Software Requirements Specification

for

AIRPORT MANAGEMENT SYSTEM

Version 1.0

Prepared by

Group Name: DBMS GROUP 18

JESVIN JAISON	B230359CS	jesvin_b230359cs@nitc.ac.in
P GOWRI SANKAR	B230477CS	gowri_b230477@nitc.ac.in
ASWIN S	B230215CS	aswin_b230215cs@nitc.ac.in
AAKANSH K ANIL	B230117CS	aakansh_b230117cs@nitc.ac.in

Instructor: **Professor Prabhu Mohandas**

Course: **CS2011E DBMS**

Date: **1/03/2025**

TABLE OF CONTENTS

CONTENTS

1 INTRODUCTION

1.1 Document Purpose
1.2 Product Scope
1.3 Intended Audience and Document Overview
1.4 Definitions, Acronyms and Abbreviations
1.5 Document Conventions
1.6 References and Acknowledgments

2 OVERALL DESCRIPTION

2.1 Product Overview
2.2 Product Functionality
2.3 Design and Implementation Constraints
2.4 Assumptions and Dependencies

3 SPECIFIC REQUIREMENTS

3.1 External Interface Requirements
3.2 Functional Requirements
3.3 Use Case Model

4 OTHER NON-FUNCTIONAL REQUIREMENTS

4.1 Performance Requirements
4.2 Safety and Security Requirements

APPENDIX A – DATA DICTIONARY

APPENDIX B - GROUP LOG

1 : Introduction

1.1 Document Purpose

The **Software Requirements Specification (SRS)** document defines the functional and non-functional requirements for the **Airport Management System**. This document serves as a detailed reference for developers, testers, and stakeholders, outlining the system's capabilities, constraints, and expected behavior.

This SRS ensures a **clear understanding of system functionalities** for managing airport operations, including flight tracking, passenger management, baggage handling, employee scheduling, and security features such as the No-Fly List.

The document will guide:

- **Developers**, to implement the system as per specifications.
- **Testers**, to validate the system against defined requirements.
- **Stakeholders**, to review and approve the system's features before development.

This document covers **all components** of the Airport Management System and provides a structured approach for its design, implementation, and maintenance.

1.2 Product Scope

The **Airport Management System (AMS)** is a **web-based application** designed to streamline airport operations by **automating flight tracking, passenger management, baggage handling, employee scheduling, and security enforcement**. The system ensures seamless coordination between different airport departments, enhancing **efficiency, security, and passenger experience**.

The **AMS operates as a centralized platform**, allowing airport staff to manage operations efficiently through an intuitive web interface. Passengers' data, flights, and baggage are **tracked in real-time**, ensuring smooth workflows and reducing manual errors. Security personnel can verify passengers against the **No-Fly List**, while airport management can oversee employee schedules and resource allocation.

- **Technology Stack:**
- **Frontend:** Next.js (React.js)
- **Backend:** Node.js with Express.js
- **Database:** MySQL
- **Deployment:** Azure
- **Security:** Role-based access control with authentication mechanisms
- **Purpose & Benefits:**

The primary objective of the AMS is to **optimize airport operations** by eliminating inefficiencies in manual processes. Unlike traditional management systems, **AMS does not**

operate flights or handle ticket bookings; rather, it serves as an **administrative and security hub**, ensuring smooth daily operations.

Key benefits include:

- **Automated tracking** of flights, baggage, and employees, reducing human error.
- **Enhanced security** through real-time **No-Fly List verification** and access control.
- **Improved passenger experience** with quick check-ins and baggage claim management.
- **Operational efficiency**, ensuring better coordination between airport departments.

The **Airport Management System** is essential for modern airports looking to transition from manual record-keeping to a **data-driven**, highly efficient, and **secure** operational model.

1.3 Intended Audience and Document Overview

This **Software Requirements Specification (SRS)** is intended for:

- **Developers:** Implement system functionalities.
- **Project Managers:** Oversee development and ensure adherence to requirements.
- **Testers:** Validate system performance and functionality.
- **Airport Staff & Security Personnel:** Operate the system for airport management and security enforcement.
- **Professors & Evaluators:** Assess the project's compliance with software engineering principles.

