PDF File
ENDIF

// === EXPORT TO EXCEL ===
IF Export Format = "Excel" THEN
    CREATE Excel Workbook using Apache POI

    // Sheet 1: Summary
    CREATE Sheet "Summary"
    ADD Title Row
    ADD KPIs Table with Formatting
        - Bold Headers
        - Color-coded Cells based on Values
        - Conditional Formatting
    ADD Charts (Excel Native Charts)

    // Sheet 2: Tickets Data
    CREATE Sheet "Tickets Detail"
    ADD Headers: TicketID, Title, Category, Location, Status, Priority,
ReportedAt, ResolvedAt, ResolutionTime
    QUERY All Tickets in Date Range
    FOR EACH Ticket DO
        ADD Row with Ticket Data
    ENDFOR
    APPLY AutoFilter
    FREEZE Top Row

    // Sheet 3: Location Analysis
    CREATE Sheet "Locations"
    ADD Location Statistics Table
    ADD Formulas for Calculations

    // Sheet 4: Technician Performance
    CREATE Sheet "Technicians"
    ADD Technician Performance Table
    ADD Formulas for Metrics
    ADD Charts

    // Sheet 5: Cost Analysis
    CREATE Sheet "Costs"
    ADD Cost Breakdown Table
    ADD SUM Formulas
    ADD Cost Charts

    // Sheet 6: Raw Data (for pivot tables)
    CREATE Sheet "Raw Data"
```

Systems Analysis and Design Project

```
ADD Complete Dataset
ENABLE Pivot Table Creation

SET Filename = "SALLEHA_Report_" + Report.DateRange + "_" + TIMESTAMP +
".xlsx"
    SAVE Excel File
ENDIF

// === EXPORT TO CSV ===
IF Export Format = "CSV" THEN
    CREATE CSV File

    // Determine CSV Content based on Report Type
    IF Administrator Selects "All Data" THEN
        // Export comprehensive dataset
        ADD CSV Headers
            TicketID, Title, Description, Category, Location, Priority, Status,
            RequesterID, TechnicianID, TechnicianName, ReportedAt, AssignedAt,
            ResolvedAt, ActualResolutionTime, PartsUsed, LaborHours, TotalCost,
            CompletedOnTime, EvidenceID

        QUERY All Tickets in Date Range
        JOIN with Related Tables
        FOR EACH Ticket DO
            ADD Row with All Fields
            ESCAPE Commas and Quotes in Text Fields
        ENDFOR
    ELSIF Administrator Selects "Summary Only" THEN
        // Export aggregated data
        ADD CSV Headers
            Metric, Value

        ADD Rows
            "Total Tickets", KPI.TotalTickets
            "Open Issues", KPI.OpenIssues
            "Resolved Issues", KPI.ResolvedIssues
            ...
    ENDIF

    SET Filename = "SALLEHA_Report_" + Report.DateRange + "_" + TIMESTAMP +
".csv"
    SAVE CSV File
ENDIF

// Store Export History
CREATE Export Record
ReportID = Report.ReportID
ExportedBy = Administrator.UserID
ExportedAt = CURRENT_DATETIME
ExportFormat = Export Format
Filename = Filename

WRITE to Export History Table
```

Systems Analysis and Design Project

```
// Provide Download
GENERATE Download Link
SET Download Expiration = CURRENT_DATETIME + 24 hours

DISPLAY "Report exported successfully"
DISPLAY "Format: " + Export Format
DISPLAY "Filename: " + Filename
PROVIDE Download Button

LOG "Report " + Report.ReportID + " exported by " + Administrator.Name + " as "
+ Export Format
ENDIF

END Process
```

Refer to:

- FR-27 (Report Export Functionality)
- Table 88-90: Data Flow 38-40 Details

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Request = View Dashboard	Y	N	N	N	N
Request = Export	N	Y	Y	Y	N
Recent Report Exists	Y	-	-	-	-
Export Format = PDF	-	Y	N	N	-
Export Format = Excel	-	N	Y	N	-
Export Format = CSV	-	N	N	Y	-
Retrieve Existing Report	X	-	-	-	-
Generate New Report	-	X	X	X	-
Render Dashboard	X	-	-	-	-
Export as PDF	-	X	-	-	-
Export as Excel	-	-	X	-	-
Export as CSV	-	-	-	X	-
Provide Download Link	-	X	X	X	-
Display Error	-	-	-	-	X

Unresolved Issues: None

Systems Analysis and Design Project

4.3.5. Fragment 5: Notification Management Processes

Number: 5.1

Name: Receive Trigger

Description: Receives and validates notification triggers from various system processes (ticket creation, status changes, assignments, etc.) and prepares them for notification generation.

Input Data Flow:

- Notification triggers from various processes (2.1, 2.3, 2.4, 3.2, 4.1)
 - Event Type
 - TicketID
 - Affected UserIDs
 - Priority Level
 - Timestamp

Output Data Flow:

- Notification Trigger Details (to Process 5.2)

Type of Process:

Online Batch Manual

Subprogram/Function Name: receiveTrigger()

Process Logic (Structured English):

```
BEGIN Process 5.1: Receive Trigger

READ Trigger Event from Source Process
    Event Type, TicketID, Affected UserIDs, Additional Data

VALIDATE Trigger Data
IF Event Type is NULL OR TicketID is NULL THEN
    LOG "Invalid trigger received"
    EXIT Process
ENDIF

// Determine affected users based on event type
INITIALIZE Recipients List

IF Event Type = "Ticket Created" THEN
    ADD All Active Administrators to Recipients
ELSIF Event Type = "Status Changed" THEN
    RETRIEVE Ticket from Tickets Database (D2)
    ADD Ticket.RequesterID to Recipients
    IF Ticket.TechicianID ≠ NULL THEN
        ADD Ticket.TechicianID to Recipients
    ENDIF
ELSIF Event Type = "Ticket Assigned" THEN
    RETRIEVE Ticket from Tickets Database (D2)
    ADD Ticket.TechicianID to Recipients
```

Systems Analysis and Design Project

```
ADD Ticket.RequesterID to Recipients
ELSIF Event Type = "Priority Changed" THEN
    RETRIEVE Ticket from Tickets Database (D2)
    IF Ticket.TechnicianID ≠ NULL THEN
        ADD Ticket.TechnicianID to Recipients
    ENDIF
    ADD All Active Administrators to Recipients
ELSIF Event Type = "Comment Added" THEN
    RETRIEVE Ticket from Tickets Database (D2)
    ADD Ticket.RequesterID to Recipients
    IF Ticket.TechnicianID ≠ NULL THEN
        ADD Ticket.TechnicianID to Recipients
    ENDIF
ELSIF Event Type = "Evidence Uploaded" THEN
    RETRIEVE Ticket from Tickets Database (D2)
    ADD Ticket.RequesterID to Recipients
    ADD All Active Administrators to Recipients
ELSIF Event Type = "Ticket Overdue" THEN
    ADD All Active Administrators to Recipients
    RETRIEVE Ticket from Tickets Database (D2)
    IF Ticket.TechnicianID ≠ NULL THEN
        ADD Ticket.TechnicianID to Recipients
    ENDIF
ENDIF

// Remove duplicate user IDs
REMOVE Duplicates from Recipients List

// Determine notification priority
IF Event Type IN ("Ticket Overdue", "Emergency Ticket", "Account Locked") THEN
    SET Priority Level = "High"
ELSIF Event Type IN ("Status Changed", "Priority Changed", "Ticket Assigned") THEN
    SET Priority Level = "Medium"
ELSE
    SET Priority Level = "Low"
ENDIF

// Create trigger details object
CREATE Trigger Details Object
TriggerID = GENERATE UUID
EventType = Event Type
TicketID = TicketID
Recipients = Recipients List
PriorityLevel = Priority Level
Timestamp = CURRENT_DATETIME
AdditionalData = Additional Data

LOG "Notification trigger received: " + Event Type + " for Ticket " + TicketID

// Pass to notification preparation
TRIGGER Process 5.2 (Prepare Notification)
    Pass Trigger Details Object

END Process
```

Systems Analysis and Design Project

Refer to:

- Table 91: Data Flow 41 Details
- Figure 44: DFD Level 1-Fragment 5

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Event Type Valid	Y	Y	Y	Y	N
TicketID Provided	Y	Y	Y	N	-
Event = Ticket Created	Y	N	N	-	-
Event = Status Changed	N	Y	N	-	-
Event = Ticket Assigned	N	N	Y	-	-
Identify Recipients	X	X	X	-	-
Set Priority Level	X	X	X	-	-
Create Trigger Object	X	X	X	-	-
Pass to Process 5.2	X	X	X	-	-
Display "Invalid TicketID"	-	-	-	X	-
Display "Invalid Event Type"	-	-	-	-	X

Unresolved Issues: None

Number: 5.2

Name: Prepare Notification

Description: Retrieves ticket details and user contact information, constructs personalized notification messages, and determines delivery methods based on user preferences.

Input Data Flow:

- Notification Trigger Details (from Process 5.1)
- Ticket Details (from Tickets Database D2)
- User Contact Information (from User Database D1)

Output Data Flow:

- Prepared Notification Message (to Process 5.3)

Type of Process:

Online Batch Manual

Subprogram/Function Name: prepareNotification()

Process Logic (Structured English):

BEGIN Process 5.2: Prepare Notification

Systems Analysis and Design Project

```
READ Trigger Details from Process 5.1
    TriggerID, EventType, TicketID, Recipients, PriorityLevel, Timestamp

// Retrieve ticket information
QUERY Tickets Database (D2) WHERE TicketID = Input TicketID
IF Ticket Not Found THEN
    LOG "Ticket not found for notification: " + TicketID
    EXIT Process
ENDIF

RETRIEVE Ticket Details
    Title, Category, Status, Priority, Location, RequesterID, TechnicianID

// Retrieve location details for context
IF Ticket.LocationID ≠ NULL THEN
    QUERY Locations Database WHERE LocationID = Ticket.LocationID
    SET Location Details = BuildingName + " - " + RoomNumber
ELSE
    SET Location Details = "Unspecified"
ENDIF

// Process notifications for each recipient
FOR EACH RecipientID in Recipients List DO

    // Retrieve recipient information
    QUERY User Database (D1) WHERE UserID = RecipientID
    IF User Not Found THEN
        LOG "User not found: " + RecipientID
        CONTINUE to Next Recipient
    ENDIF

    RETRIEVE User Details
        FullName, Email, PhoneNumber, Role

    // Retrieve notification preferences
    QUERY Notification Preferences WHERE UserID = RecipientID
    IF Preferences Not Found THEN
        // Use default preferences
        SET InAppEnabled = TRUE
        SET EmailEnabled = TRUE
        SET SMSEnabled = FALSE
        SET PushEnabled = TRUE
    ELSE
        RETRIEVE Preferences
            InAppEnabled, EmailEnabled, SMSEnabled, PushEnabled,
            DoNotDisturbStart, DoNotDisturbEnd
    ENDIF

    // Check Do Not Disturb settings
    SET Current Time = CURRENT_TIME
    IF DoNotDisturbStart ≠ NULL AND DoNotDisturbEnd ≠ NULL THEN
        IF Current Time BETWEEN DoNotDisturbStart AND DoNotDisturbEnd THEN
            IF PriorityLevel ≠ "High" THEN
                LOG "User " + RecipientID + " in Do Not Disturb mode"
```

Systems Analysis and Design Project

```
QUEUE Notification for Later Delivery
CONTINUE to Next Recipient
ENDIF
ENDIF
ENDIF

// Build notification message based on event type and user role
INITIALIZE Message = ""
INITIALIZE Subject = ""

IF EventType = "Ticket Created" THEN
    SET Subject = "New Maintenance Ticket Created"
    SET Message = "A new maintenance ticket has been submitted."
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "Title: " + Ticket.Title + "\n" +
                  "Category: " + Ticket.Category + "\n" +
                  "Location: " + Location Details + "\n" +
                  "Priority: " + Ticket.Priority

ELSIF EventType = "Status Changed" THEN
    SET Subject = "Ticket Status Updated"
    IF User.Role = "Resident" THEN
        SET Message = "Your maintenance ticket status has been updated to: " +
                      Ticket.Status
    ELSE
        SET Message = "Ticket " + TicketID + " status changed to: " +
                      Ticket.Status
    ENDIF
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "New Status: " + Ticket.Status + "\n" +
                  "Location: " + Location Details

ELSIF EventType = "Ticket Assigned" THEN
    SET Subject = "New Task Assigned"
    IF User.Role = "Technician" THEN
        SET Message = "A new maintenance task has been assigned to you."
        SET Details = "Ticket ID: " + TicketID + "\n" +
                      "Title: " + Ticket.Title + "\n" +
                      "Category: " + Ticket.Category + "\n" +
                      "Priority: " + Ticket.Priority + "\n" +
                      "Location: " + Location Details
    ELSE
        SET Message = "Your ticket has been assigned to a technician."
        SET Details = "Ticket ID: " + TicketID
    ENDIF

ELSIF EventType = "Priority Changed" THEN
    SET Subject = "Ticket Priority Updated"
    SET Message = "The priority of ticket " + TicketID +
                  " has been changed to: " + Ticket.Priority
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "New Priority: " + Ticket.Priority + "\n" +
                  "Location: " + Location Details
```

Systems Analysis and Design Project

```
ELSIF EventType = "Comment Added" THEN
    SET Subject = "New Comment on Your Ticket"
    SET Message = "A new comment has been added to your maintenance ticket."
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "Status: " + Ticket.Status

ELSIF EventType = "Evidence Uploaded" THEN
    SET Subject = "Maintenance Evidence Submitted"
    IF User.Role = "Resident" THEN
        SET Message = "The technician has uploaded completion evidence for your
ticket."
    ELSE
        SET Message = "Completion evidence has been uploaded for ticket " +
TicketID
    ENDIF
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "Please review the evidence."

ELSIF EventType = "Ticket Overdue" THEN
    SET Subject = "URGENT: Ticket Overdue"
    SET Message = "Ticket " + TicketID + " is overdue and requires immediate
attention."
    SET Details = "Ticket ID: " + TicketID + "\n" +
                  "Category: " + Ticket.Category + "\n" +
                  "Priority: " + Ticket.Priority + "\n" +
                  "Location: " + Location Details
ENDIF

// Determine notification type for database
IF EventType = "Ticket Created" OR EventType = "Ticket Overdue" THEN
    SET NotificationType = "Alert"
ELSIF EventType = "Ticket Assigned" THEN
    SET NotificationType = "Assignment"
ELSIF EventType = "Priority Changed" THEN
    SET NotificationType = "PriorityChange"
ELSE
    SET NotificationType = "StatusChange"
ENDIF

// Check for notification grouping (last 10 minutes)
QUERY Notifications Database WHERE
    UserID = RecipientID AND
    RelatedTicketID = TicketID AND
    NotificationType = NotificationType AND
    CreatedAt >= (CURRENT_DATETIME - 10 minutes) AND
    IsRead = FALSE

IF Similar Unread Notification Exists THEN
    // Update existing notification instead of creating new
    SET GroupedNotification = TRUE
    UPDATE Existing Notification
        Message = "Multiple updates for ticket " + TicketID
        UpdatedAt = CURRENT_DATETIME
ELSE
```

Systems Analysis and Design Project

```
SET GroupedNotification = FALSE
ENDIF

// Generate notification ID
GENERATE NotificationID

// Create prepared notification object
CREATE Prepared Notification
    NotificationID = NotificationID
    UserID = RecipientID
    UserName = User.FullName
    UserEmail = User.Email
    UserPhone = User.PhoneNumber
    UserRole = User.Role
    RelatedTicketID = TicketID
    NotificationType = NotificationType
    Subject = Subject
    Message = Message
    Details = Details
    PriorityLevel = PriorityLevel
    DeliveryMethods = {
        InApp: InAppEnabled,
        Email: EmailEnabled,
        SMS: SMSEnabled,
        Push: PushEnabled
    }
    IsGrouped = GroupedNotification
    CreatedAt = CURRENT_DATETIME
    ExpirationDate = CURRENT_DATETIME + 90 days

LOG "Notification prepared for User " + RecipientID + " - " + EventType

// Send to delivery process
TRIGGER Process 5.3 (Send Notification)
    Pass Prepared Notification Object

ENDFOR

END Process
```

Refer to:

- Table 92-96: Data Flow 42-46 Details
- FR-10 (Status Change Notifications)
- FR-18 (Technician Notifications)
- FR-25 (Administrator Notifications)

Decision Table:

Systems Analysis and Design Project

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Ticket Found	Y	Y	Y	Y	Y	N
User Found	Y	Y	Y	Y	N	-
In DND Mode	Y	Y	N	N	-	-
Priority = High	Y	N	-	-	-	-
Recent Similar Notification	-	-	Y	N	-	-
Retrieve Preferences	X	X	X	X	-	-
Build Message	X	X	X	X	-	-
Override DND	X	-	-	-	-	-
Queue for Later	-	X	-	-	-	-
Group Notification	-	-	X	-	-	-
Create New Notification	X	-	-	X	-	-
Send to Process 5.3	X	-	X	X	-	-
Log “User Not Found”	-	-	-	-	X	-
Log “Ticket Not Found”	-	-	-	-	-	X

Unresolved Issues: None

Number: 5.3

Name: Send Notification

Description: Delivers notifications to users through multiple channels (in-app, email, SMS, push notifications) based on user preferences and priority levels.

Input Data Flow:

- Prepared Notification Message (from Process 5.2)

Output Data Flow:

- Notification (to Resident/Technician/Administrator)
- Stored Notification Record (to Notifications Database)

Type of Process:

Online Batch Manual

Subprogram/Function Name: sendNotification()

Process Logic (Structured English):

BEGIN Process 5.3: Send Notification

```
READ Prepared Notification from Process 5.2
    NotificationID, UserID, UserName, UserEmail, UserPhone, UserRole,
    RelatedTicketID, NotificationType, Subject, Message, Details,
    PriorityLevel, DeliveryMethods, IsGrouped, CreatedAt, ExpirationDate
```

Systems Analysis and Design Project

```
// Store notification in database first
CREATE Notification Record
    NotificationID = NotificationID
    UserID = UserID
    RelatedTicketID = RelatedTicketID
    NotificationType = NotificationType
    Message = Message
    IsRead = FALSE
    DeliveryMethod = "multi-channel"
    PriorityLevel = PriorityLevel
    ExpirationDate = ExpirationDate
    CreatedAt = CreatedAt

WRITE Notification Record to Notifications Database

LOG "Notification " + NotificationID + " stored for User " + UserID

// Track delivery status for each channel
INITIALIZE Delivery Status = {
    InApp: FALSE,
    Email: FALSE,
    SMS: FALSE,
    Push: FALSE
}

// =====
// CHANNEL 1: IN-APP NOTIFICATION
// =====

IF DeliveryMethods.InApp = TRUE THEN
    TRY
        // Check if user has active session
        QUERY Active Sessions WHERE UserID = UserID

        IF Active Session Exists THEN
            // Send via WebSocket
            CREATE WebSocket Payload
                type: "notification"
                notificationID: NotificationID
                ticketID: RelatedTicketID
                subject: Subject
                message: Message
                priority: PriorityLevel
                timestamp: CreatedAt

            SEND WebSocket Message to User Session

            // Update notification badge count
            INCREMENT User.NotificationBadgeCount
            SEND Badge Update via WebSocket

            // Play sound for high priority
            IF PriorityLevel = "High" THEN
```

Systems Analysis and Design Project

```
    SEND Sound Trigger "urgent.mp3"
ELSE
    SEND Sound Trigger "notification.mp3"
ENDIF

SET Delivery Status.InApp = TRUE
LOG "In-app notification sent to User " + UserID
ELSE
    LOG "No active session for User " + UserID + " - notification queued"
ENDIF

CATCH Error
    LOG "In-app notification failed: " + Error.Message
    SET Delivery Status.InApp = FALSE
ENDTRY
ENDIF

// =====
// CHANNEL 2: EMAIL NOTIFICATION
// =====

IF DeliveryMethods.Email = TRUE THEN
    IF UserEmail ≠ NULL AND UserEmail ≠ "" THEN
        TRY
            // Build HTML email template
            CREATE Email Content
                To: UserEmail
                Subject: "[SALLEHA] " + Subject

                Body (HTML):
                <html>
                    <body style="font-family: Arial, sans-serif;">
                        <div style="background: #f5f5f5; padding: 20px;">
                            <div style="background: white; padding: 20px; border-
radius: 8px;">
                                <h2 style="color: #333;">Hello, {UserName}</h2>
                                <p>{Message}</p>
                                <div style="background: #f9f9f9; padding: 15px;
margin: 15px 0;">
                                    <pre style="margin: 0;">{Details}</pre>
                                </div>
                                <a href="https://salleha.system.com/tickets/
{RelatedTicketID}" style="display: inline-block; background:
#007bff;
color: white; padding: 10px 20px;
text-decoration: none; border-radius:
5px;">
                                    View Ticket
                                </a>
                            <hr style="margin: 20px 0;">
                            <p style="font-size: 12px; color: #666;">
                                You are receiving this notification because you
have
                            </p>
                        </div>
                    </body>
                </html>
            ENDTRY
        ENDIF
    ENDIF
ENDIF
```

Systems Analysis and Design Project

```
email notifications enabled.  
<a href="https://salleha.system.com/  
preferences">  
    Manage your notification preferences  
</a>  
</p>  
</div>  
</div>  
</body>  
</html>  
  
// Send email via Nodemailer  
SEND Email via SMTP  
    Service: Gmail SMTP / SendGrid / AWS SES  
    From: "SALLEHA System <no-reply@salleha.system.com>"  
  
    SET Delivery Status.Email = TRUE  
    LOG "Email sent to " + UserEmail  
  
    CATCH Error  
        LOG "Email delivery failed: " + Error.Message  
        SET Delivery Status.Email = FALSE  
    ENDTRY  
ELSE  
    LOG "No email address for User " + UserID  
ENDIF  
ENDIF  
  
