A Software Requirements Specification

on

"FLYERVOICE"

Submitted in Partial fulfillment for the award of degree of

Bachelor of Technology

(Information Technology)

Mentor:

Mrs. Shalini Singal

Assistant Professor-II

Submitted By:

Gorang Sharma

Harshvardhan Singh

Deepak sharma

Coordinator:

Dr. Priyanka Yadav

Assistant Professor

Department of Information Technology

Swami Keshvanand Institute of Technology, M & G, Jaipur

Ramnagaria (Jagatpura), Jaipur-302017

Session 2024-2025

Contents

Chapter 1

Introduction

1.1 Purpose

Digital feedback software allows airports and airlines to capture and analyze the voices of passengers regarding airport staff, amenities, and overall experiences. This tool provides real-time feedback, enabling the recognition of key trends and the identification of recurring issues that can be resolved more efficiently. By increasing the volume of passenger feedback by 100 %, airports and airlines can address concerns faster, monitor the performance of staff and tellers, and compare passenger satisfaction across various touchpoints. Ultimately, this system helps enhance passenger delight and overall happiness, leading to improved service quality.

1.2 Scope

The Airport Feedback App helps staff in making every effort to reduce the stress of passengers' journeys. The real-time feedback system has expanded to allow passengers to rate lounges, staff, stores, and more, alongside the cleanliness of facilities. A focus on actionable insights has significantly improved airport productivity and refined service standards to enhance the overall passenger experience.

Our project focuses on business process automation by computerizing various functions of the Airport Feedback App. Through the system, staff can fill in forms, generate multiple copies simultaneously, and directly print the manifest without manual creation, saving time. The system assists staff in tracking their efforts across different working areas, enabling efficient resource utilization by boosting productivity through automation. Additionally, it generates various types of information for multiple uses, ensuring it meets user requirements. The system is designed to be user-friendly for both passengers and operators, offering ease of use, a good interface, scalability, and delivery within the set schedule and budget.

1.3 Definitions, Acronyms, and Abbreviations

- **Admin (Administrator)**
  He has the authority to add/delete users, grant permission to doctors and others.
- **UML (Unified Modeling Language)**
  It is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document.
- **XML (Extensible Markup Language)**
  It is a text-based format that lets developers describe, deliver, and exchange structured data between a range of applications to clients for display and manipulation.
- **HTTP (Hypertext Transfer Protocol)**
  It's a service protocol.
- **JSON (JavaScript Object Notation)**
  It is a text format for storing and transporting data.
- **NPM (Node Package Manager)**
  It is a package manager for the JavaScript programming language.
- **API (Application Program Interface)**
  It's a way for two or more computer programs to communicate with each other.
- **JSX (JavaScript XML)**
  It's a syntax extension to JavaScript that allows us to write HTML in React.

1.4 References

- **Object Oriented Modeling and Design with UML** - Michael Blaha, James Rambaugh.
- **Software Engineering, Seventh Edition** - Ian Sommerville.
- **Wikipedia** - www.wikipedia.com
- **MongoDB** - www.mongodb.com
- **Node.js** - www.w3schools.com
- **Express.js** - www.tutorialspoint.com
- **React.js** - www.reactjs.org

1.5 Technologies to be Used

- **MongoDB** : Non-Relational Database Management System.
- **Git, GitHub** : Version Control System.
- **React.js** : Web Development.
- **Node.js, Express.js** : Web Development.

1.6 Overview

The existing system includes feedback forms and passenger registration, but it has some drawbacks, most notably the absence of machine learning integration. In the proposed system, both passengers and admins will be able to register, and admins will have access to the feedback data. Our plan involves registering users, providing them with flight details, and applying machine learning to generate meaningful trends and results from the collected data. This will help enhance the overall efficiency and effectiveness of the system.

Chapter 2

Literature Survey

2.1 Review Of Related Work

Significant advancements have been made in areas relevant to feedback systems, particularly in user experience (UX) design, sentiment analysis, and data analytics. The following works are pertinent to the development of this project:

- **User Experience Design:** Effective user feedback systems hinge on intuitive UX design. Research by Nielsen Norman Group emphasizes the importance of usability testing and iterative design to enhance user satisfaction and engagement. Tools like Figma and Adobe XD have been utilized to create user-centric designs that improve interaction and feedback collection processes. Studies show that well-designed interfaces significantly increase user participation and the quality of feedback provided.
- **Sentiment Analysis:** With the rise of Natural Language Processing (NLP), sentiment analysis has become a crucial method for interpreting user feedback. Models like VADER and TextBlob are commonly employed to analyze sentiments in user comments, providing insights into customer satisfaction and pain points. Research by Pak and Paroubek (2010) demonstrated that sentiment analysis tools can effectively classify feedback as positive, negative, or neutral, enabling organizations to address user concerns proactively.
- **Data Analytics in Feedback Systems:** Analyzing feedback data allows organizations to uncover trends and improve service quality. Tools like Tableau and Power BI facilitate the visualization of feedback data, enabling decision-makers to identify key areas for improvement. Research by Cheng et al. (2018) indicates that leveraging data analytics enhances decision-making processes by providing actionable insights derived from user feedback.