· **Document Overview**

This document is structured as follows:

1. **Introduction:** Defines the system's purpose, scope, and audience.
2. **Overall Description:** Provides an overview of system functionalities, constraints, and dependencies.
3. **Specific Requirements:** Details functional and non-functional requirements.
4. **Use Case Model:** Illustrates user interactions with the system.
5. **Non-Functional Requirements:** Covers security, performance, and quality attributes.
6. **Appendices:** Includes additional references like a **Data Dictionary** and **Group Log**.

For best readability:

- **Managers & Evaluators** should focus on **Sections 1 & 2**.
- **Developers & Testers** should prioritize **Sections 3 & 4**.
- **Airport Staff & Security** should refer to **Sections 2 & 3**.

This format ensures each stakeholder finds relevant information efficiently.

1.4 Definitions, Acronyms and Abbreviations

API: Application Programming Interface

CRUD: Create, Read, Update and Destroy

DBMS: Database Management System

UI/UX: User Interface/User Experience

ETA: Estimated Time of Arrival

SQL: Structured Query Language

1.5 Document Conventions

This document follows the IEEE formatting requirements. The text is written in Arial with a font size of 11 or 12, with spacing and 1-inch margins. Italics are used for comments and notes.

1.6 References and Acknowledgments

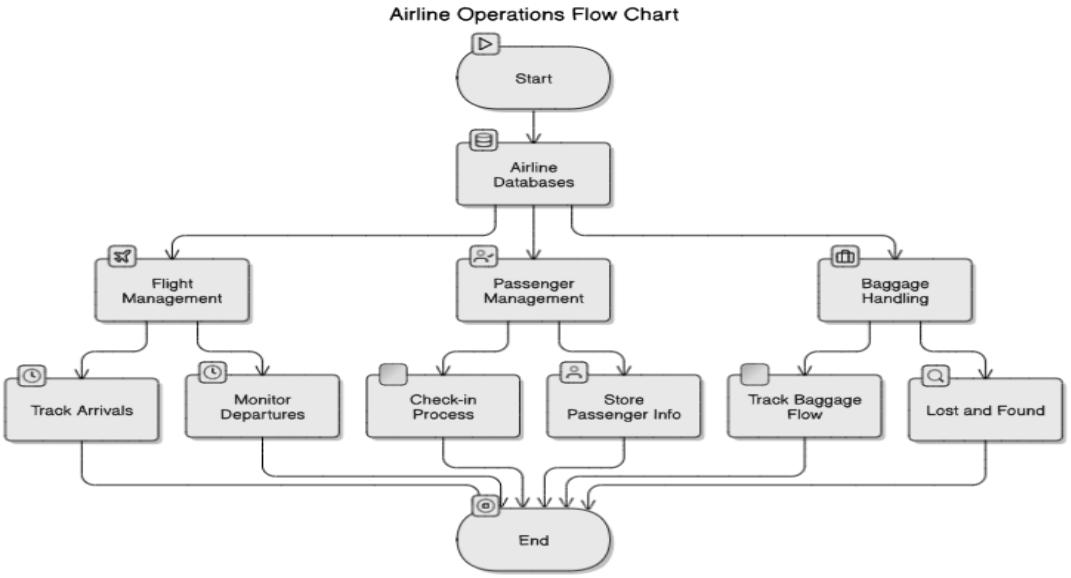
1. IEEE Standard for Software Requirement Specifications
2. React.js and Node.js Official Documentation
3. MySQL Documentation

2 : Overall Description

2.1 Product Overview

The **Airport Management System (AMS)** is a **self-contained, web-based application** designed to streamline airport operations by automating flight tracking, passenger management, baggage handling, employee scheduling, and security enforcement. It replaces traditional manual processes and fragmented record-keeping systems with an integrated, **real-time** solution, ensuring seamless coordination across all airport departments.

AMS is not a follow-on member of an existing product family but rather a **new, standalone system** developed to meet the operational and security challenges of a modern airport. The system **interfaces with external services**, such as airline databases, security agencies for No-Fly List verification, and internal airport subsystems for resource management. Additionally, it provides a **centralized dashboard** for administrators and airport staff to oversee daily operations efficiently.