// =====  
// CHANNEL 3: SMS NOTIFICATION  
// =====  
  
IF DeliveryMethods.SMS = TRUE THEN  
    IF PriorityLevel = "High" OR PriorityLevel = "Immediate" THEN  
        IF UserPhone ≠ NULL AND UserPhone ≠ "" THEN  
            TRY  
                // Build SMS message (max 160 characters)  
                SET SMS.Text = "SALLEHA ALERT: " + Message  
                IF LENGTH(SMS.Text) > 140 THEN  
                    SET SMS.Text = SUBSTRING(SMS.Text, 0, 137) + "..."  
                ENDIF  
  
                // Add short URL  
                GENERATE Short URL for Ticket  
                FullURL: "https://salleha.system.com/tickets/" +  
RelatedTicketID  
                ShortURL: "https://slh.tk/t/" + RelatedTicketID  
  
                SET SMS.Text = SMS.Text + " View: " + ShortURL  
  
                // Send via Twilio API  
                SEND SMS via Twilio  
                To: UserPhone  
                From: Configured Twilio Number
```

Systems Analysis and Design Project

Body: SMS Text

```
SET Delivery Status.SMS = TRUE
LOG "SMS sent to " + UserPhone

CATCH Error
    LOG "SMS delivery failed: " + Error.Message
    SET Delivery Status.SMS = FALSE
ENDTRY

ELSE
    LOG "No phone number for User " + UserID
ENDIF

ELSE
    LOG "SMS only sent for high priority notifications"
ENDIF

ENDIF

// =====
// CHANNEL 4: PUSH NOTIFICATION (Mobile App)
// =====

IF DeliveryMethods.Push = TRUE THEN
    TRY
        // Check if user has FCM token (mobile app installed)
        QUERY Firebase Tokens WHERE UserID = UserID

        IF FCM Token Exists THEN
            // Build push notification payload
            CREATE Push Notification
                to: User.FCMTOKEN
                notification:
                    title: Subject
                    body: Message
                    sound: (PriorityLevel = "High" ? "urgent.mp3" : "default.mp3")
                    badge: User.NotificationBadgeCount + 1
                    icon: "ic_notification"
                    color: "#007bff"
                data:
                    type: NotificationType
                    ticketID: RelatedTicketID
                    notificationID: NotificationID
                    priority: PriorityLevel
                    deepLink: "salleha://tickets/" + RelatedTicketID

            // Send via Firebase Cloud Messaging
            SEND Push Notification via FCM API

            SET Delivery Status.Push = TRUE
            LOG "Push notification sent to User " + UserID
        ELSE
            LOG "No FCM token for User " + UserID + " - mobile app not installed"
        ENDIF

    CATCH Error
```

Systems Analysis and Design Project

```
LOG "Push notification failed: " + Error.Message
SET Delivery Status.Push = FALSE
ENDTRY
ENDIF

// =====
// UPDATE DELIVERY STATUS
// =====

// Store delivery attempts
CREATE Delivery Log Record
    NotificationID = NotificationID
    InAppDelivered = Delivery Status.InApp
    EmailDelivered = Delivery Status.Email
    SMSDelivered = Delivery Status.SMS
    PushDelivered = Delivery Status.Push
    AttemptedAt = CURRENT_DATETIME

WRITE to Delivery Log Table

// Check if at least one channel succeeded
SET Success Count = 0
IF Delivery Status.InApp = TRUE THEN INCREMENT Success Count
IF Delivery Status.Email = TRUE THEN INCREMENT Success Count
IF Delivery Status.SMS = TRUE THEN INCREMENT Success Count
IF Delivery Status.Push = TRUE THEN INCREMENT Success Count

IF Success Count = 0 THEN
    // All channels failed - queue for retry
    LOG "WARNING: All notification channels failed for " + NotificationID
    CREATE Retry Job
        NotificationID = NotificationID
        RetryAt = CURRENT_DATETIME + 5 minutes
        RetryCount = 1
    WRITE to Notification Retry Queue
ELSE
    LOG "Notification " + NotificationID + " delivered via " + Success Count + " channel(s)"
ENDIF

// Send success response
DISPLAY "Notification sent successfully"

END Process
```

Refer to:

- Table 97-98: Data Flow 47 Details
- FR-10 (Status Change Notifications)
- FR-18 (Technician Notifications)
- FR-25 (Administrator Notifications)

Decision Table:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
In-App Enabled	Y	Y	Y	N	N	N
Email Enabled	Y	Y	N	Y	Y	N
SMS Enabled	Y	N	N	Y	N	N
Priority = High	Y	Y	Y	Y	N	-
Send In-App	X	X	X	-	-	-
Send Email	X	X	-	X	X	-
Send SMS	X	-	-	X	-	-
Send Push	X	X	X	X	X	-
Store in Database	X	X	X	X	X	X
Log Delivery	X	X	X	X	X	X
Queue for Retry (if all fail)	-	-	-	-	-	X

Unresolved Issues: None

4.3.6. Summary of All Process Specifications

This document contains **20 comprehensive process specifications** covering all major system processes:

Fragment 1: User Management (4 Processes)

1. **1.1 User Registration** - Account creation with validation and email verification
2. **1.2 User Authentication** - Login with JWT, CAPTCHA, and session management
3. **1.3 Password Recovery** - Reset forgotten passwords with verification codes
4. **1.4 Account Management** - Profile updates and admin user management

Fragment 2: Maintenance Requests (5 Processes)

5. **2.1 Create Maintenance Ticket** - Submit tickets with duplicate detection
6. **2.2 View/Search Tickets** - Query and filter tickets by various criteria
7. **2.3 Update Ticket Status** - Manage ticket lifecycle and assignments
8. **2.4 Manage Comments & Evidence** - Add comments and upload completion evidence
9. **2.5 Send Notifications** - Multi-channel notification dispatcher

Fragment 3: Assignment & Tasks (4 Processes)

10. **3.1 View Pending/Unassigned Tickets** - Display unassigned tickets for admins
11. **3.2 Assign Ticket & Set Priority** - Assign technicians and set priorities
12. **3.3 Notify Technician** - Send assignment notifications to technicians
13. **3.4 Update Assignment Status** - Handle accept/decline and reassignment

Fragment 4: Analytics & Reporting (4 Processes)

14. **4.1 Record Maintenance Completion** - Archive completed work with metrics
15. **4.2 Update Maintenance History** - Store historical records for analysis
16. **4.3 Generate Reports & Metrics** - Compile data and calculate KPIs
17. **4.4 Provide Analytics Dashboard / Export** - Display dashboards and export reports

Systems Analysis and Design Project

Fragment 5: Notification Management (3 Processes)

18. **5.1 Receive Trigger** - Capture notification events and identify affected users
19. **5.2 Prepare Notification** - Construct personalized messages with ticket context
20. **5.3 Send Notification** - Deliver via multiple channels (in-app, email, SMS, push)

Systems Analysis and Design Project

4.4. Object-Oriented Analysis

4.4.1. Activity Diagrams

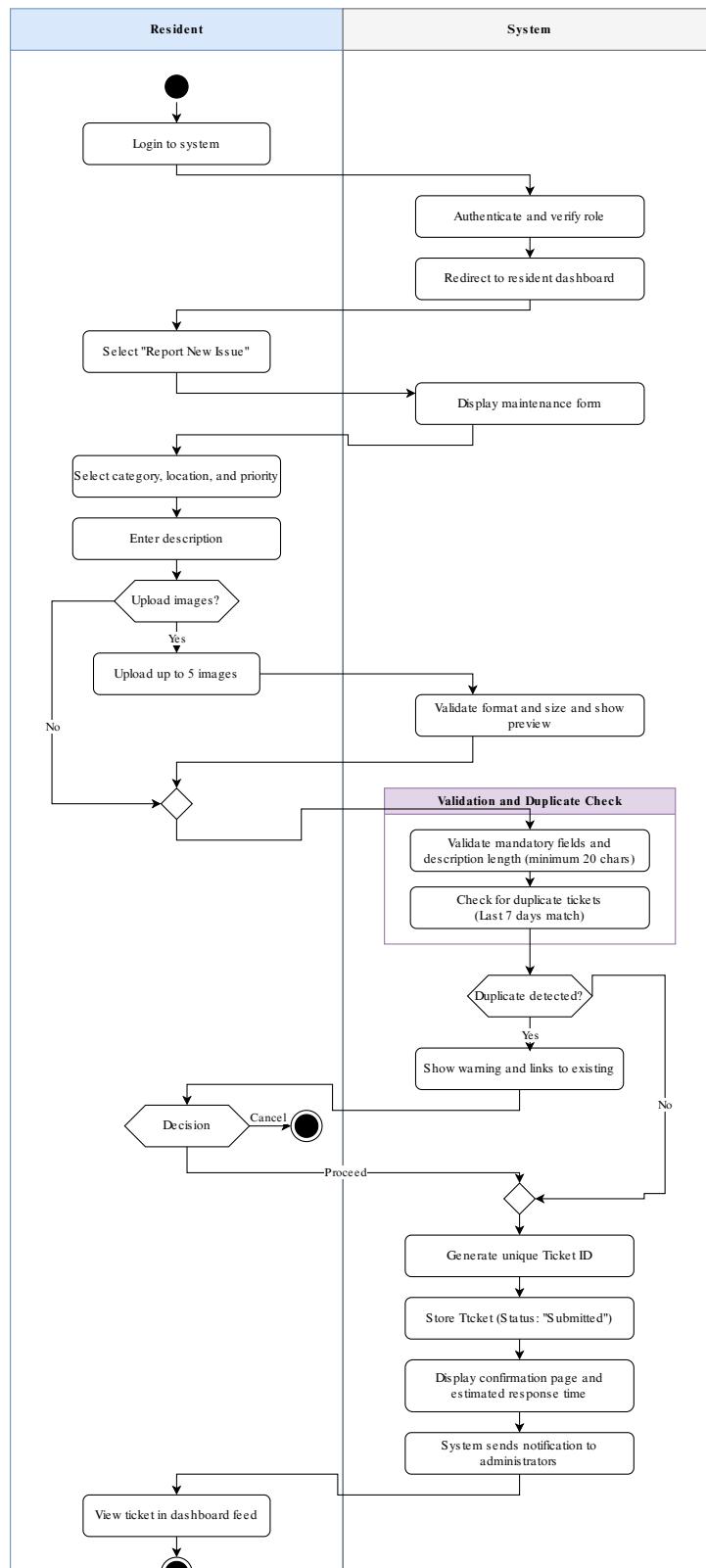


Figure 54: Resident Activity Diagram

Systems Analysis and Design Project

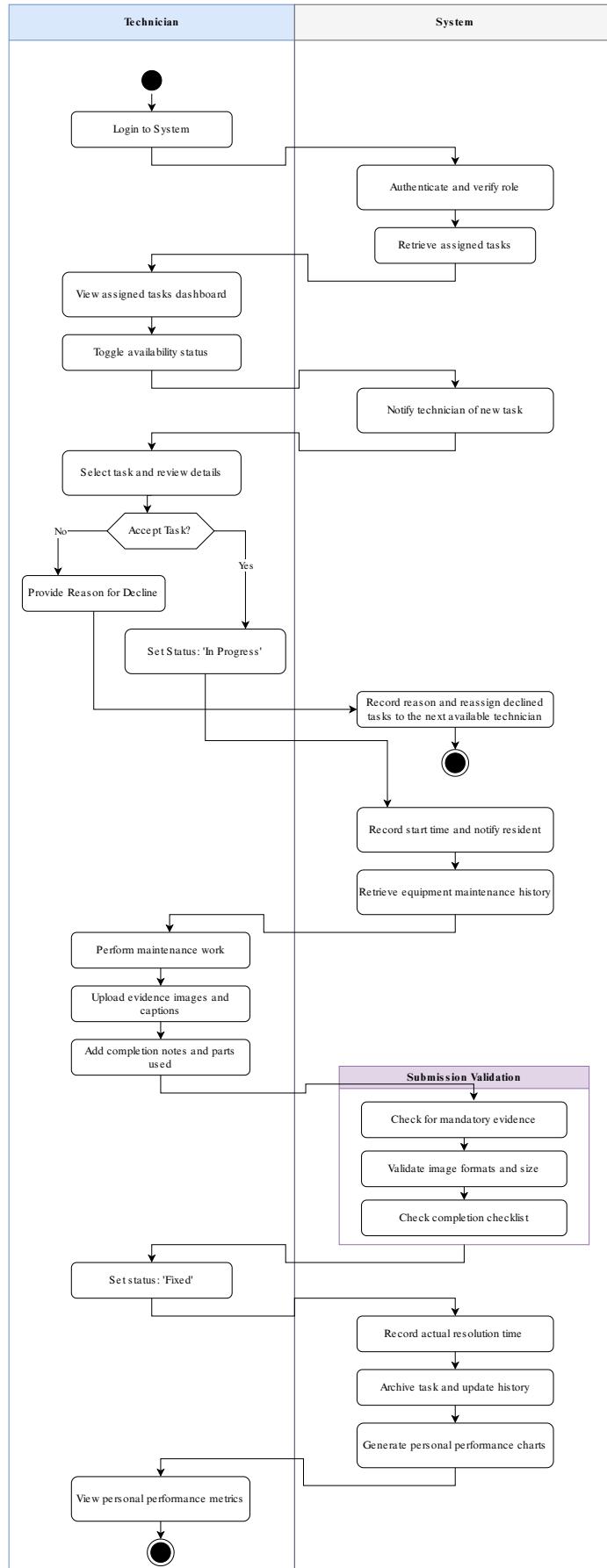


Figure 55: Technician Activity Diagram
185

4.5. System Architecture Design

4.5.1. Architectural Style

4.5.1.1. 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

4.5.1.2. 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.

4.5.2. Technology Stack

4.5.2.1. Frontend Technologies

4.5.2.1.1. 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

4.5.2.1.2. 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

Systems Analysis and Design Project

4.5.2.2. Backend Technologies

4.5.2.2.1. 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

4.5.2.2.2. 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

4.5.2.3. 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

4.5.2.4. 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)

Systems Analysis and Design Project

4.5.3. Component/Module-Level View

4.5.3.1. System Components Diagram

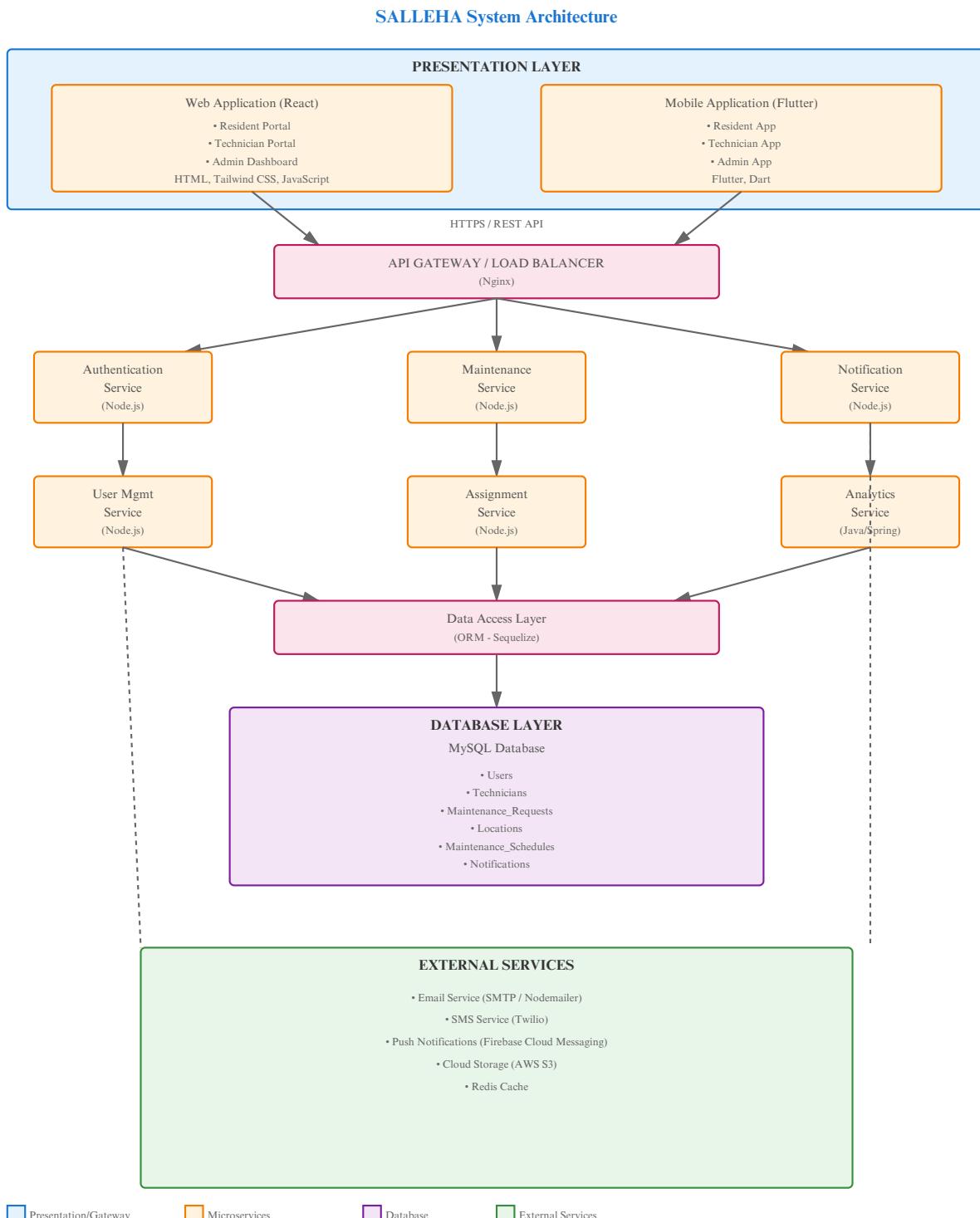


Figure 56: System Components Diagram

4.5.3.2. Core Modules and Components

4.5.3.2.1. MODULE 1: Authentication & Authorization Module

Responsibilities:

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

Systems Analysis and Design Project

- 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

4.5.3.2.2. 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

4.5.3.2.3. 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)

Systems Analysis and Design Project

- **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

4.5.3.2.4. 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

4.5.3.2.5. 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

4.5.3.2.6. MODULE 6: Notification & Communication Module

Responsibilities:

- Real-time push notifications (FR-10, FR-18, FR-25)
- Email notifications for status changes

Systems Analysis and Design Project

- 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

4.5.3.2.7. 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

4.5.3.2.8. 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

Systems Analysis and Design Project

4.5.3.2.9. 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

4.5.4. 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

4.5.5. Architectural Design Decisions

4.5.5.1. Why Three-Tier Architecture?

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

4.5.5.2. Why RESTful API?

- Platform Independence:** Web and mobile apps consume the same endpoints
- Statelessness:** Each request contains all necessary information (JWT tokens)
- Caching:** HTTP caching mechanisms improve performance (NFR-1)

Systems Analysis and Design Project

4. **Industry Standard:** Well-documented and widely supported

4.5.5.3. 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)

4.5.5.4. 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

4.5.5.5. 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

4.5.6. Security Architecture

4.5.6.1. 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

4.5.6.2. 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)

4.5.7. 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)

Systems Analysis and Design Project

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

4.5.8. Deployment Architecture

4.5.8.1. Production Environment

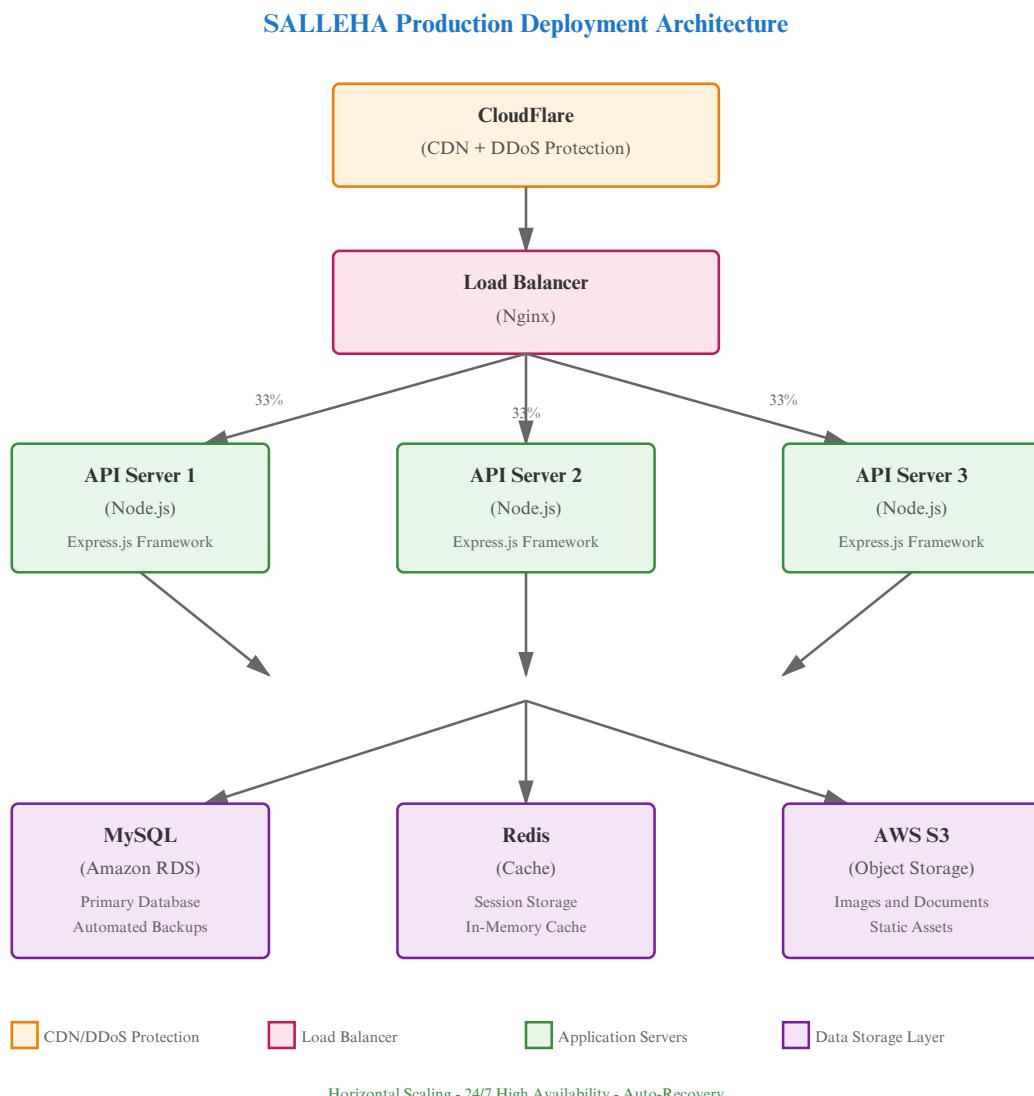