2.2 Product Functionality

The **Airport Management System (AMS)** automates key airport operations to enhance **efficiency, security, and passenger experience**. The core functionalities include:

- **Flight Management:** Tracks real-time flight schedules, updates delays/cancellations, and manages terminal assignments.
- **Passenger Management:** Handles check-ins, verifies travel documents, and maintains passenger records.
- **Baggage Management:** Monitors baggage flow, prevents loss, and manages lost-and-found services.
- **Security & No-Fly List Verification:** Cross-checks passengers against the No-Fly List, ensuring restricted individuals are flagged.
- **Employee Scheduling:** Manages airport staff roles, duty shifts, and real-time attendance.
- **Admin & Reporting Panel:** Provides centralized control over airport operations, generating reports on flights, passengers, and security.

This streamlined functionality ensures **automation, security, and operational efficiency** across all airport departments.

2.3 Design and Implementation Constraints

The **Airport Management System (AMS)** is subject to the following key constraints:

- **Technology:** Built as a **web-based** application using **Next.js (React.js)**, **Node.js (Express.js)**, and **MySQL**, deployed on **Azure Cloud**.
- **Security:** Implements **role-based access control (RBAC)**, **AES-256 encryption**, and real-time **No-Fly List verification**.
- **Performance:** Must process **flight updates within 2 seconds** and complete **passenger check-ins within 5 seconds**.
- **Integration:** Connects with **airline databases** for schedules and **security agencies** for No-Fly List enforcement.

These constraints ensure the system remains **efficient, secure, and scalable** for modern airport operations.

2.4 Assumptions and Dependencies

- **Network Connectivity:** The system requires a **stable internet connection** for real-time updates.
- **Database & Hosting:** Relies on **MySQL** for data storage and **Azure Cloud** for deployment.
- **Third-Party Integrations:** Depends on **airline databases** for flight schedules and **security agencies** for No-Fly List verification.
- **Hardware Requirements:** Airport staff must use **modern web browsers** for system access.

These dependencies ensure seamless system functionality and integration

3 : Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The Airport Management System (AMS) web application provides the following user interfaces:

Admin Dashboard

- Manage flights, employees, and system monitoring
- Update security controls and No-Fly List

Flight Management Interface

- Track flights, assign gates, and update statuses in real-time

Passenger Check-in & Boarding Interface

- Register passengers, issue boarding passes, verify security clearances

Baggage Handling Interface

- Process check-ins, track baggage movement, manage lost-and-found

Security & No-Fly List Verification Interface

- Verify passenger identities against databases, receive alerts for flagged individuals

Employee Management Interface

- Schedule shifts, assign duties, monitor attendance and performance

Notifications & Alerts Interface

- Send flight updates, emergency alerts, and staff reminders

3.1.2 Hardware Interfaces

For the web application, the following hardware interfaces are relevant:

Client Devices

- Desktop computers, laptops, tablets, and smartphones with web browsers
- Support for touchscreen interfaces on kiosks and mobile devices

Network Infrastructure

- Secure internet connectivity for accessing the web application
- Local area network (LAN) for internal airport communications

Peripheral Devices

- Barcode scanners for boarding pass and baggage tag reading
- Printers for boarding passes and baggage tags
- Biometric scanners for employee access and passenger verification (if applicable)

3.1.3 Software Interfaces

The AMS web application integrates with:

Web Server

- Apache or Nginx to host the application

Database Management System

- MySQL for storing airport-related data

Authentication System

- For secure user login and access control

Flight Information Display System (FIDS) API

- For real-time flight updates

Weather Information API

- For real-time weather data

3.2 Functional Requirements

3.2.1 User Authentication and Access Control
F1: The system shall allow administrators, airport staff, and security personnel to create accounts and log in with role-based access control.