Figure 57: Deployment Diagram

4.5.8.2. 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

Systems Analysis and Design Project

4.5.9. 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

4.5.10. 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

4.5.11. 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

4.5.12. 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

Systems Analysis and Design Project

4.6. Object-Oriented Design

4.6.1. Class Diagram

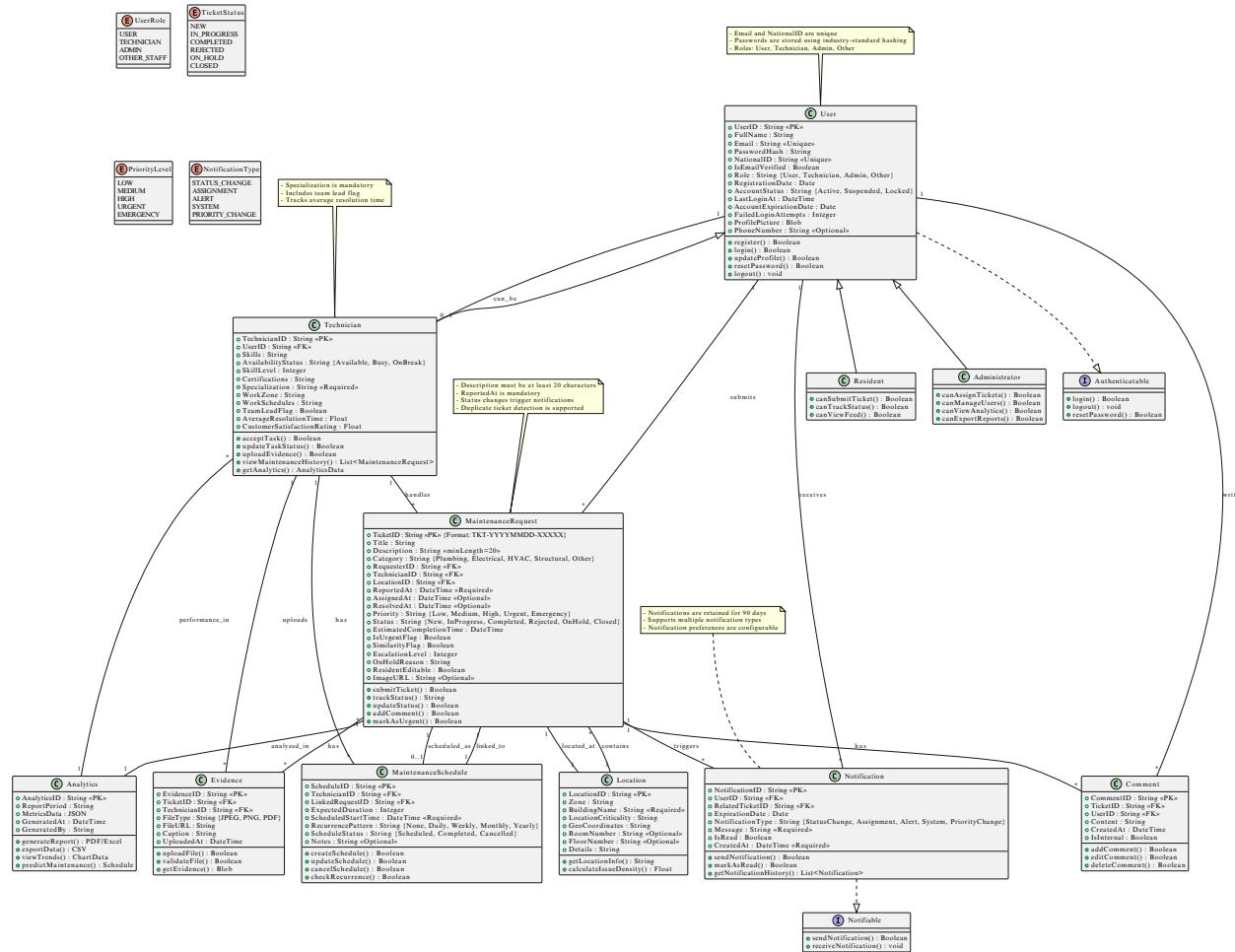


Figure 58: Class Diagram

Systems Analysis and Design Project

4.6.2. Sequence Diagrams

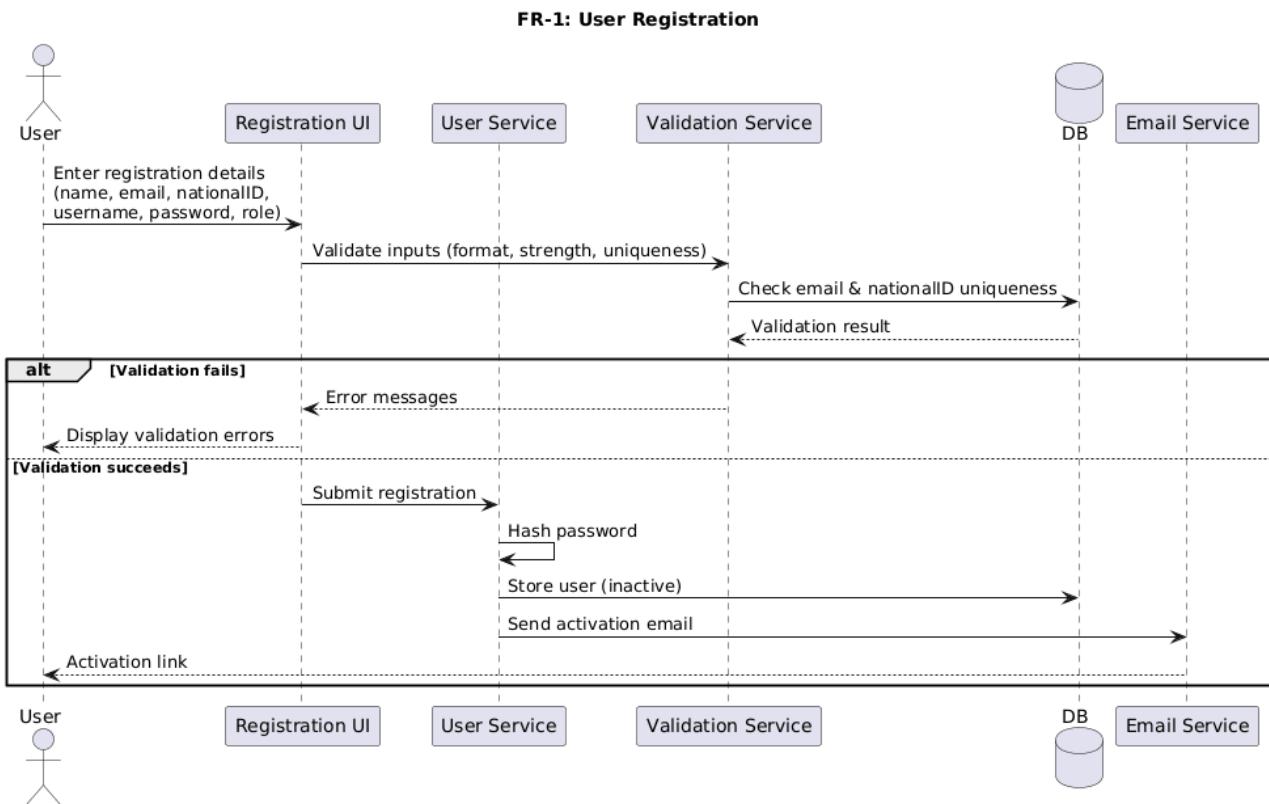


Figure 59: FR-1 Sequence Diagram

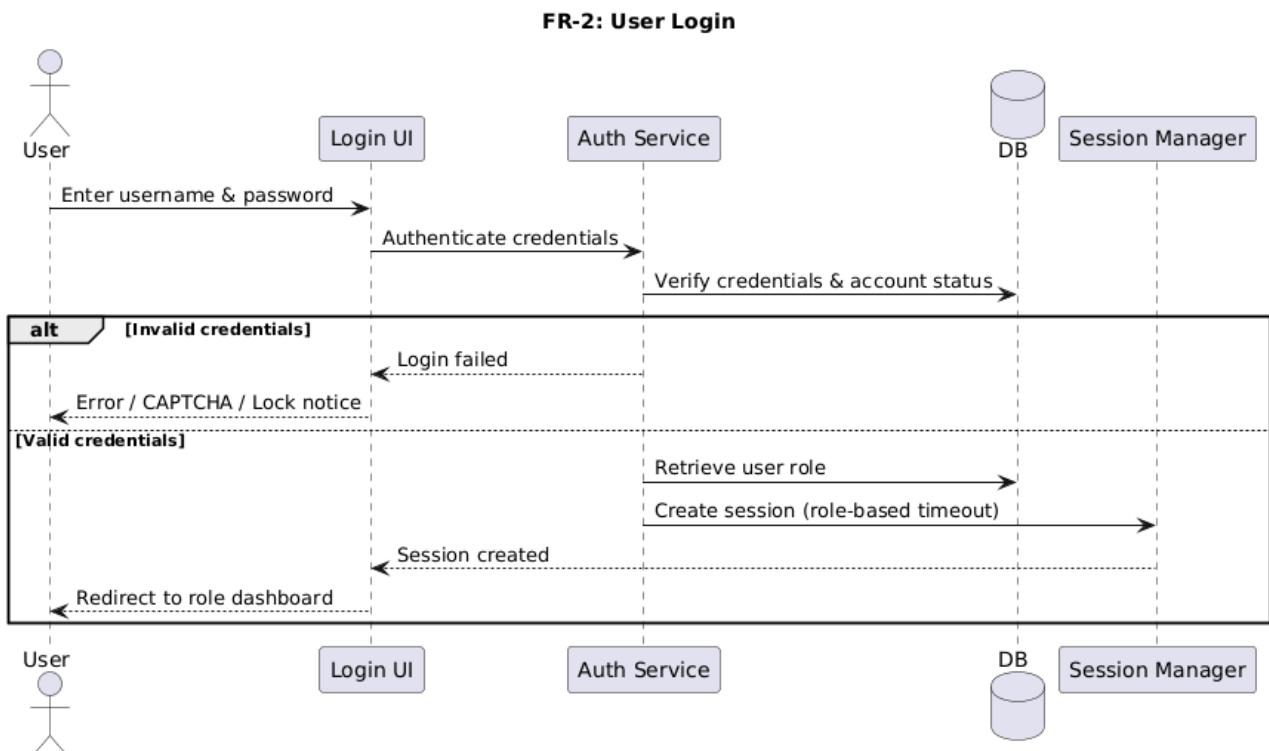


Figure 60: FR-2 Sequence Diagram

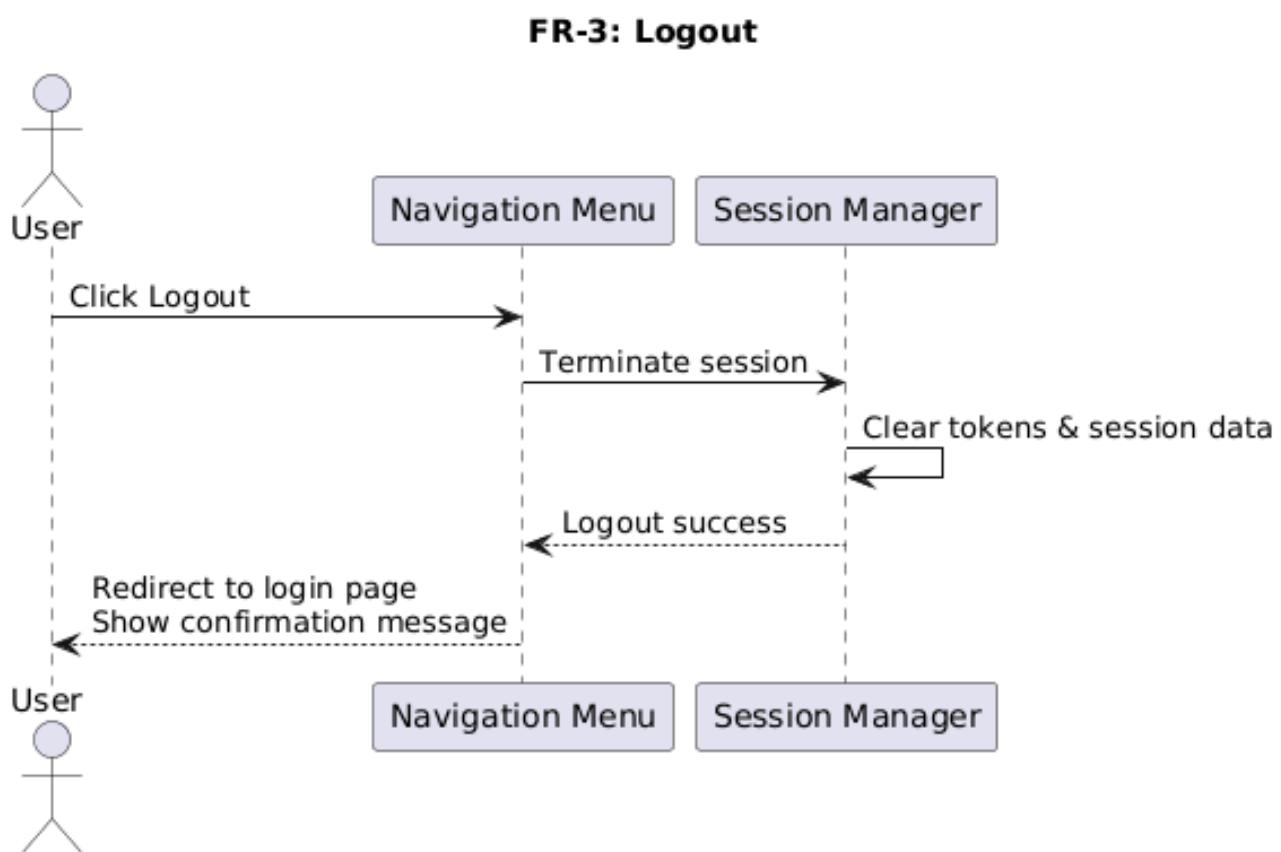


Figure 61: FR-3 Sequence Diagram

Systems Analysis and Design Project

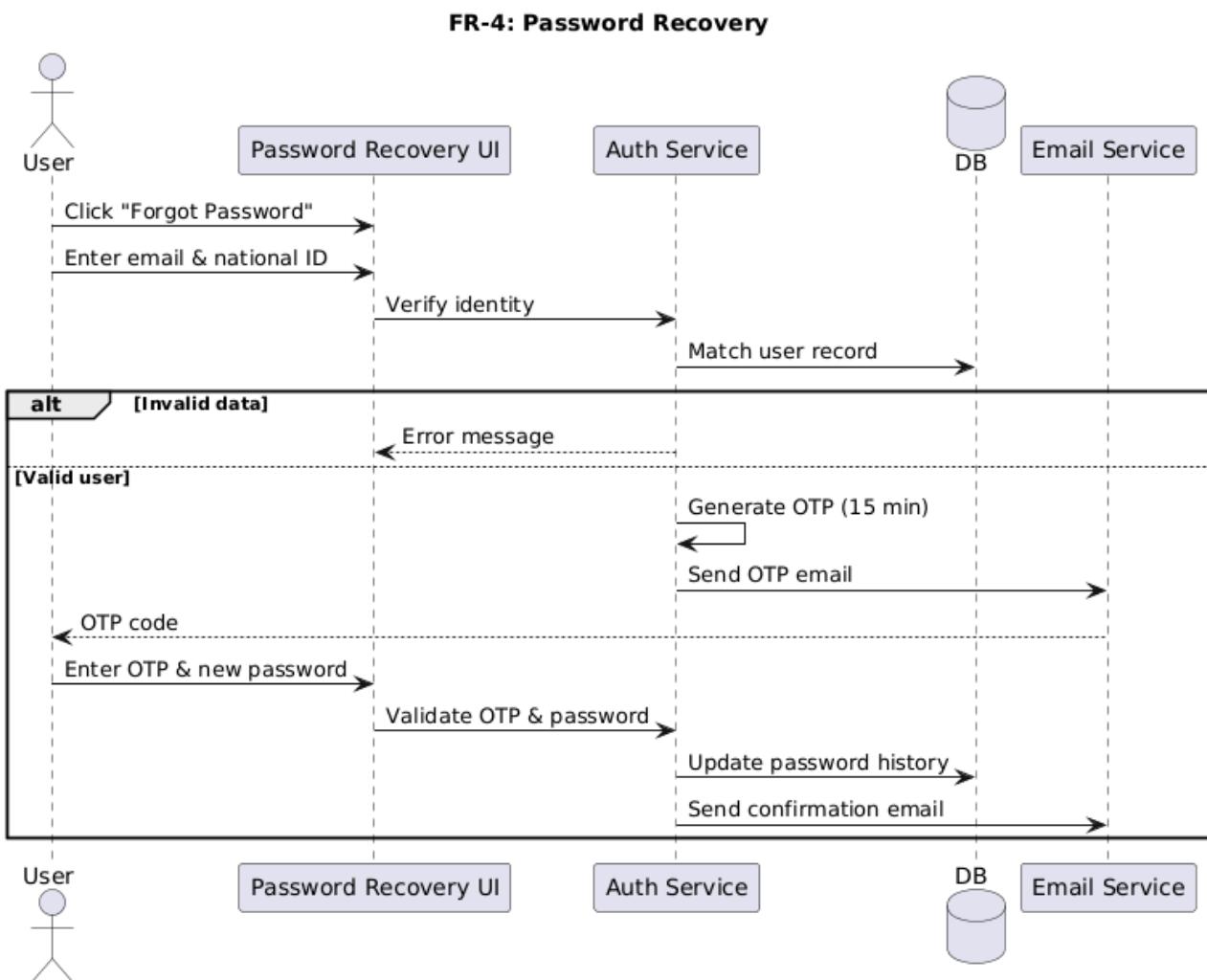


Figure 62: FR-4 Sequence Diagram

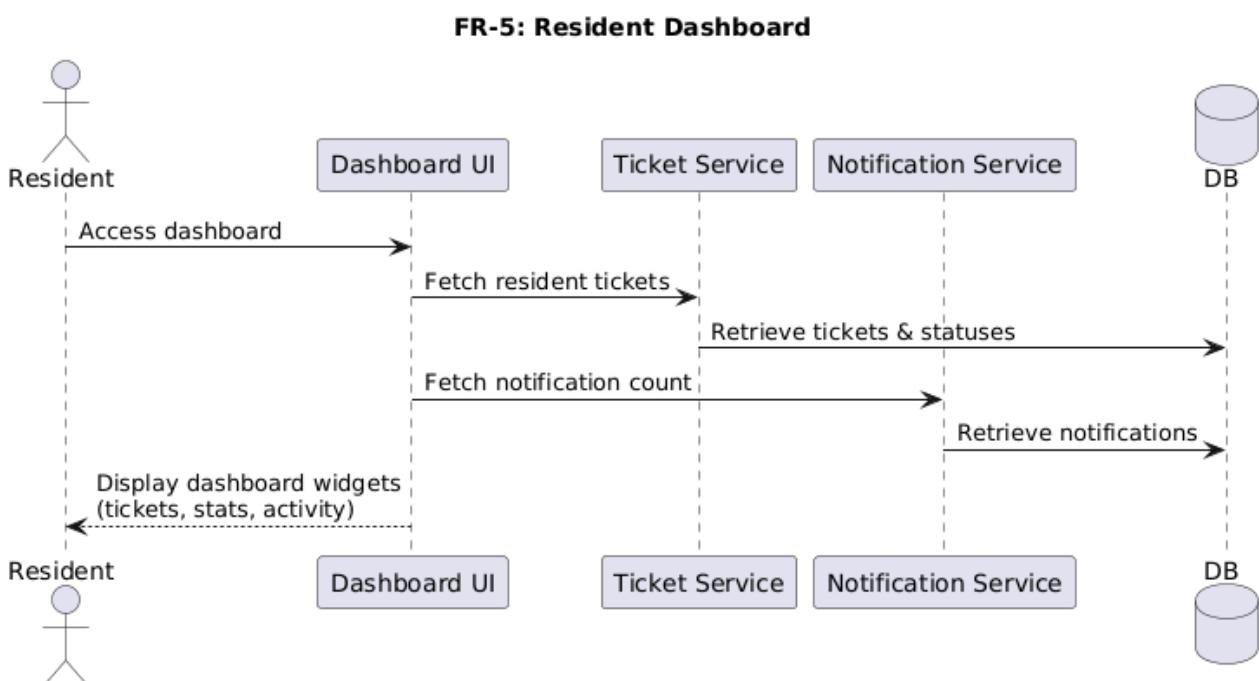


Figure 63: FR-5 Sequence Diagram

FR-6: Resident Navigation Menu

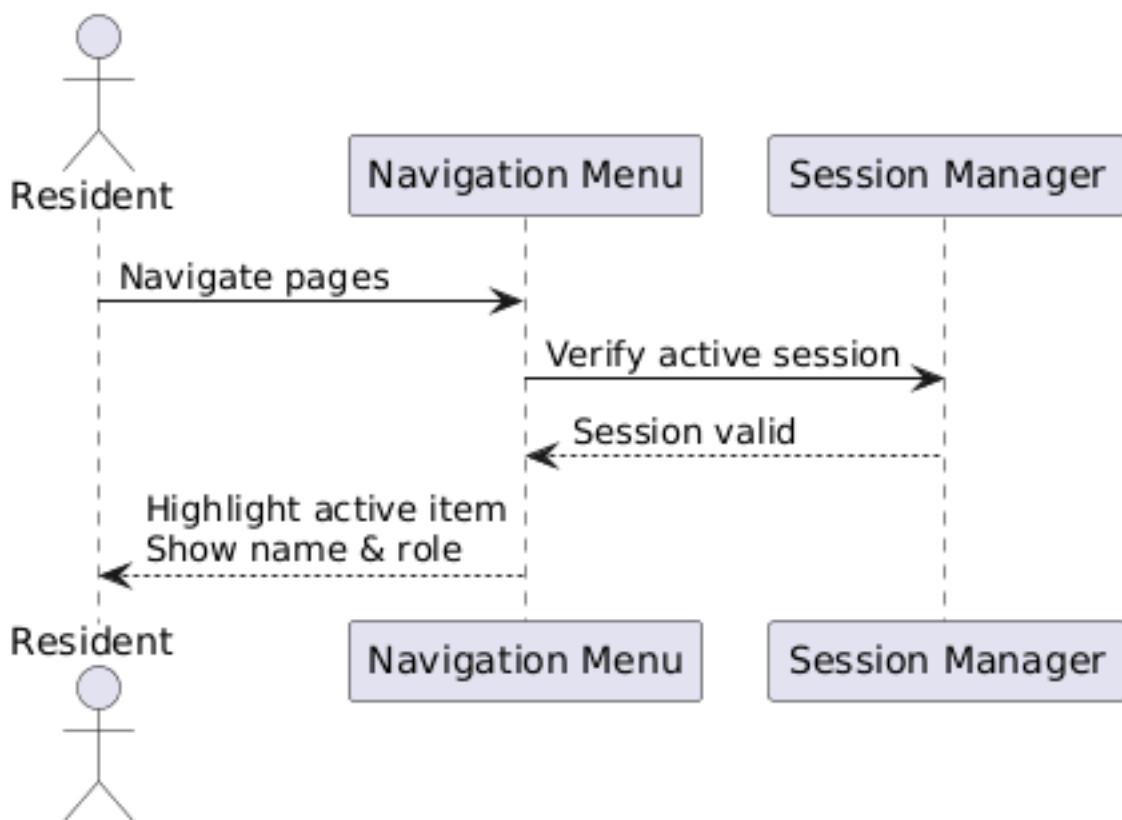


Figure 64: FR-6 Sequence Diagram

FR-7: Maintenance Issue Feed

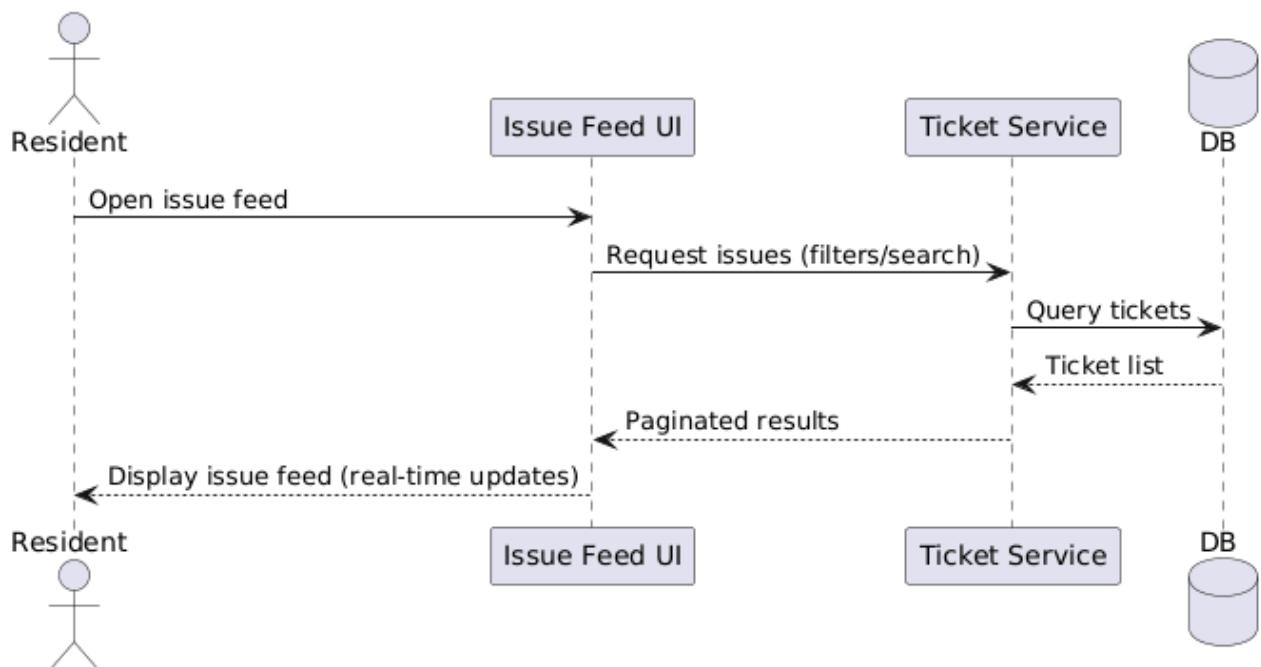


Figure 65: FR-7 Sequence Diagram

Systems Analysis and Design Project

FR-8: Open Maintenance Ticket

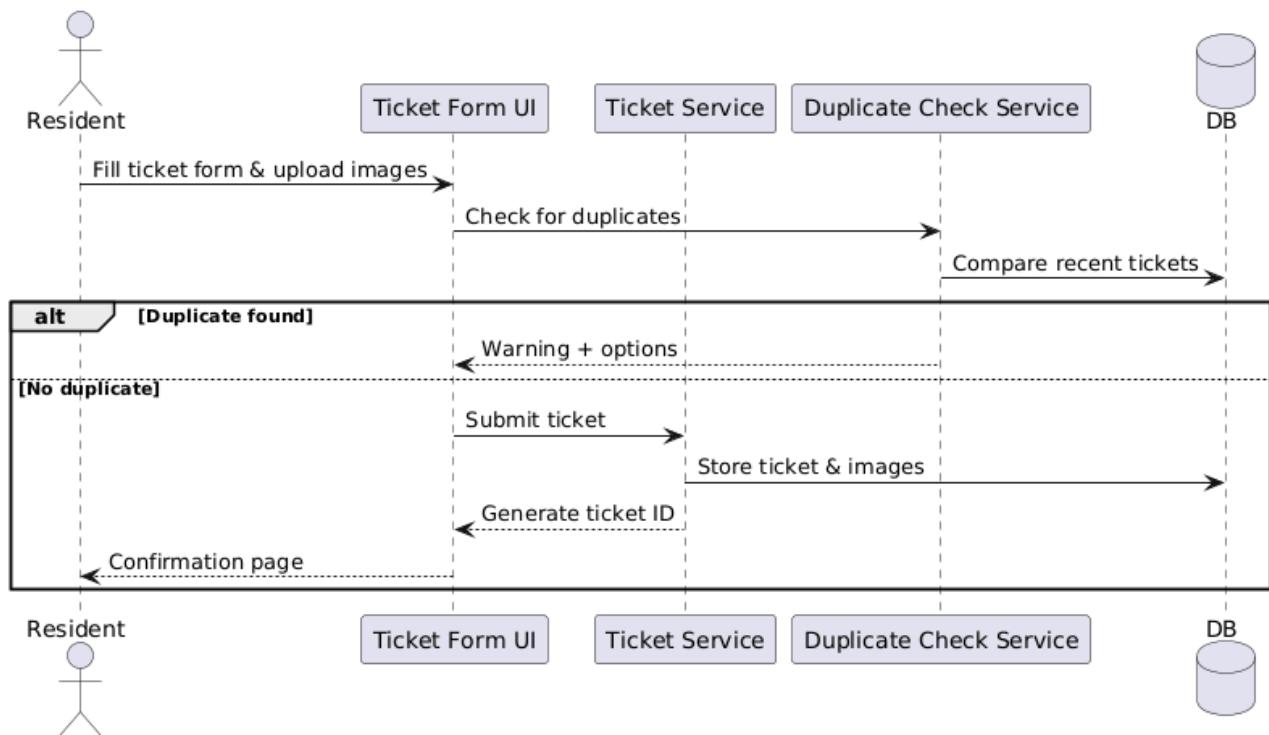


Figure 66: FR-8 Sequence Diagram

FR-9: Ticket Status Tracking

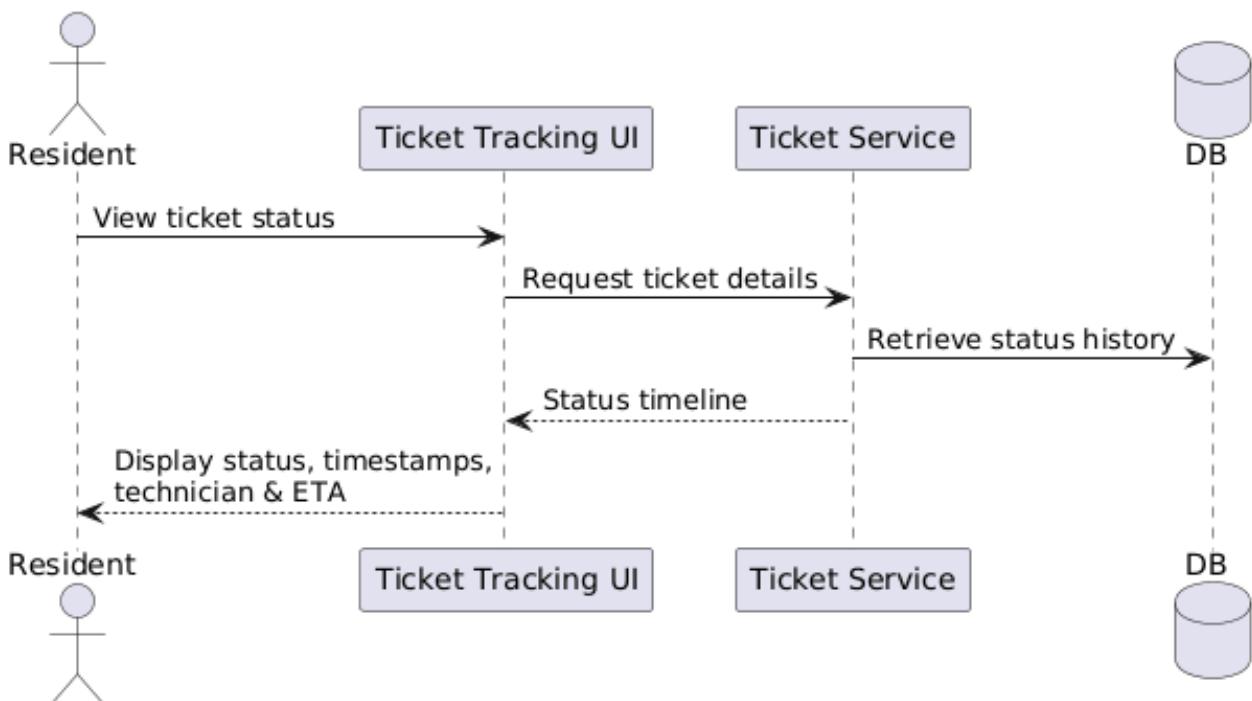


Figure 67: FR-9 Sequence Diagram

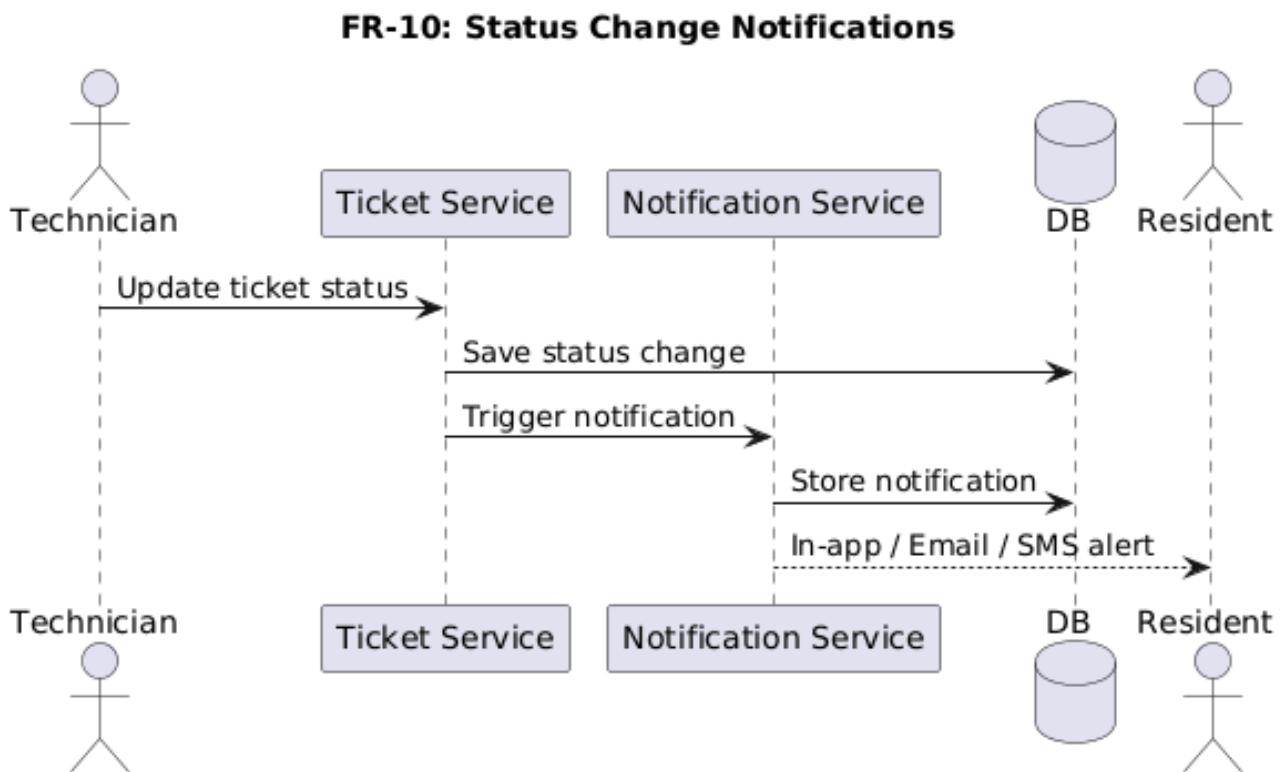


Figure 68: FR-10 Sequence Diagram

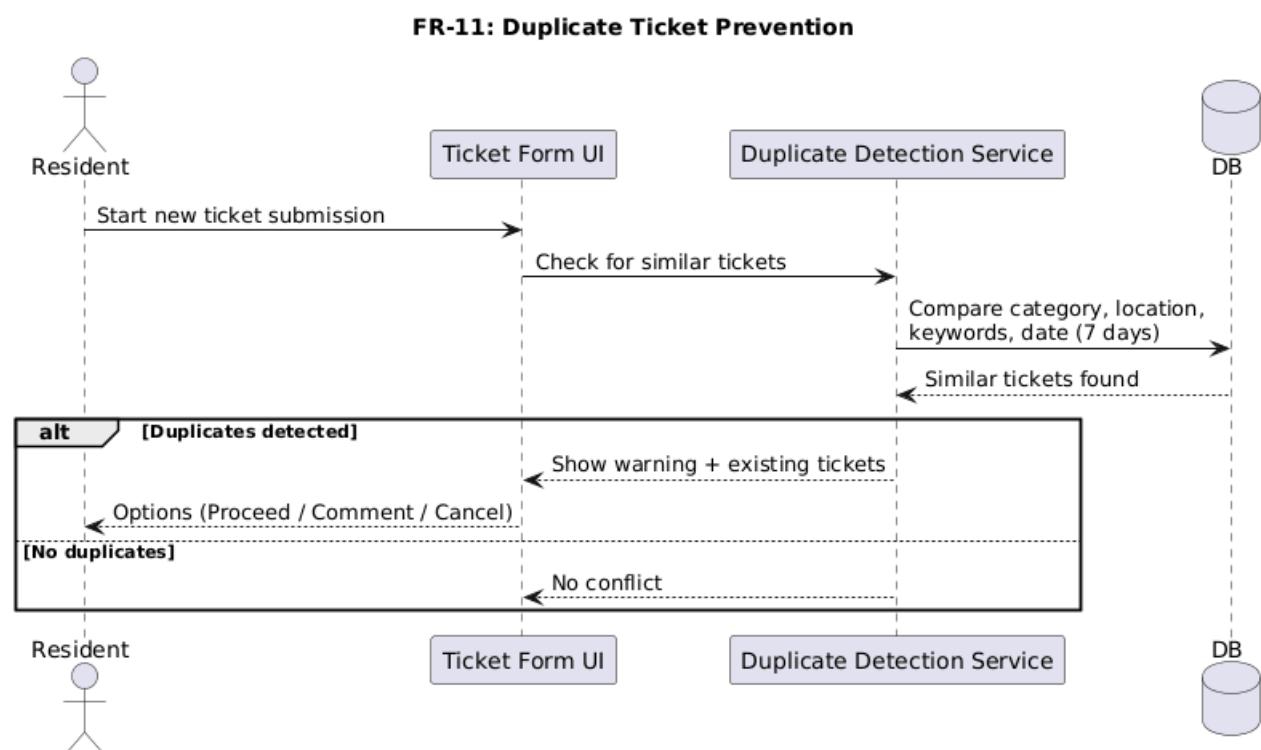


Figure 69: FR-11 Sequence Diagram

Systems Analysis and Design Project

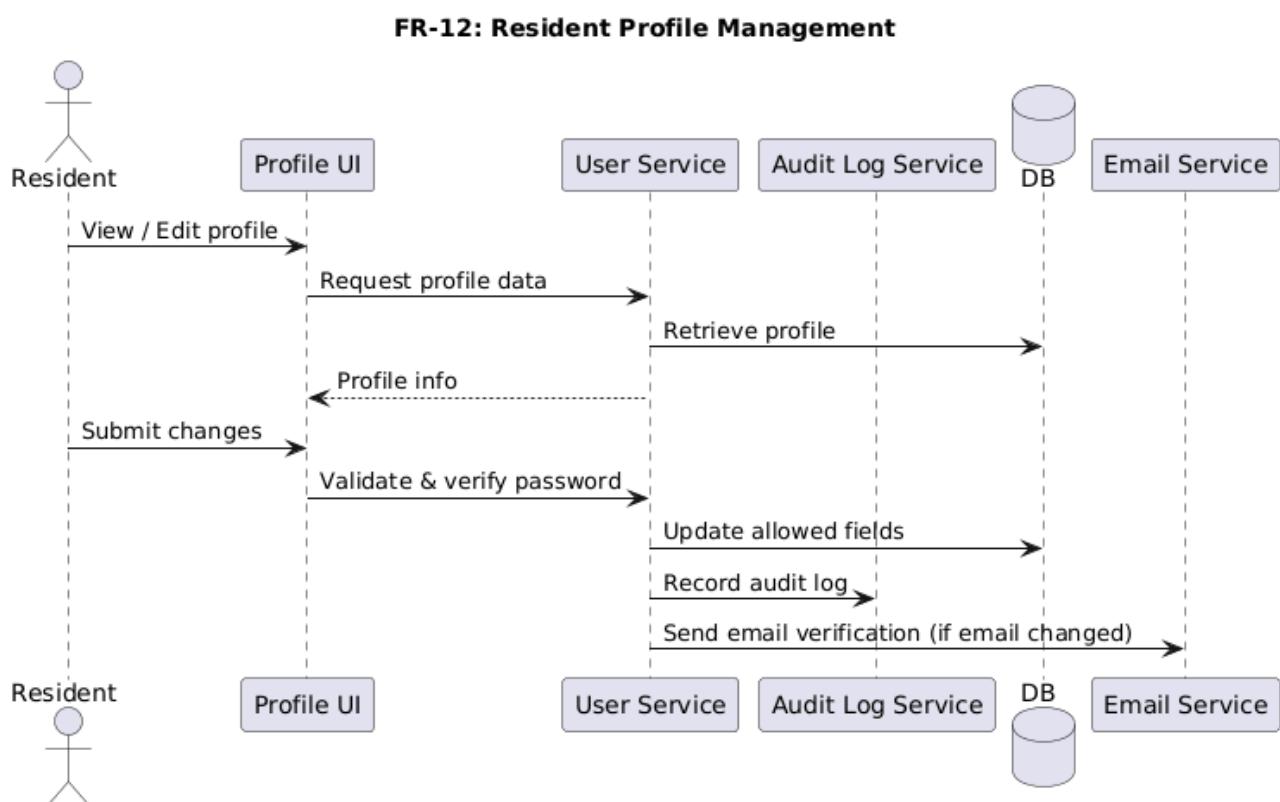


Figure 70: FR-12 Sequence Diagram

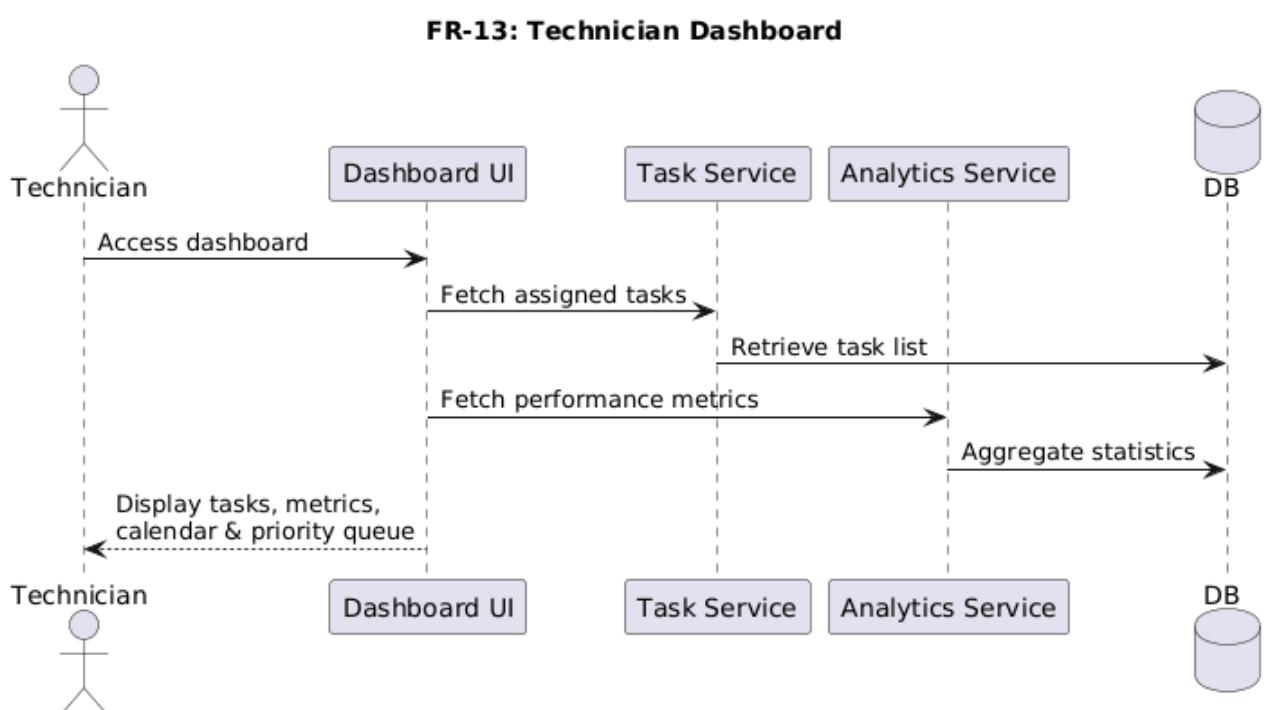


Figure 71: FR-13 Sequence Diagram

FR-14: Technician Navigation Menu

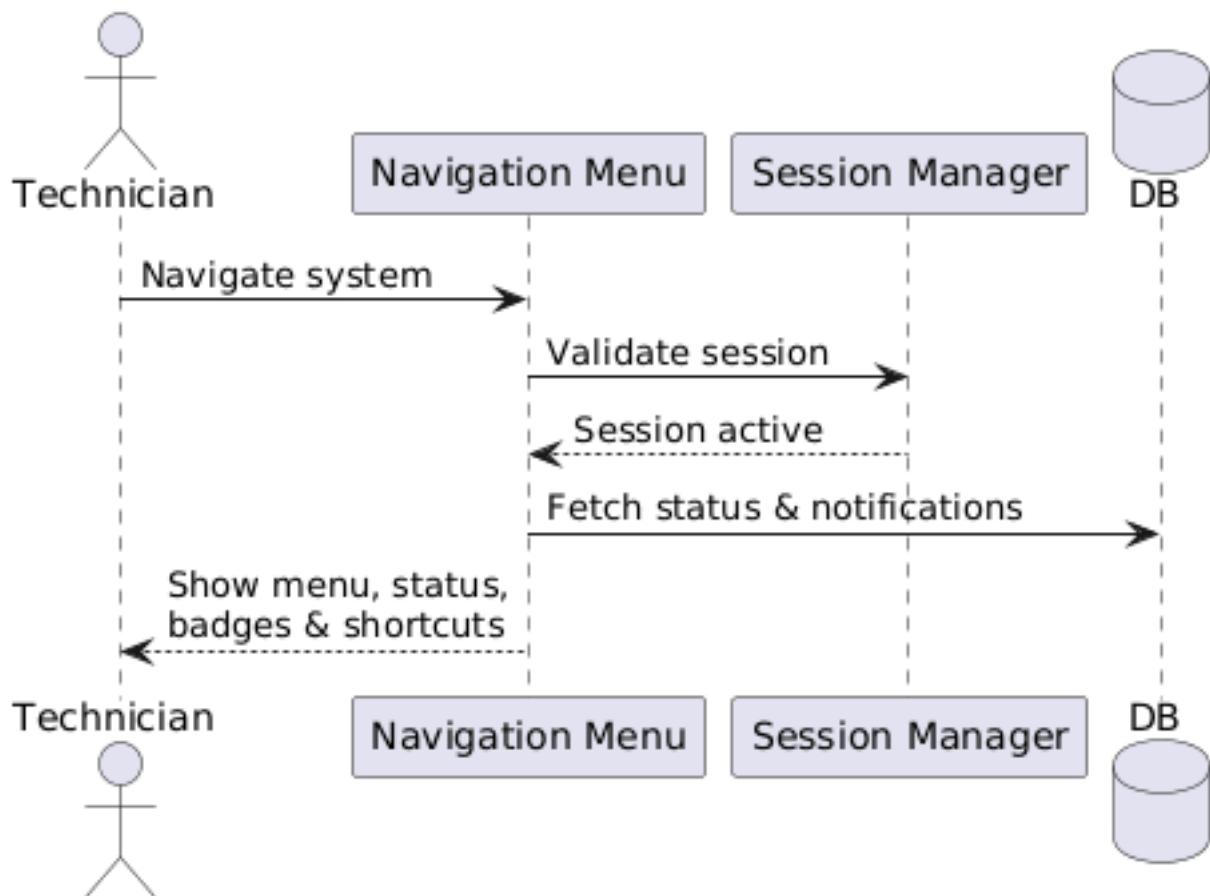


Figure 72: FR-14 Sequence Diagram

Systems Analysis and Design Project

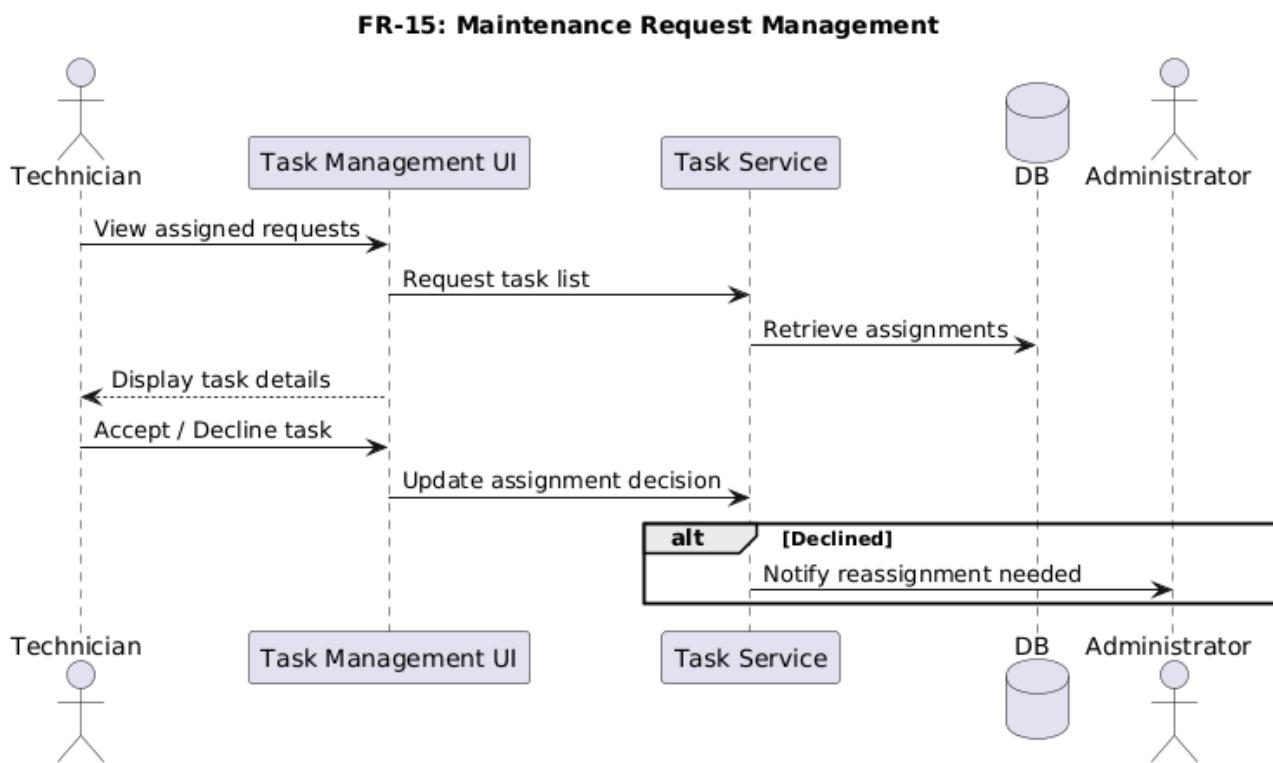


Figure 73: FR-15 Sequence Diagram

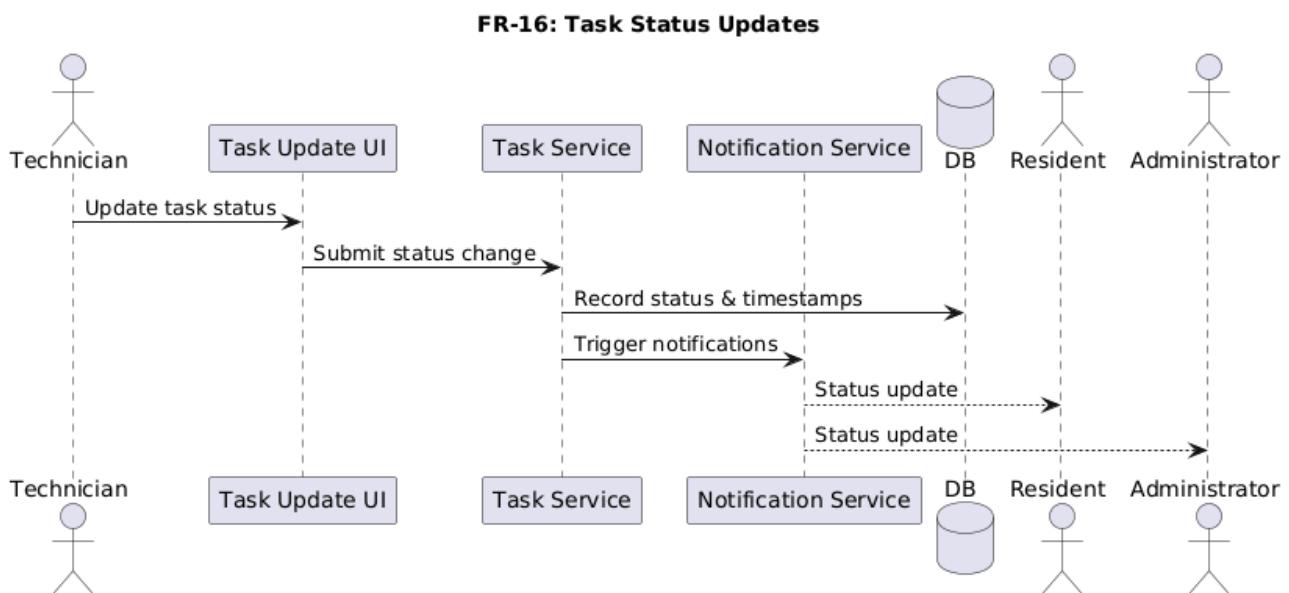


Figure 74: FR-16 Sequence Diagram

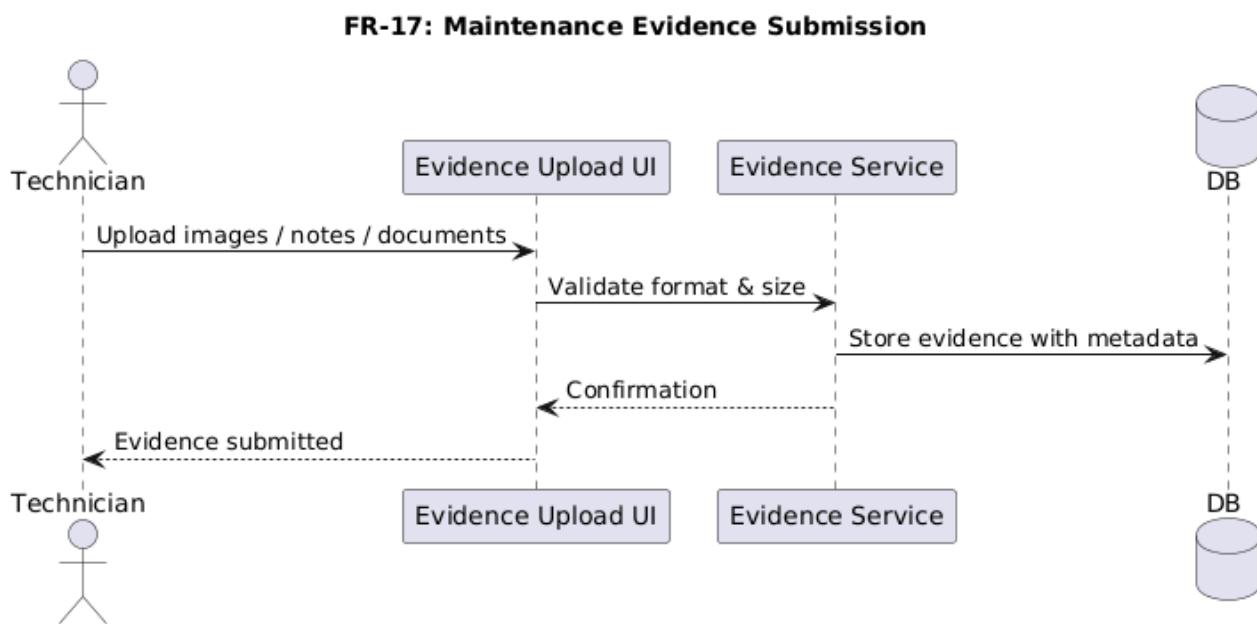


Figure 75: FR-17 Sequence Diagram

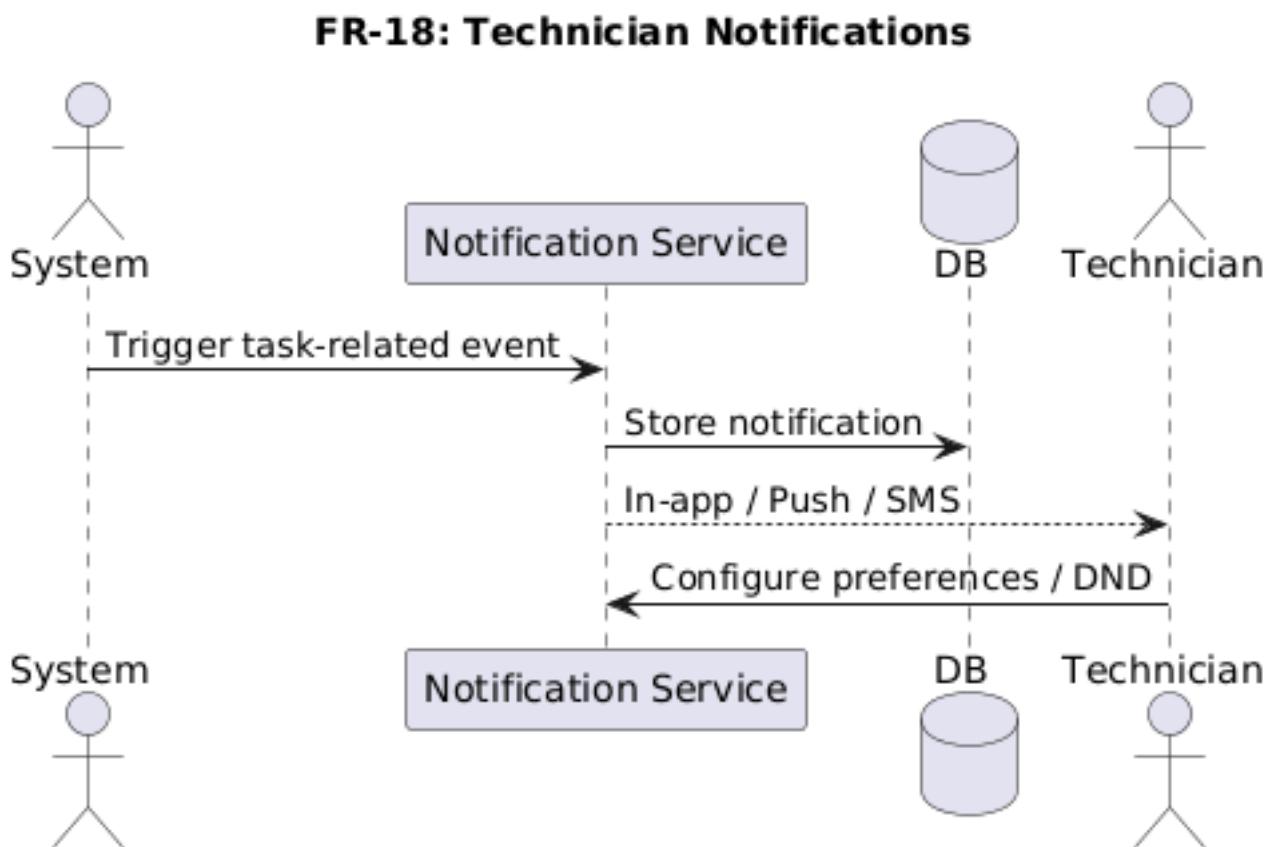


Figure 76: FR-18 Sequence Diagram

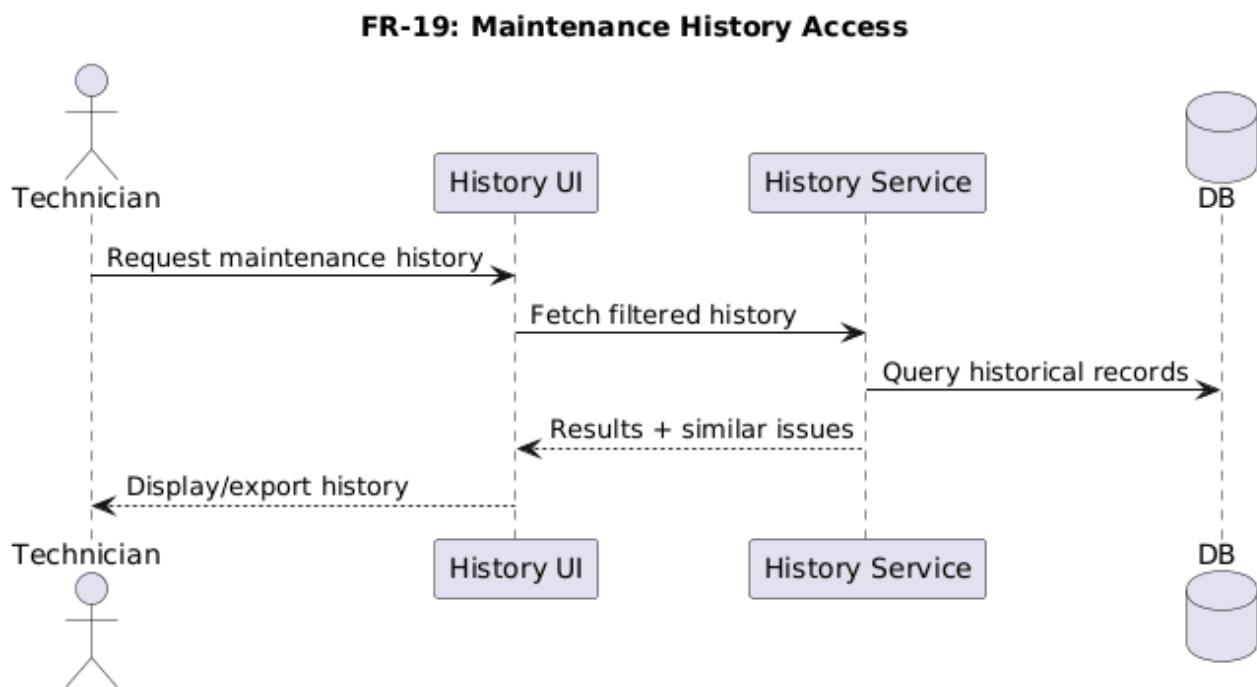


Figure 77: FR-19 Sequence Diagram

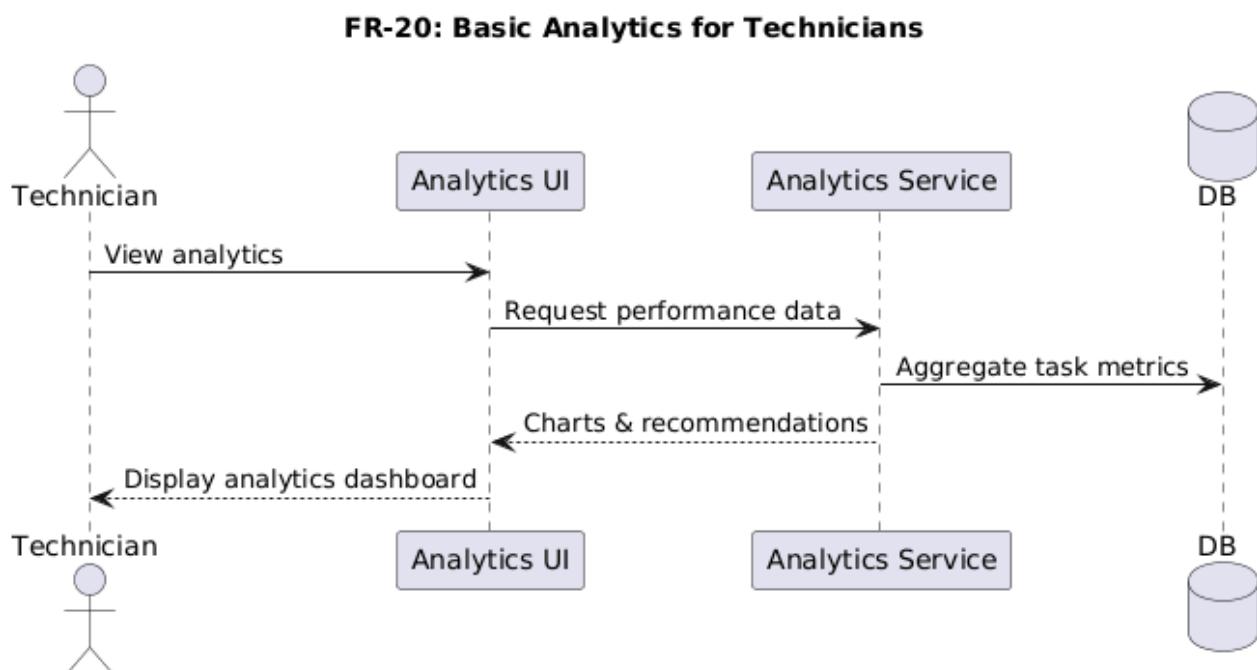


Figure 78: FR-20 Sequence Diagram

Systems Analysis and Design Project

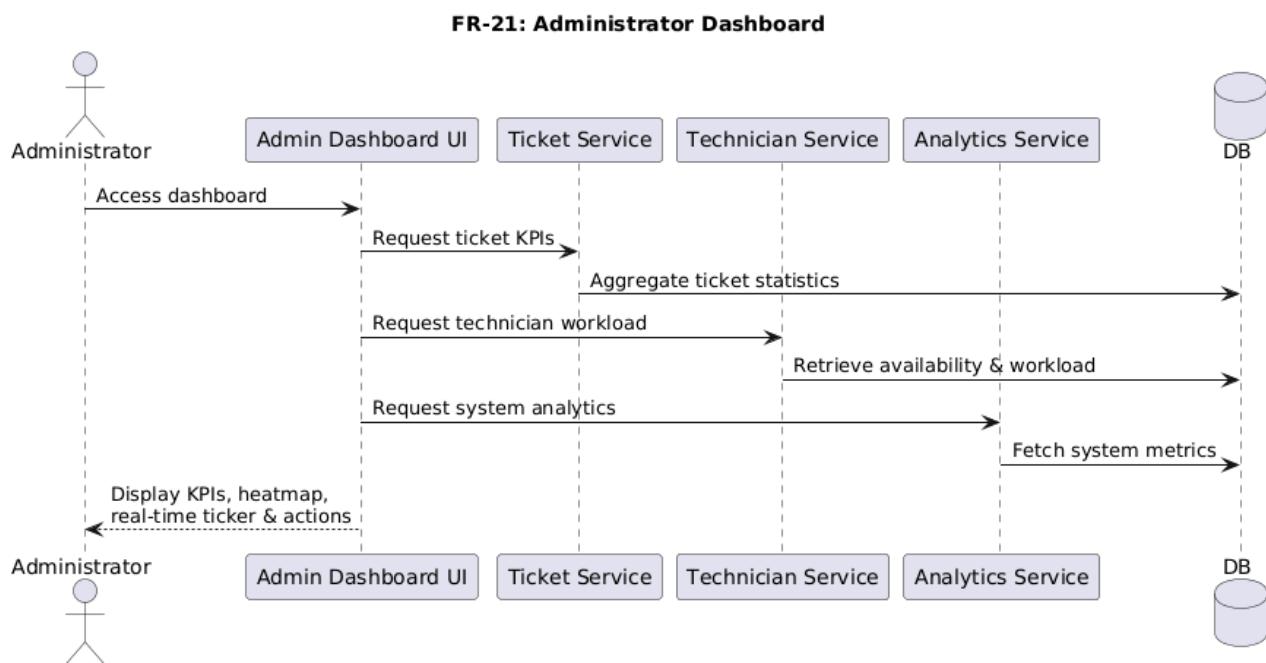


Figure 79: FR-21 Sequence Diagram

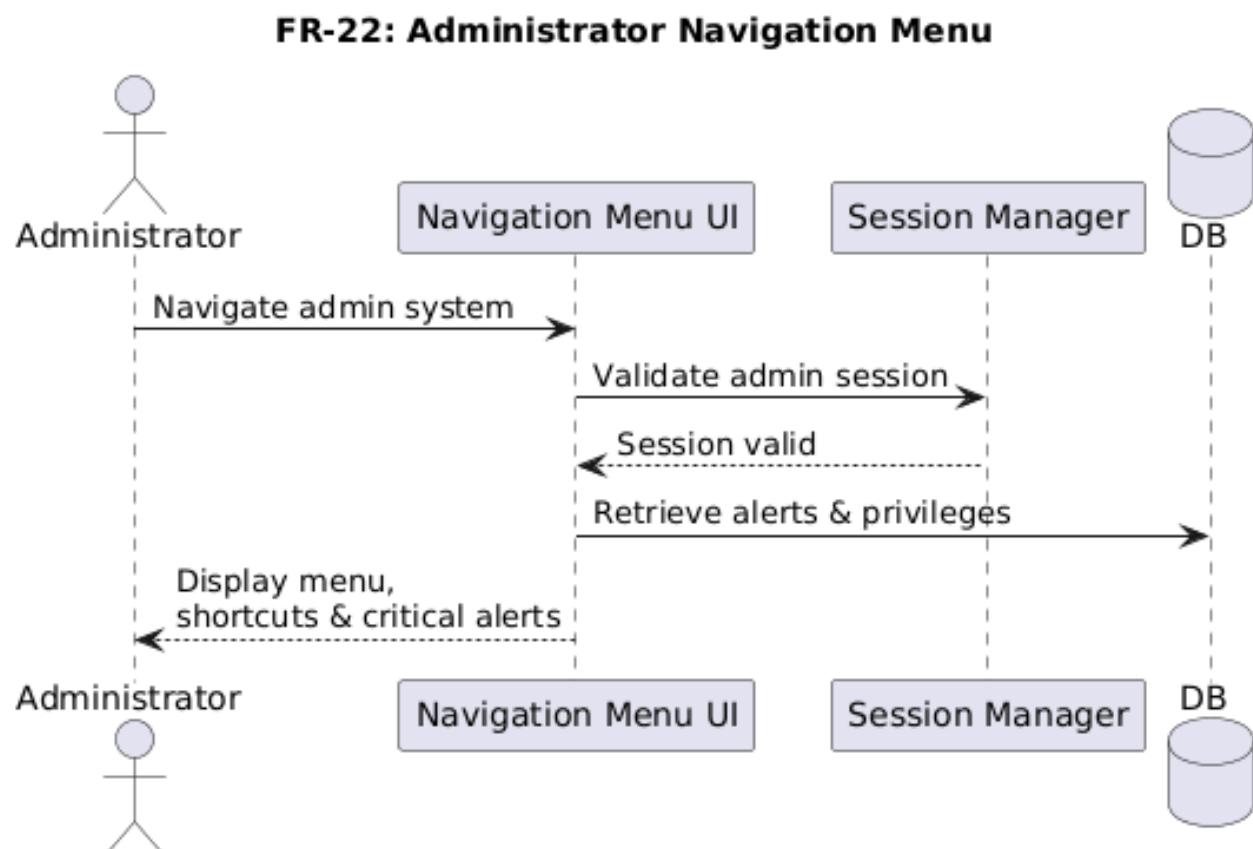


Figure 80: FR-22 Sequence Diagram

Systems Analysis and Design Project

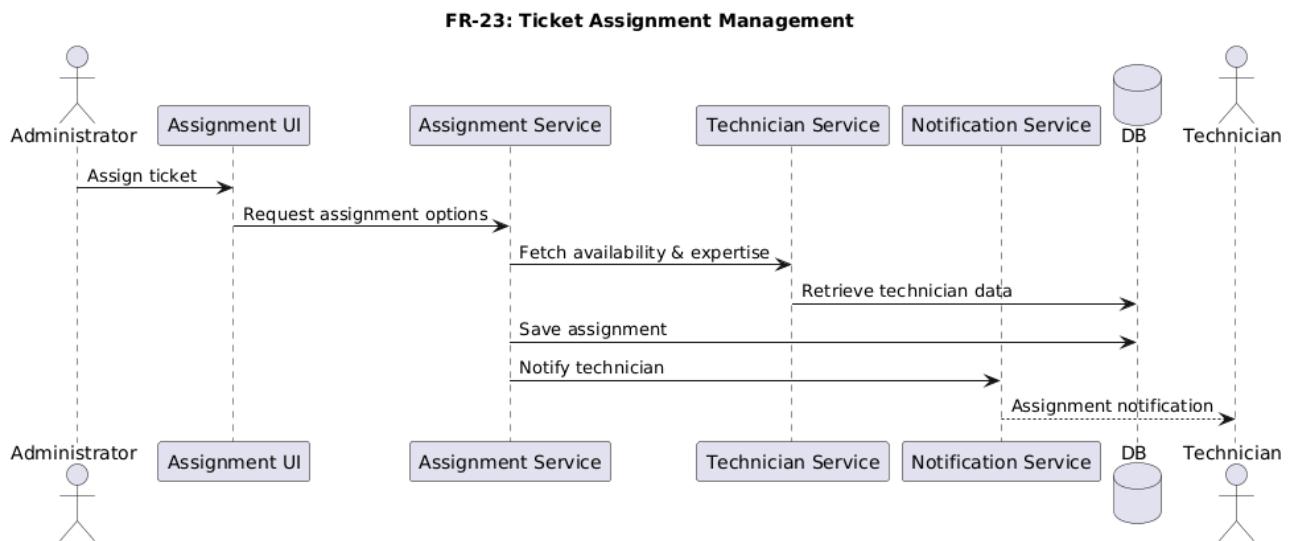


Figure 81: FR-23 Sequence Diagram

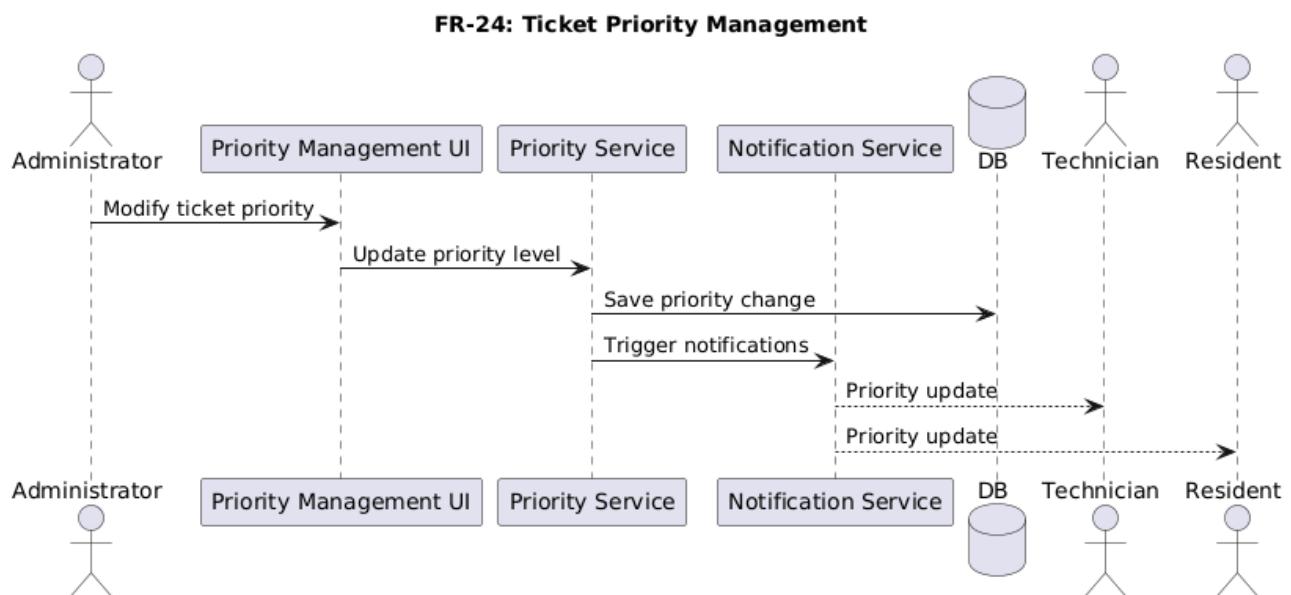


Figure 82: FR-24 Sequence Diagram

FR-25: Administrator Notifications

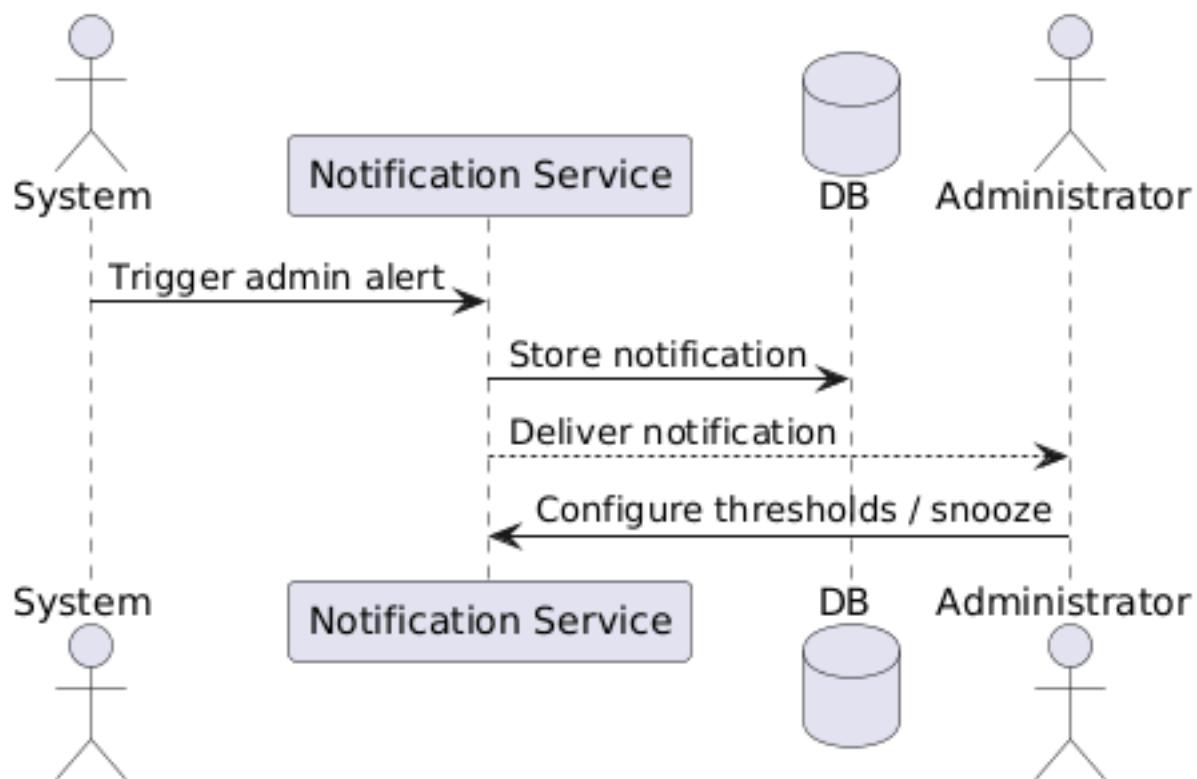


Figure 83: FR-25 Sequence Diagram

FR-26: Analytics and Reporting

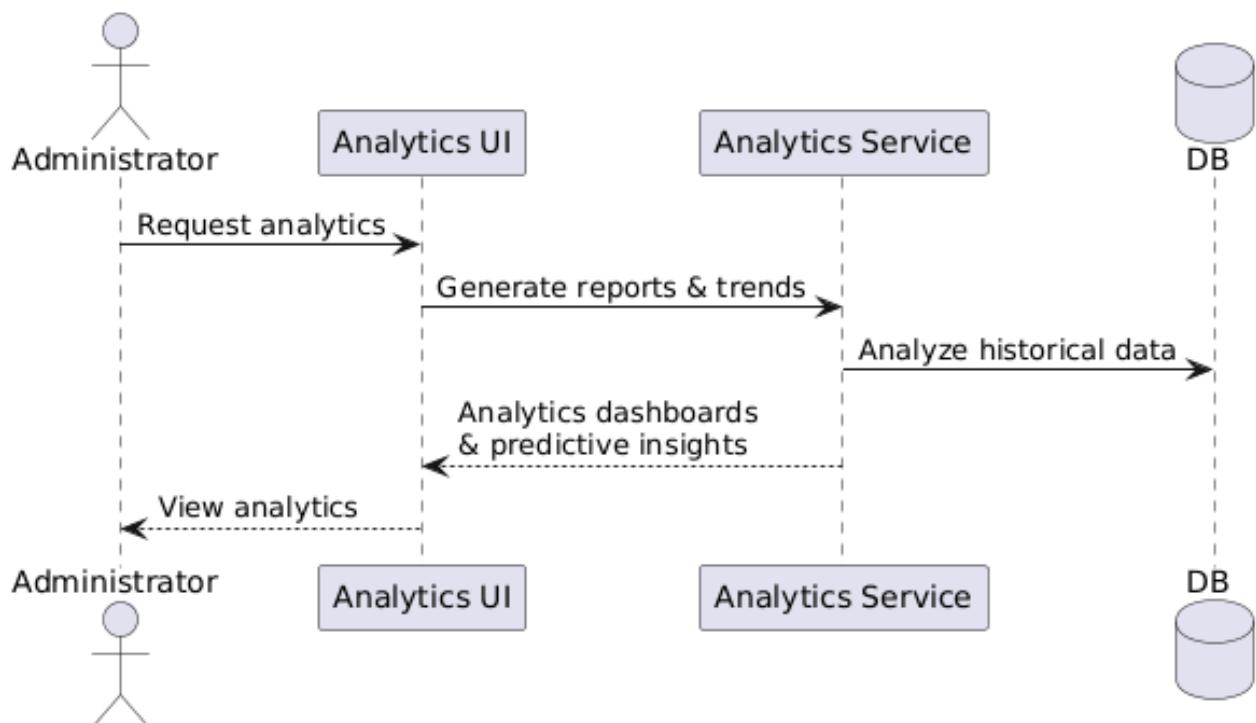


Figure 84: FR-26 Sequence Diagram

Systems Analysis and Design Project

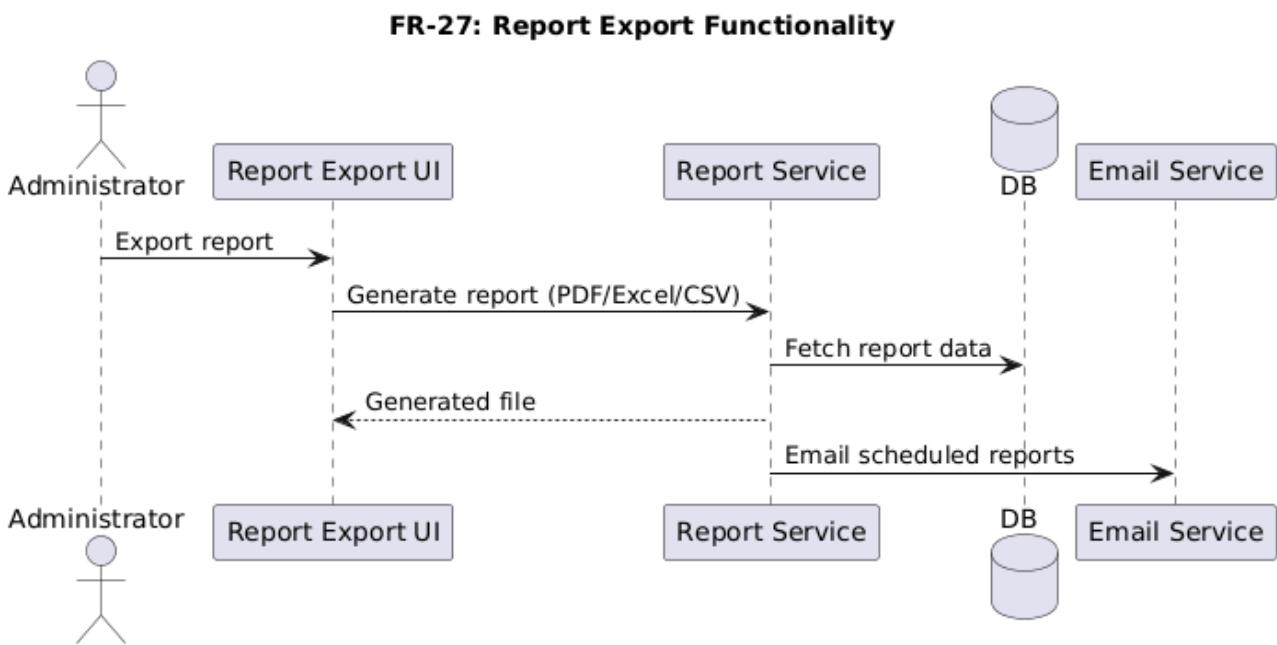


Figure 85: FR-27 Sequence Diagram

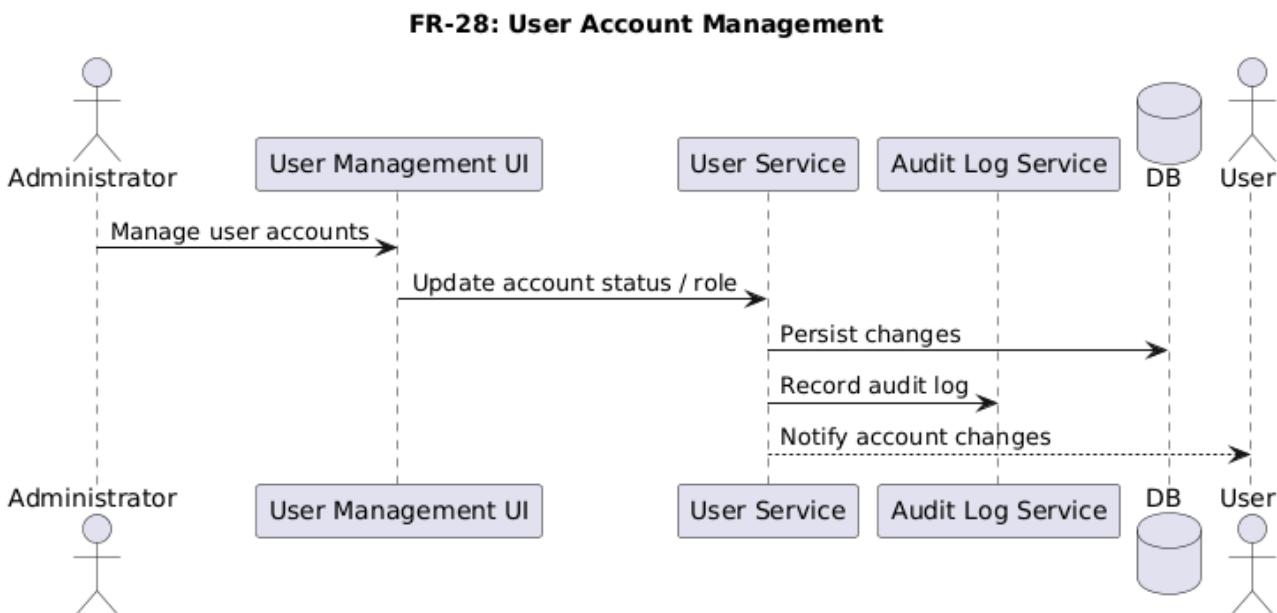


Figure 86: FR-28 Sequence Diagram

4.6.3. 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)

Systems Analysis and Design Project

- 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:

- + register(fullName: String, email: String, nationalID: String, username: String, password: String, role: String): Boolean
- + login(username: String, password: String, captcha: String): Boolean
- + logout(): Boolean
- + resetPassword(email: String, nationalID: String): Boolean
- + verifyEmail(token: String): Boolean
- + 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:

- + submitMaintenanceRequest(category: String, description: String, locationID: String, priority: String, images: List): String
- + viewDashboard(): void
- + viewTicketStatus(ticketID: String): String
- + addTicketComment(ticketID: String, comment: String): Boolean
- + markTicketAsUrgent(ticketID: String): Boolean
- + configureNotificationPreferences(): void

Systems Analysis and Design Project

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:

- + viewAssignedTasks(): List
- + acceptTask(ticketID: String): Boolean
- + declineTask(ticketID: String, reason: String): Boolean
- + updateTaskStatus(ticketID: String, status: String): Boolean
- + uploadMaintenanceEvidence(ticketID: String, images: List, notes: String): Boolean
- + requestAdditionalInfo(ticketID: String): Boolean
- + 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:

- + viewSystemDashboard(): void
- + assignTicket(ticketID: String, technicianID: String): Boolean
- + reassignTicket(ticketID: String, technicianID: String): Boolean
- + manageUserAccount(userID: String, action: String): Boolean
- + updateTicketPriority(ticketID: String, priority: String): Boolean
- + generateReports(type: String, dateRange: String): File
- + configureSystemRules(): void

Systems Analysis and Design Project

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:

- + create(): Boolean
- + updateStatus(status: String): Boolean
- + addComment(comment: String): Boolean
- + 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)

Systems Analysis and Design Project

- notificationType: String (private)
- isRead: Boolean (private)
- createdAt: DateTime (private)
- expirationDate: DateTime (private)

Methods:

- + 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:

- + 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]

Systems Analysis and Design Project

- 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]

Systems Analysis and Design Project

- 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.

Systems Analysis and Design Project

- 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 component.
- 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.

Systems Analysis and Design Project

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

Systems Analysis and Design Project

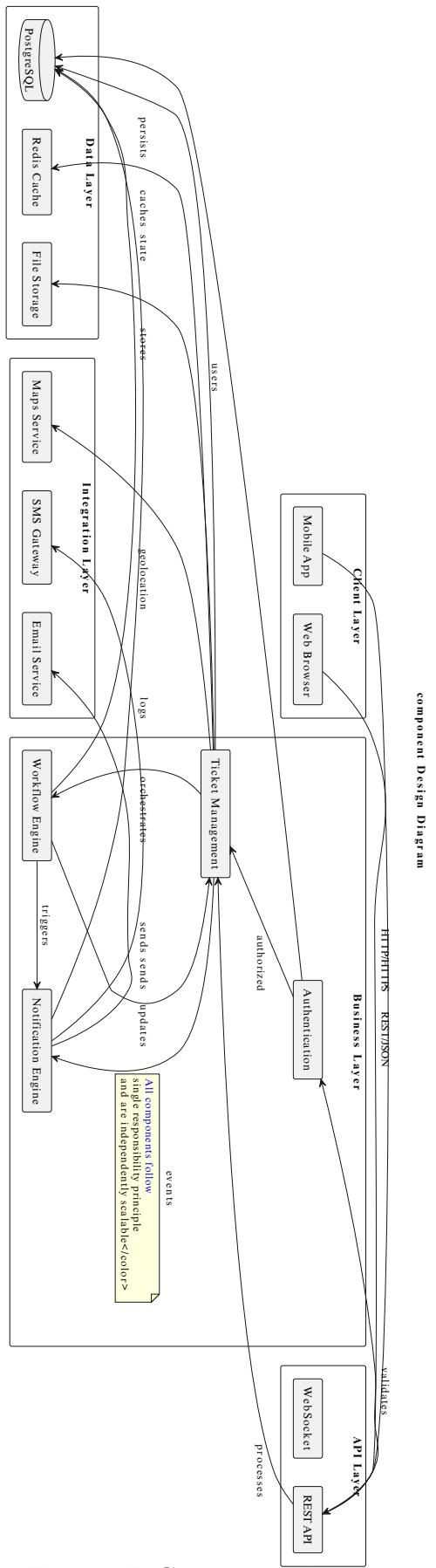


Figure 87: Component Diagram

Systems Analysis and Design Project

4.7. ERD analysis and Database Design

4.7.1. Database Design (ERD)

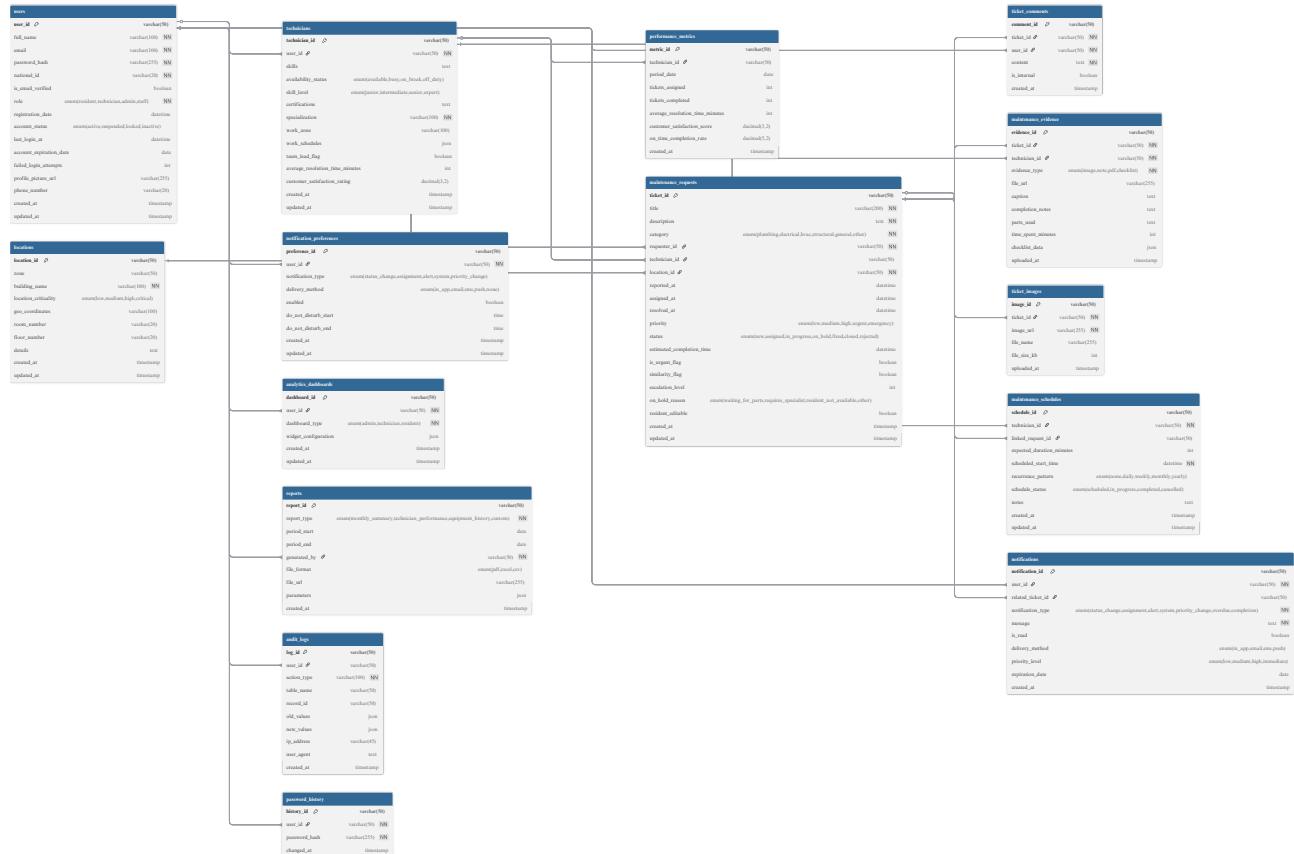


Figure 88: Entity Relation Diagram

4.7.2. Object to ER Mapping

Table 115: 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

Systems Analysis and Design Project

Table 116: 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

Systems Analysis and Design Project

Table 117: 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

Systems Analysis and Design Project

Table 118: 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

Systems Analysis and Design Project

Table 119: 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

Systems Analysis and Design Project

Table 120: 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

5. User Manual

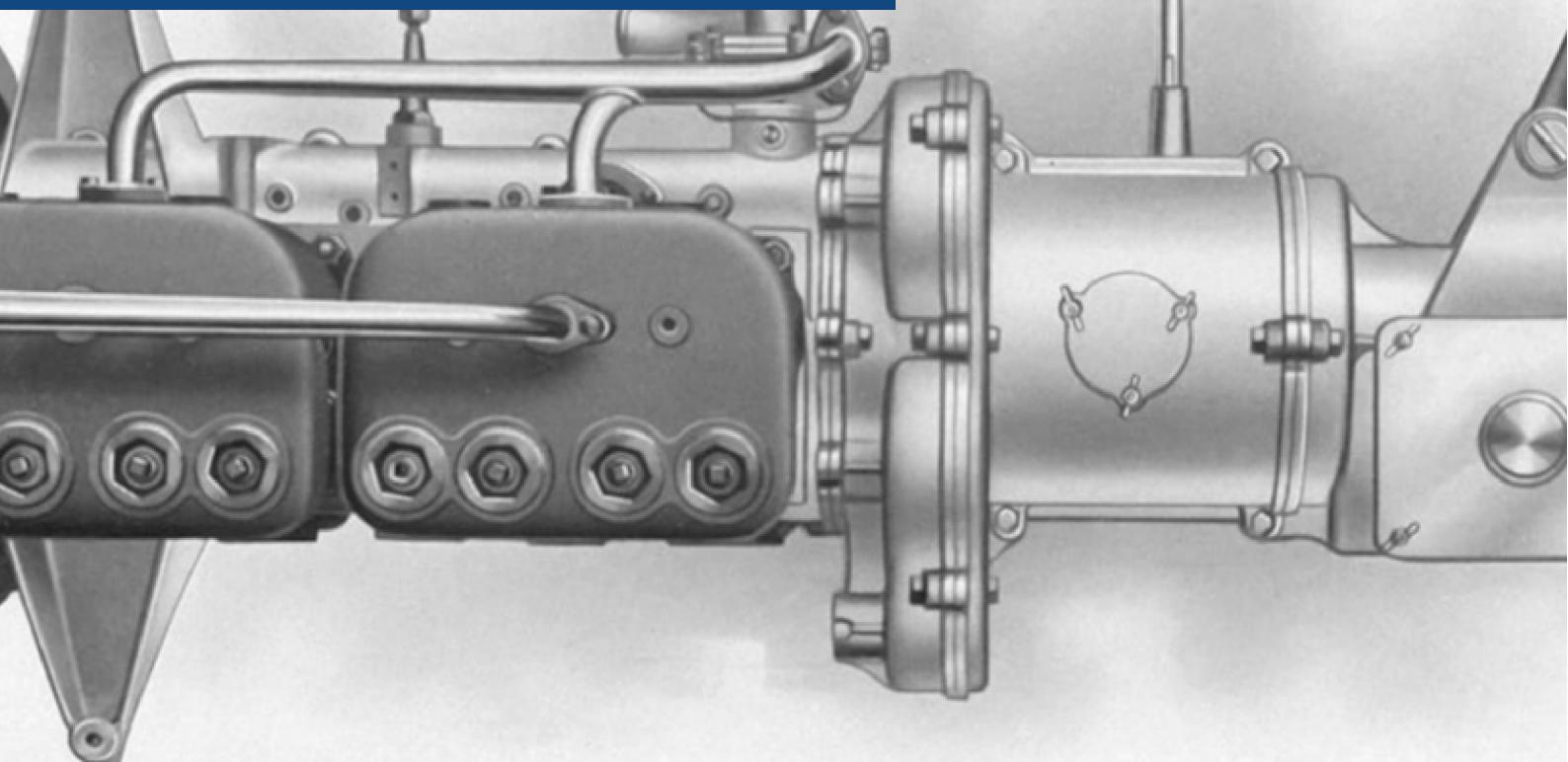
2025-
2026

SALLEHA

Supervisor: Prof
Hamad Alsawalqah

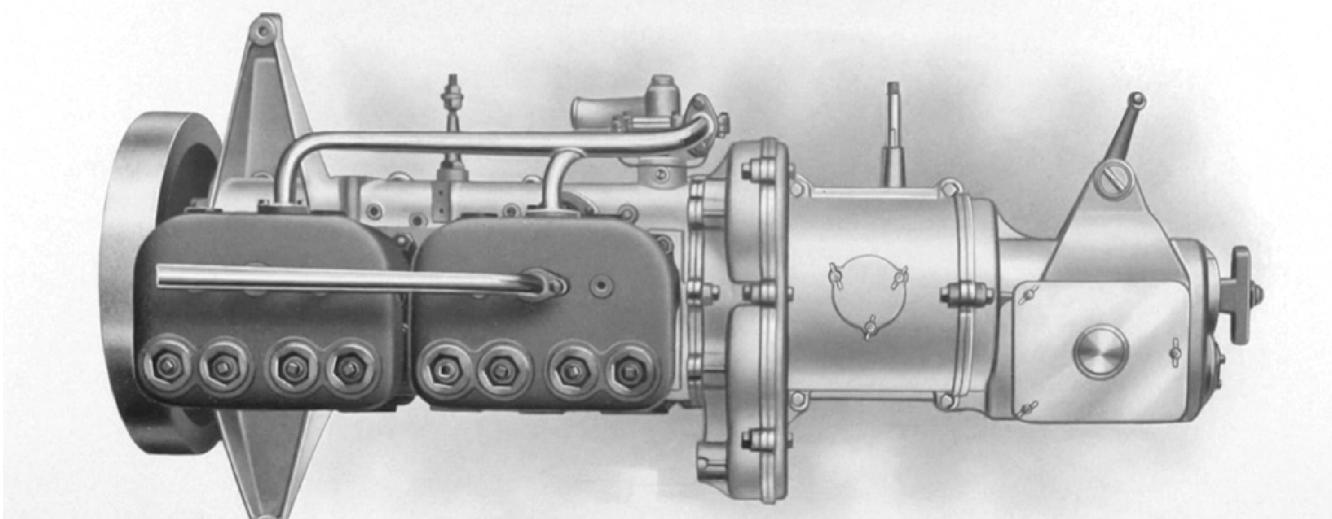


USER MANUAL



INDICE

1. Introduction
2. Getting Started
3. System Interface
4. Using the System
5. Frequently Asked Questions
6. Troubleshooting Tips
7. Legal and Safety Information
8. Support and Contact Information
9. Conclusion



1. Introduction

Overview

Salleh System is a maintenance management system designed to help residents submit maintenance requests easily, while enabling administrators and technicians to manage, track, and resolve maintenance issues efficiently. Purpose of the System The purpose of the system is to streamline the maintenance request process, reduce response time, avoid duplicate requests, and improve communication between residents, technicians, and administrators.

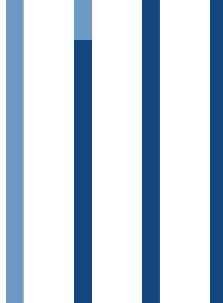
Key features and benefits

- Easy submission of maintenance requests
- Real-time tracking of request status
- Technician task assignment and updates
- Notification system for status changes
- Centralized dashboard for all users

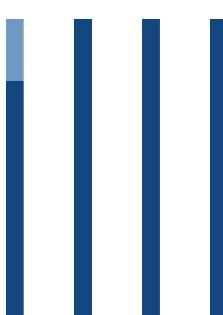
2. Getting Started

System Requirements - Internet connection - Web browser (Chrome, Edge, or Firefox) - Mobile or desktop device Installation Instructions No installation is required. The system is accessed through a web browser using the provided system URL.

First-time Setup

- 
- Open the system URL - Create a new account - Verify your email (if required) - Log in using your credentials

Account Creation and Login Process

- 
- Click Register and enter required personal information - Create a username and password - Log in using your username, password, and CAPTCHA verification
-

Account Creation and Login Process

SALLEHA
RESIDENT ACCESS

Create your account
Register once to manage all your maintenance requests, track updates, and adjust your notification preferences.

SALLEHA is designed to be clear, calm, and accessible so everyone in your community can get the help they need.

Register as a resident

Fill in a few details so we can connect your account to the right building and unit.

Already have an account? [Sign in instead](#)

First name **Last name**

Enter first name Enter last name

Email address

name@example.com

Mobile number (optional) **Preferred language**

Add a mobile number

Building

Unit / Apartment **Access or registration code (if provided)**

e.g. 402 Enter code

Create password **Confirm password**

Create a secure password

Re-enter password

Accessibility preferences

Use larger text and higher contrast

I agree to receive important updates about maintenance requests and building notices at the contact details provided.

Create resident account

Cancel and go back

By creating an account, you confirm that your details are correct so your requests can be routed to the right team.
If you need help registering, please contact your building office.

SALLEHA
ACCESS PORTAL

Welcome back
Sign in to view your request status, submit new maintenance tickets, and manage your building profile.

Need assistance? Contact your building administrator or support desk.

Sign in to your account

Please enter your registered email and password.

New to SALLEHA? [Create an account](#)

Email address

name@example.com

Password [Forgot password?](#)

Enter your password

Sign in

OR

Sign in with access code

3. System Interface

Navigation Overview

The system uses a simple menu-based navigation with access to Dashboard, Maintenance Requests, Notifications, and Profile settings.

Dashboard Description

The dashboard displays:

- Current maintenance requests
- Request statuses
- Notifications and updates
- Quick access to main features

Resident dashboard:

The screenshot shows the SALLEHA Resident Portal Dashboard. On the left is a dark sidebar with the SALLEHA logo, 'RESIDENT PORTAL', and links for 'Dashboard' (highlighted), 'My Tickets', 'Notifications', 'Profile', 'Settings', and 'Logout'. The main area has a light blue header with the title 'Dashboard', a search bar, and a user profile for 'Sarah Jennings, Unit 402'. Below the header is a section titled 'Welcome back, Sarah' with a message: 'Here's what's happening with your maintenance requests today.' It shows three summary boxes: 'Open Requests' (2), 'In Progress' (1), and 'Completed (30d)' (5). Underneath are sections for 'Recent Requests' (listing 'Leaking Faucet in Kitchen', 'AC Not Cooling', and 'Light Bulb Replacement') and 'Quick Actions' (buttons for 'New Request' and 'Help & Support').

Technician dashboard:

The screenshot shows the Technician dashboard for Salleha. On the left is a sidebar with a dark blue background. It includes links for Assigned Tickets, History, Profile, and Logout. Below these is a user profile section for "Ahmed Al-Sayed" (Technician) with a small profile picture. The main area is titled "Assigned Tickets" and contains a table of six maintenance requests. The columns are: TICKET ID, LOCATION, DESCRIPTION, PRIORITY, DATE CREATED, STATUS, and ACTION. Each row shows a ticket number, its location, a brief description, its priority (e.g., High, Medium, Low), the date it was created, its current status (Received or In Progress), and a "View Details" button.