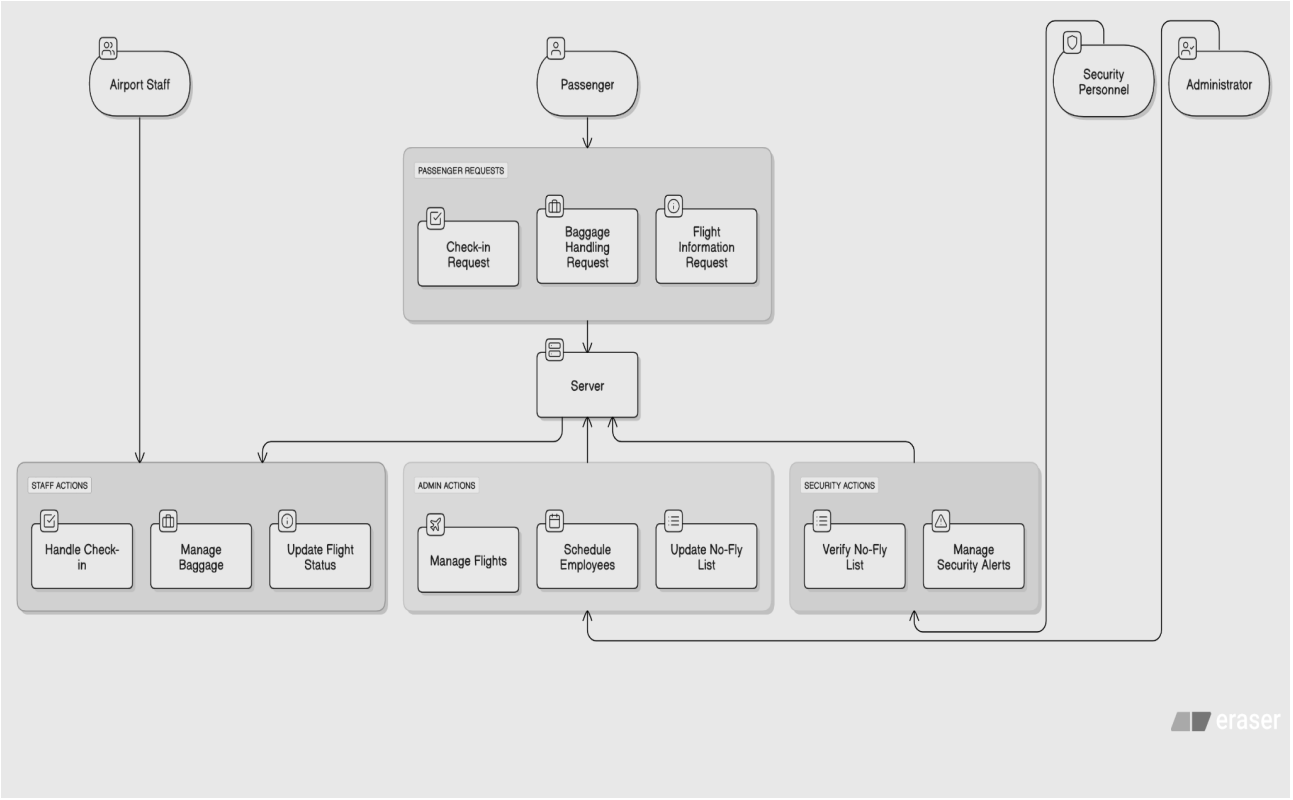
3.2.2 Flight Management
F2: The system shall enable administrators to add, update, and delete flight records.
F3: The system shall provide real-time updates for flight statuses.

3.2.3 Passenger Management
F4: The system shall allow airport staff to register new passengers and link them to specific flights.
F5: The system shall facilitate passenger check-in by verifying boarding passes and travel documents.

3.2.4 Baggage Tracking
F6: The system shall enable tracking of passenger baggage from check-in to boarding.

3.2.5 Security Measures
F7: The system shall automatically cross-check passenger details against the No-Fly List during the check-in process.
F8: The system shall allow authorized personnel to manage entries in the No-Fly List.

3.3 Use Case Model



U1: Sign Up / Log in

Author: Gowri

Purpose: Allows airport staff and administrators to create an account or log in to access the system.

Requirements Traceability: The system must securely store user credentials in the database.

Priority: High

Pre-conditions: The user must have internet access.

Post-conditions: The user is logged in and can access system functionalities based on their role.

Actors: Airport Staff, Administrator

Extends: None

Flow of Events:

Basic Flow:

1. User enters their username and password.
2. The system validates the entered credentials.
3. If valid, the user is directed to their role-specific dashboard.

    Alternative Flow:

    If the user is new, they can register by providing necessary details.

    Exceptions:

    An error message is displayed for incorrect credentials.

    Includes: None

    Notes/Issues: None

U2: Manage Flights

Author: Jeswin

Purpose: Allows administrators to add, update, or delete flight information.

Requirements Traceability: The system must maintain accurate flight records.

Priority: High

Pre-conditions: The administrator must be logged in.

Post-conditions: Flight information is updated in the system.