TICKET ID	LOCATION	DESCRIPTION	PRIORITY	DATE CREATED	STATUS	ACTION
#TK-8024	Bldg A, Floor 3	AC unit making loud rattling noise in hallway	High	Oct 30, 09:45 AM	Received	<button>View Details</button>
#TK-7992	Main Lobby	Automatic door sensor malfunction, stuck o...	High	Oct 29, 04:15 PM	In Progress	<button>View Details</button>
#TK-7985	Bldg B, Room 204	Leaking faucet in kitchenette area	Medium	Oct 29, 01:30 PM	Received	<button>View Details</button>
#TK-7950	Parking Zone C	Light fixture flickering near elevator	Low	Oct 28, 11:00 AM	In Progress	<button>View Details</button>
#TK-7948	Conf Room 1	Projector screen stuck halfway down	Medium	Oct 28, 09:15 AM	Received	<button>View Details</button>
#TK-7921	Bldg A, 1st Floor	Keycard reader intermittent failure	High	Oct 27, 03:20 PM	In Progress	<button>View Details</button>

Admin dashboard:

The screenshot shows the Admin dashboard for Salleha. The left sidebar has a dark blue background with links for Dashboard, All Tickets, Assignment Mgmt, Priority Mgmt, Notifications, Analytics & Reports, User Accounts, and Admin Profile. The main area is titled "Dashboard" and features several key performance indicators (KPIs) and visualizations. At the top right is a search bar and a date range selector set to "Oct 1 - Oct 31, 2023". Below the KPIs is a bar chart titled "Ticket Volume Overview" comparing weekly and monthly ticket volumes. To the right is a section titled "Technician Workload" showing the progress of four technicians: Ahmed K. (85%), Sara M. (45%), John D. (62%), and Fatima R. (20%). At the bottom are four "Quick Actions" buttons: Assign Ticket, Priorities, Reports, and Technicians.

TOTAL TICKETS
1,248
▲ 12% vs last month

OPEN
42
— Same as yesterday

IN PROGRESS
18
↗ 5 new today

COMPLETED
1,188
🕒 98% success rate

HIGH PRIORITY
7
🕒 Requires attention

AVG RESPONSE
2h 15m
🕒 -15m improvement

Ticket Volume Overview

Weekly Monthly

Technician Workload

Technician	Progress (%)
Ahmed K.	85%
Sara M.	45%
John D.	62%
Fatima R.	20%

Assign Ticket
Manage unassigned tasks

Priorities
Update urgency levels

Reports
Generate monthly report

Technicians
View availability

Key Sections of the Interface

- Dashboard
- Maintenance Requests

- Notifications
- User Profile

Ticket screen:

The screenshot shows the 'My Requests' section of the SALLEHA platform. On the left is a dark sidebar with navigation links: Dashboard, Tickets (selected), Notifications, and Settings. The main area has a light blue header with the title 'My Requests'. Below the header are search and filter controls: a search bar ('Search tickets by ID, title...'), dropdowns for 'Status: All' and 'Sort: Newest', and a '+ New Request' button. The main content area displays four ticket cards in a grid:

Icon	Title	Location	Status	Created/Updated
	Leaking Kitchen Faucet REQ-4022	Unit 402 • Kitchen	In Progress	Today, 9:41 AM
	AC Unit Not Cooling REQ-3901	Unit 402 • Living Room	Pending	Yesterday
	Flickering Hallway Light REQ-3855	Building A • 4th Floor Hall	Completed	Oct 24, 2023
	Keycard Access Issue REQ-3810	Main Entrance	Completed	Oct 20, 2023

Notification screen:

The screenshot shows the 'Notifications' section of the SALLEHA platform. On the left is a dark sidebar with navigation links: Dashboard, Tickets, Notifications (selected), and Settings. The main area has a light blue header with the title 'Notifications'. Below the header are tabs for 'All', 'Tickets', and 'System', and a 'Mark all as read' button. The main content area displays notifications grouped by time: TODAY, YESTERDAY, and EARLIER.

- TODAY**
 - Technician Assigned**
Your request #REQ-4022 (Leaking Kitchen Faucet) has been assigned to technician Mike Johnson. 10:42 AM
 - New Comment**
Admin added a note to #REQ-4022: "Please ensure the area is clear for access." 09:15 AM
- YESTERDAY**
 - Request Completed**
Request #REQ-3855 (Flickering Hallway Light) has been marked as completed. 4:30 PM
 - Scheduled Maintenance**
The main elevator will be under maintenance tomorrow from 10 AM to 2 PM. 11:00 AM
- EARLIER**
 - Request Approved**
Your request #REQ-3810 (Keycard Access Issue) has been approved and is now pending assignment. Oct 20

Icons and Buttons

- Add: Create a new request
- Notification icon: View alerts
- Edit: Modify information
- Status icons: Show request progress

4. Using the System

Step-by-step Instructions for Common Tasks

Task 1: Submitting a Maintenance Request

- Log in to the system
- Click Report New Issue
- Select category, priority, and location
- Enter a description
- Upload images (optional)
- Submit the request

Tickets request screen:

SALLEHA

New Request

Dashboard Tickets Notifications Settings

Issue Details

Ticket Title
Brief summary of the issue (e.g., Leaking faucet)

Category
Select category...

Urgency
Normal

Description
Please describe the problem in detail. Include any specific noises, error codes, or circumstances.

Location

Building
Building A - North Tower

Unit / Apartment
Unit 402

Specific Room / Area
e.g., Kitchen, Master Bath

Photos & Attachments

Click to upload or drag and drop
SVG, PNG, JPG or GIF (max. 5MB)

Privacy

Submit Anonymously
Your name and contact details will be hidden from the technician. You will still receive status updates via the app.

Priority management:

The screenshot shows the Salleha Priority Management interface. On the left sidebar, under the 'Priority Mgmt' section, the 'Priority Levels & Definitions' table is displayed. It lists four priority levels: Critical (red), High (orange), Normal (blue), and Low (grey). Each row includes a description, SLA target, and edit/delete actions. Below this, the 'Active Tickets by Priority' section shows five recent tickets. The first ticket, #TK-2942, is a critical AC unit malfunction assigned to Ahmed Khalil and is in progress. The second ticket, #TK-2935, is a high-priority pool pump noise report unassigned. The third ticket, #TK-2941, is a normal-leaking faucet report assigned to Fatima R. and completed. The fourth ticket, #TK-2931, is a normal-lobby light flickering report assigned to John Doe and is in progress.