Actors: Administrator, Server

Extends: None

Flow of Events:

Basic Flow:

1. Administrator selects to add, update, or delete a flight.
2. System prompts for flight details (number, airline, schedule, etc.).
3. Administrator enters or modifies the information.
4. System validates and saves the changes.

    Alternative Flow:

If deleting a flight, the system asks for confirmation before removal.

Exceptions:

Invalid flight information triggers an error message.

Includes: None

Notes/Issues: Ensure real-time updates to flight information displays.

U3: Check-in Passenger

Author: Aakansh

Purpose: Allows airport staff to check in passengers for their flights.

Requirements Traceability: The system must update passenger status and link to baggage.

Priority: High

Pre-conditions: Passenger must have a valid ticket, and the staff must be logged in.

Post-conditions: Passenger is checked in, and boarding pass is issued.

Actors: Airport Staff, Passenger, Server

Extends: None

Flow of Events:

Basic Flow:

1. Staff enters passenger's ticket information.
2. System verifies flight details and passenger information.
3. Staff records any baggage information.
4. System generates a boarding pass.

Alternative Flow:

If the passenger is on the No-Fly List, alert security.

Exceptions:

Invalid ticket information prompts an error message.

Includes: U4

Notes/Issues: Ensure quick processing for efficient check-in lines.

U4: Manage Baggage

Author: Ashwin

Purpose: Tracks and manages passenger baggage throughout the airport.

Requirements Traceability: The system must maintain accurate baggage location and status.

Priority: High

Pre-conditions: Passenger must be checked in.

Post-conditions: Baggage is tracked and linked to the passenger's flight.

Actors: Airport Staff, Server

Extends: None

Flow of Events:

Basic Flow:

1. Staff scans baggage tag at check-in.
2. System links baggage to passenger and flight.
3. Baggage status is updated as it moves through the airport.

Alternative Flow:

For lost baggage, initiate a search protocol.

Exceptions:

Mismatched baggage tags trigger an alert.

Includes: None

Notes/Issues: Integrate with baggage handling systems for real-time tracking.

U5: Update Flight Status

Author: Gowri

Purpose: Allows administrators to update flight statuses in real-time.

Requirements Traceability: The system must reflect accurate and timely flight information.

Priority: High

Pre-conditions: Administrator must be logged in, and the flight must exist in the system.

Post-conditions: Flight status is updated and displayed on information boards.

Actors: Administrator, Server

Extends: None

Flow of Events:

Basic Flow:

1. Administrator selects a flight to update.
2. System displays current flight status.
3. Administrator enters new status (on time, delayed, cancelled, etc.).
4. System updates the status and notifies relevant parties.

    Alternative Flow:

    For significant delays or cancellations, trigger passenger notification protocol.

    Exceptions:

    Invalid status updates are rejected by the system.

    Includes: None

    Notes/Issues: Ensure updates are immediately reflected on all information displays.

U6: Manage No-Fly List

Author: Jeswin

Purpose: Allows security personnel to manage and update the No-Fly List.

Requirements Traceability: The system must maintain an up-to-date and secure No-Fly List.

Priority: High

Pre-conditions: Security personnel must be logged in with appropriate access rights.

Post-conditions: No-Fly List is updated and changes are logged.

Actors: Security Personnel, Server

Extends: None

Flow of Events:

Basic Flow:

1. Security personnel selects to view, add, or remove entries from the No-Fly List.
2. System displays current No-Fly List entries.
3. Security personnel makes necessary changes.
4. System validates and saves the changes.

    Alternative Flow:

    If removing an entry, the system asks for confirmation and reason for removal.

    Exceptions:

    Unauthorized access attempts are logged and reported.

    Includes: None

    Notes/Issues: Ensure strict access control and maintain an audit trail of all changes.

U7: Generate Reports

Author: Aakansh

Purpose: Allows administrators to generate various operational reports.

Requirements Traceability: The system must provide accurate and timely reporting capabilities.

Priority: Medium

Pre-conditions: Administrator must be logged in.

Post-conditions: Requested report is generated and available for viewing or download.

Actors: Administrator, Server

Extends: None

Flow of Events:

Basic Flow:

1. Administrator selects the type of report to generate (e.g., flight statistics, passenger throughput, baggage handling efficiency).