Order	Priority Name	Description & SLA	SLA Target	Actions
1	Critical	Issues posing immediate safety risks or total system failure. Requires immediate intervention.	2 Hours	edit delete
2	High	Major functionality impaired but no immediate danger. Affects multiple users or key amenities.	24 Hours	edit delete
3	Normal	Standard maintenance requests, cosmetic issues, or single-user inconveniences.	48 Hours	edit delete
4	Low	Scheduled maintenance, suggestions, or non-urgent inquiries.	5 Days	edit delete

Active Tickets by Priority				
Ticket Details	Subject	Current Priority	Assigned To	Status
#TK-2942 Created 2 hrs ago	AC Unit Malfunction - Block B	Critical	Ahmed Khalil	IN PROGRESS
#TK-2935 Created 1 day ago	Pool pump noise	High	Unassigned	OPEN
#TK-2941 Created 3 days ago	Leaking faucet - Apt 402	Normal	Fatima R.	COMPLETED
#TK-2931 Created 3 days ago	Lobby light flickering	Normal	John Doe	IN PROGRESS

Task 2: Tracking Request Status

- Open Dashboard
- View submitted requests
- Check current status and updates

Tickets screen:

The screenshot shows the Salleha Tickets screen. The left sidebar has a 'Tickets' section selected. The main area is titled 'My Requests' and displays four recent tickets. Each ticket card includes the title, location, status, and creation date.

REQUEST ID	TITLE	LOCATION	STATUS	CREATED
#REQ-4022	Leaking Kitchen Faucet	Unit 402 • Kitchen	In Progress	Today, 9:41 AM
#REQ-3901	AC Unit Not Cooling	Unit 402 • Living Room	Pending	Yesterday
#REQ-3855	Flickering Hallway Light	Building A • 4th Floor Hall	Completed	Oct 24, 2023
#REQ-3810	Keypad Access Issue	Main Entrance	Completed	Oct 20, 2023

Task 3: Managing Requests (Admin/Technician)

- View assigned or incoming requests
- Update request status
- Add comments or evidence
- Close request after resolution

Technician side:

Ticket details

The screenshot shows the Salleha ticket management interface. On the left is a dark sidebar with a user icon, the name 'Salleha', and navigation links for 'Assigned Tickets', 'History', and 'Profile'. At the bottom of the sidebar are 'Logout' and the technician's profile information: 'Ahmed Al-Sayed' and 'Technician'. The main area has a light gray background. At the top center is a header 'Ticket Details' with a back arrow and a bell icon. Below it is a ticket card for '#TK-8024'. The ticket card includes the following details:

- Received**
- AC unit making loud rattling noise in hallway**
- LOCATION**: Bldg A, Floor 3
- PRIORITY**: High Priority
- CREATED**: Oct 30, 09:45 AM
- FULL DESCRIPTION**: Staff reported a loud, persistent rattling sound coming from the ceiling-mounted AC unit in the main corridor of Building A, 3rd Floor. The noise seems mechanical and increases when the fan speed is high. It is disrupting nearby offices. Please inspect the fan motor and mounting brackets for loose components.
- Attachments**: A thumbnail for 'AC Unit'.

To the right of the ticket card is a 'Update Status' panel with fields for 'Received', 'Add progress notes...', and a file upload section for 'Attach photo or document'. A large blue button at the bottom right says 'Add Update'. Below this is an 'Activity History' panel showing two entries:

- System** (9:45 AM): Ticket created and assigned to Ahmed Al-Sayed.
- Sarah Jenkins** (9:40 AM): "The noise is very distracting, please fix ASAP." Attachment added: ac_noise.mp4

Maintenance History

Maintenance History

ID	Subject	Location	Date	Priority	Action
#TK-7045	Leak in pantry sink faucet	Bldg B, Floor 1	Oct 28, 2023	Normal	>
#TK-6892	Fluorescent light replacement in hallway	Bldg A, Floor 2	Oct 25, 2023	Low	>
#TK-6501	Server room cooling malfunction	Server Room, Main	Oct 20, 2023	High	>
#TK-6100	Broken door handle repair	Lobby Entrance	Oct 15, 2023	Normal	>
#TK-5900	Elevator 2 button panel stuck	Elevator 2	Oct 10, 2023	High	>

Admin side:

Admin assignment management

Ticket Assignment Management

Show: Unassigned	Location: All	Urgency: High+			
#TK-2942	AC Unit Malfunction	Block B, Floor 3	Critical	UNASSIGNED	2 hrs ago
#TK-2937	Gym treadmill repair	Rec Center	Normal	UNASSIGNED	1 day ago
#TK-2935	Pool pump noise	Main Pool	High	UNASSIGNED	1 day ago
#TK-2941	Leaking faucet	Apt 402	Normal	ASSIGNED	3 days ago
#TK-2931	Lobby light flickering	Tower A Lobby	Normal	UNASSIGNED	3 days ago

Assign Technician Selected: #TK-2942

Find technician...
Ahmed Khalil HVAC Specialist Available 2 active
John Doe General Repair Busy 5 active
Sara M. Electrical Available 1 active
Fatima R. Plumbing Available 0 active

Assigning Ahmed Khalil to ticket #TK-2942.

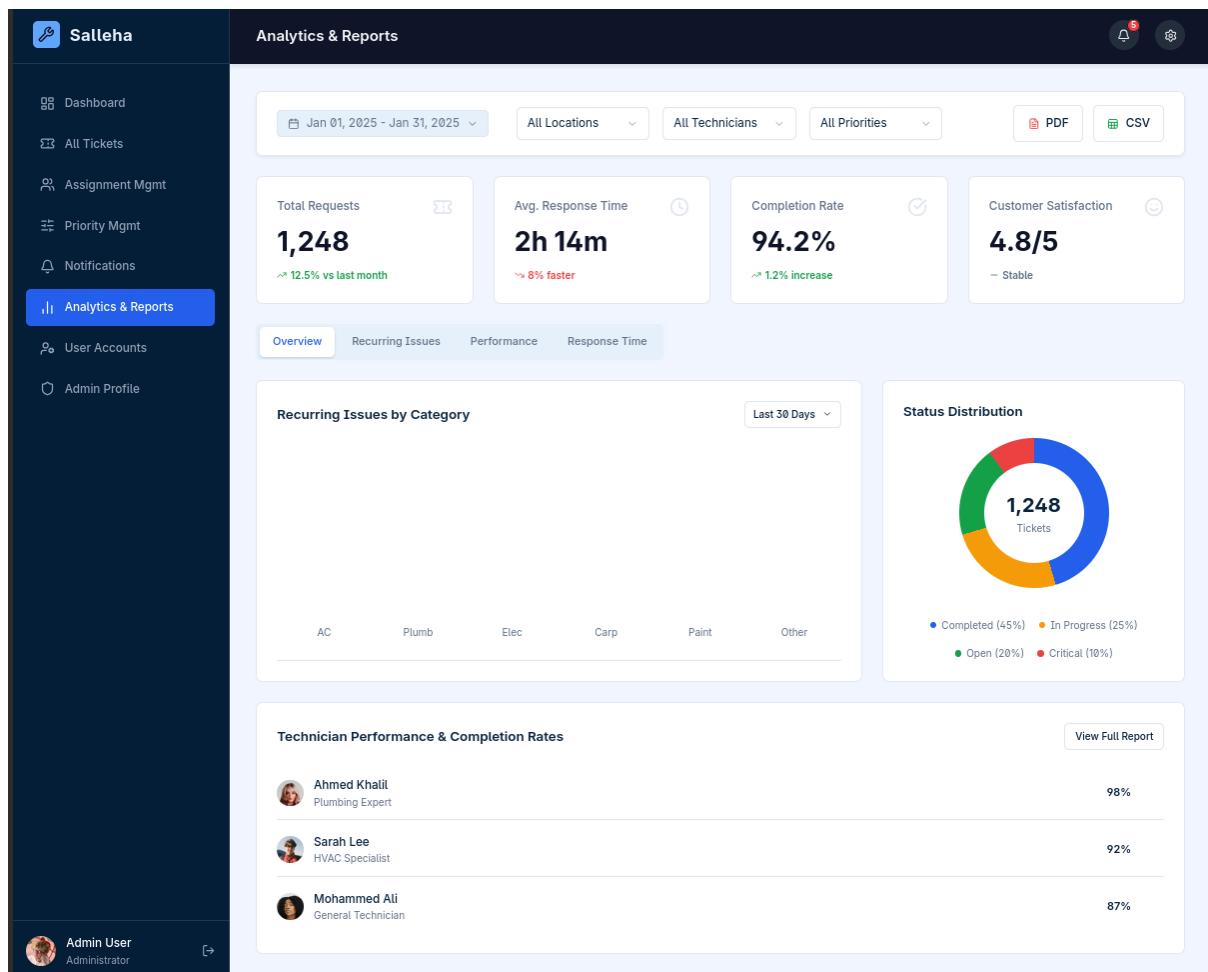
Confirm Assignment

Admin all tickets

Advanced Features

- Duplicate request detection
- Priority escalation

- Notification preferences
- Report generation (Admin only)



5. Frequently Asked Questions (FAQs)

1. How do I submit a maintenance request?

Log in to the system, go to the Dashboard, click on Report New Issue, fill in the required details, and submit the request.

2. What information is required to submit a maintenance request?

You need to select the issue category, location, priority level, and provide a clear description. Uploading images is optional but recommended.

3. Can I upload images with my maintenance request?

Yes. You can upload up to 5 images to help technicians better understand the issue.

4. How can I track the status of my request?

You can track your request from the Dashboard under My Maintenance Requests, where the current status is displayed.

5. What do the different request statuses mean?

Submitted: Request has been sent successfully

Assigned: Technician has been assigned

In Progress: Work is currently being done

Awaiting Response: More information is required

Resolved: Issue has been fixed

Closed: Request is completed and archived

6. Can I edit or cancel a request after submitting it?

You can edit or cancel the request only before it is assigned to a technician.

7. What happens if I submit a duplicate request?