2. System prompts for report parameters (date range, specific flights, etc.).

3. Administrator enters required parameters.

4. System generates and displays the report.

Alternative Flow:

Administrator can choose to export the report in various formats (PDF, CSV, etc.).

Exceptions:

If data is unavailable for the selected parameters, the system notifies the administrator.

Includes: None

Notes/Issues: Ensure report generation doesn't impact system performance during peak hours

4 Other Non-functional Requirements

4.1 Performance Requirements

- **Real-Time Updates:** Flight schedules, baggage status, and security verifications must be processed **within 2 seconds**.
- **Passenger Check-Ins:** The system should complete **check-in and security verification within 5 seconds**.
- **Concurrent Users:** Must support **high traffic loads** during peak hours without performance degradation.
- **Database Efficiency:** Queries for passenger, flight, and baggage information should return results **within 3 seconds**.
- **System Uptime:** AMS should maintain **99.9% availability**, ensuring minimal downtime for airport operations.

These requirements ensure **fast, reliable, and efficient** airport management.

4.2 Safety and Security Requirements

- **Access Control:** Implements **role-based access control (RBAC)** to restrict unauthorized system access.
- **Data Protection:** All sensitive passenger and employee data is **encrypted using AES-256**.
- **No-Fly List Enforcement:** Real-time verification against **security agency databases** to flag restricted individuals.
- **Secure Transactions:** Ensures **secure login mechanisms**, preventing unauthorized data access.
- **Audit Logs:** Maintains **detailed logs** of all security-related activities for compliance and monitoring.
- **Emergency Protocols:** System includes provisions for **manual overrides and emergency alerts** in case of security threats.

These measures ensure **data integrity, passenger security, and regulatory compliance**.

4.3 Software Quality Attributes

-
    - **Reliability:** The system must maintain **99.9% uptime**, ensuring uninterrupted airport operations.
    - **Security:** Implements **AES-256 encryption**, **RBAC**, and **audit logs** for data protection and compliance.
    - **Performance:** Must process **flight updates within 2 seconds** and **passenger check-ins within 5 seconds**.

- **Scalability:** Designed to handle **high concurrent traffic** and future integrations like **biometric authentication**.
- **Usability:** Provides an **intuitive, web-based interface** accessible via modern browsers.
- **Maintainability:** Modular architecture allows **easy updates and system enhancements**.

These attributes ensure **efficiency, security, and long-term adaptability** of the system.

Appendix A – Data Dictionary

Variable Name	Type	Description	Example Values	Related Operations
flight_id	Integer	Unique identifier for a flight	1023, 2045	Create, Read, Update, Delete
airline_name	String	Name of the airline	Air India, Emirates	Read
arrival_time	Timestamp	Flight arrival time	2025-03-01 15:30:00	Read, Update
departure_time	Timestamp	Flight departure time	2025-03-01 17:00:00	Read, Update
passenger_id	Integer	Unique identifier for a passenger	34567, 87654	Create, Read, Update, Delete
passenger_name	String	Full name of the passenger	John Doe, Alice Smith	Read
passport_number	String	Passenger's passport number	A1234567, B9876543	Read, Update

baggage_id	Integer	Unique identifier for baggage	56789, 12345	Create, Read, Update, Delete
baggage_weight	Float	Weight of baggage (kg)	15.5, 23.0	Read, Update
security_status	String	Passenger clearance status	Cleared, Flagged	Read, Update
no_fly_list_status	Boolean	Indicates if passenger is on No-Fly List	True, False	Read, Update
employee_id	Integer	Unique identifier for airport staff	101, 202	Create, Read, Update, Delete
role	String	Employee role	Security, Check-in Staff	Read, Update
shift_start	Timestamp	Employee shift start time	2025-03-01 09:00:00	Read, Update
shift_end	Timestamp	Employee shift end time	2025-03-01 17:00:00	Read, Update

Appendix B - Group Log

Date	Attendees	Topics Discussed	Actions taken
25/02/2025	Jesvin,Aakansh,Ashwin, Gowri	Initial planning and the preparation of the SRS document was started.	Defining roles.
26/02/2025	Jesvin,Aakansh,Ashwin, Gowri	ER diagram and database schema design.	Working on the ER diagram.
27/02/2025	Jesvin,Aakansh,Ashwin, Gowri	Use case model diagram and product overview diagram	Completed the SRS document