The system checks for similar requests submitted within the last 7 days and will notify you if a possible duplicate is detected.

8. How will I be notified about updates?

You will receive notifications through the system dashboard and, if enabled, via email or SMS.

9. What should I do if my request is urgent?

You can mark the request as urgent during submission or update its priority later if allowed.

10. Why was my request marked as “Awaiting User Response”?

This means the technician or administrator needs additional information to continue working on the issue.

11. Can technicians communicate with residents through the system?

Yes. Technicians can add comments and request additional information through the request details page.

12. What should I do if I forget my password?

Click on Forgot Password, enter your registered email or national ID, and follow the password reset instructions.

13. Why is my account locked?

Your account may be locked due to multiple failed login attempts. Please wait or contact support.

14. Can administrators reassign a maintenance request?

Yes. Administrators can reassign requests to another technician if needed.

15. How long does it take to resolve a maintenance request?

Resolution time depends on the issue type, priority level, and technician availability.

16. Can I view my past maintenance requests?

Yes. All previous requests are available in your request history.

17. Is my personal information secure?

Yes. The system uses secure authentication and access control to protect user data.

18. What file formats are supported for image uploads?

Common image formats such as JPG and PNG are supported.

19. What should I do if the system is not loading properly?

Try refreshing the page, checking your internet connection, or logging out and back in.

20. Who can I contact for technical support?

You can contact system support via the support email or phone number listed in the Support section.

6. Troubleshooting Tips

Issues and solutions

- Login failed: Ensure correct username/password
- Images not uploading: Check file size and format

Error Codes and Their Meanings

- ERR-401: Unauthorized access
- ERR-403: Account locked or inactive
- ERR-500: System error

Contacting Support

Contact system support through the provided support email or phone number.

7. Administrative Management Screens

This section presents administrative interfaces used to manage user accounts, system notifications, and internal system operations.

Admin user accounts:

User Account Management

Technicians Residents / Users Administrators Roles & Permissions

Search by name, email or ID... Filter Export + Add New User

	Technician	Contact Info	Skills	Current Load	Status	Actions
<input type="checkbox"/>	Ahmed Khalil ID: TECH-001	+971 50 123 4567 ahmed.k@salleha.com	Plumbing General	2 Active	Available	Edit Delete
<input type="checkbox"/>	Sarah Lee ID: TECH-004	+971 52 987 6543 sarah.l@salleha.com	Electrical HVAC	5 Active	Busy	Edit Delete
<input type="checkbox"/>	John Doe ID: TECH-007	+971 55 444 3333 john.d@salleha.com	Carpentry	0 Active	Offline	Edit Delete
<input type="checkbox"/>	Mohammed Ali ID: TECH-009	+971 50 111 2222 moh.al@salleha.com	HVAC Plumbing	3 Active	Available	Edit Delete

Showing 1-4 of 24 technicians

Admin notifications:

Notifications Real-time updates active

All Notifications Unread Only Archived Mark all as read

New Tickets 2 New

- New maintenance request #TK-2943 Resident in Unit 305 reported "Water leakage in master bathroom ceiling". Photos attached. Category: Plumbing [View Ticket](#) 10 mins ago
- New maintenance request #TK-2942 Concierge reported "Lobby automated door stuck halfway". Category: Electrical [View Ticket](#) 45 mins ago

Priority Changes 1 Update

- Priority escalated for #TK-2930 System auto-escalated ticket due to SLA breach risk. Priority changed from Normal to High. [Review Priority](#) 2 hours ago

Technician Updates 1 New

- Technician Ahmed K. is now Available Ahmed Khalil has completed all assigned tasks for the morning shift and is ready for new assignments. [Assign Tasks](#) 1 hour ago
- Technician Sarah L. checked in late Clock-in time recorded at 09:15 AM (Scheduled: 09:00 AM). 4 hours ago

Completed Tasks

- Ticket #TK-2900 marked as Resolved "Replace hallway lightbulb on 4th floor" was completed by John Doe. Resident confirmation pending. [View Report](#) Yesterday at 4:30 PM

8. Legal and Safety Information

Terms of Service

Users must comply with system usage policies and provide accurate information.

Privacy Policy

User data is securely stored and used only for system operations.

Safety Guidelines

Do not submit false or misleading maintenance requests.

9. Support and Contact Information

How to Contact Support

- Email: support@sallehsystem.com
- Phone: 0797777777

Hours of Availability

Sunday to Thursday, 9:00 AM - 5:00 PM

10. Conclusion

Summary of Features

Salleh System provides an efficient and user-friendly way to manage maintenance requests from submission to resolution.

Final Thoughts on System Usage

Using the system correctly ensures faster response times, better communication, and improved maintenance management.

Anas AL-Jallad	0225343
Shaima Bushnaq	0227646
Haneen Alajaleen	0226320
Mosa Daradkah	0222634
Orjoan Aldabaibah	0224933

6. References

- [1] K. E. Kendall and J. E. Kendall, **Systems Analysis and Design**, 11th ed. Hoboken, NJ, USA: Pearson, 2023.
- [2] J. W. Satzinger, R. B. Jackson, and S. D. Burd, **Systems Analysis and Design in a Changing World**, 7th ed. Boston, MA, USA: Cengage Learning, 2016.
- [3] TutorialsPoint, “System Analysis and Design Tutorial.” [Online]. Available: https://www.tutorialspoint.com/system_analysis_and_design/
- [4] dbdiagram.io, “Database Diagram Design Tool.” [Online]. Available: <https://dbdiagram.io/>
- [5] draw.io, “Online Diagramming Tool.” [Online]. Available: <https://www.diagrams.net/>
- [6] Figma, “Collaborative Interface Design Tool.” [Online]. Available: <https://www.figma.com/>
- [7] PlantUML, “Text-Based UML Diagram Generator.” [Online]. Available: <https://plantuml.com/>