2.2 Knowledge Gaps

Despite the advancements in these fields, certain knowledge gaps exist that this project aims to address:

- **Integration of Feedback Mechanisms:** While many feedback systems exist, few integrate various feedback collection methods (e.g., surveys, reviews, real-time chat) into a unified platform. The combination of multiple feedback channels remains underexplored, particularly in enhancing the user experience.
- **Real-Time Sentiment Analysis:** While sentiment analysis tools can process user feedback effectively, real-time analysis in feedback systems is still challenging. Delays in processing feedback can hinder timely responses to user concerns, affecting overall satisfaction.
- **Scalability of Feedback Systems:** Many existing feedback applications struggle with scalability, particularly during peak usage times. This can result in data loss or delayed responses, which negatively impacts user trust and engagement.

2.3 Comparative Analysis

The current solutions—user experience design tools, sentiment analysis algorithms, and data analytics platforms—each offer valuable capabilities in their respective areas but also present challenges:

- User experience design tools are crucial for creating engaging interfaces but require extensive user testing to ensure effectiveness in different contexts.
- Sentiment analysis tools, while effective for static feedback, often struggle with nuances in language, particularly in diverse user demographics or informal feedback contexts.
- Data analytics platforms provide robust insights but may lack real-time capabilities, making it challenging to act on user feedback promptly.

2.4 Summary

The Cloud-Based Feedback App aims to integrate these technologies into a unified, real-time platform, addressing existing limitations while leveraging the strengths of current solutions. By utilizing advanced data analytics and sentiment analysis, the app will enhance user engagement and satisfaction by providing timely responses to feedback. This approach will make feedback collection and analysis faster and more accessible, serving as a comprehensive solution for organizations seeking to improve their services and customer experiences.

Chapter 3

Specific Requirements

3.1 Functional Requirements

- **Feedback Submission Module**: Allows users to submit feedback about the airport services via a web-based interface. The feedback includes ratings, comments, and suggestions.
- **Data Storage Module**: Stores feedback data securely in a MongoDB database, ensuring data integrity and availability for analysis.
- **Feedback Analysis Module**: Analyzes the feedback data to extract insights, such as common issues, frequently mentioned areas for improvement, or highly rated services.
- **Web-Based User Interface**: Provides a web-based UI using Node.js, Express.js, and React.js for both passengers submitting feedback and airport management accessing reports.

3.2 Non-Functional Requirements

- **Performance**: The system should perform efficiently, handling multiple feedback submissions concurrently without lag.
- **Scalability**: The system must scale to accommodate a growing number of users, including high traffic during peak travel seasons.
- **Reliability**: The system should be highly available with minimal downtime to ensure feedback can be submitted at any time.
- **Security**: Secure handling of user feedback is critical. The system should use HTTPS encryption and data protection measures for all communications.
- **User Experience**: The system should provide an intuitive, user-friendly interface, making it easy for passengers to submit feedback quickly.

- **Maintainability**: The system should be easy to maintain and update, ensuring that new feedback forms or analytic features can be added without major disruptions.
- **Compliance**: The system should comply with data protection regulations (e.g., GDPR) for handling sensitive user feedback data.

3.3 Hardware Requirements

- **Development Environment:** A system with a minimum of 8GB RAM and 500GB storage.
- **Server Environment:** Cloud-based or on-premise servers running WASCE (WebSphere Application Server Community Edition) for hosting the feedback system.

3.4 Software Requirements

- **Operating System:** Windows 10/11, Linux.
- **Languages:** Node.js, Express.js, React.js, JavaScript.
- **Database:** MongoDB for storing feedback data.
- **Web Server:** WASCE, compatible with any operating system.
- **Web Browser:** Any modern browser for accessing the web-based interface.

3.5 Development Methodology: Waterfall Model

The system development follows the Waterfall model, progressing through clearly defined stages from requirements gathering to deployment. Each phase must be completed before moving on to the next.

The sequential phases in the Waterfall model for the Airport Feedback App development are:

1. **Requirement Gathering and Analysis:** All possible requirements of the Airport Feedback App are captured in this phase and documented in a requirement specification document.
2. **System Design:** The requirement specifications from the first phase are studied, and the system design is prepared. The system design helps in specifying hardware and system requirements and defines the overall system architecture.
3. **Implementation:** The system is first developed in small modules, which are integrated in the next phase. Each module is developed and tested for its functionality (Unit Testing).

4. **Integration and Testing:** All the units developed in the implementation phase are integrated into a complete system and tested for any faults and failures.

5. **Deployment of System:** After thorough testing, the Airport Feedback App is deployed in the customer environment.

6. **Maintenance:** Post-deployment, any issues identified in the live environment are fixed via patches, and newer versions of the app are released to enhance the product.

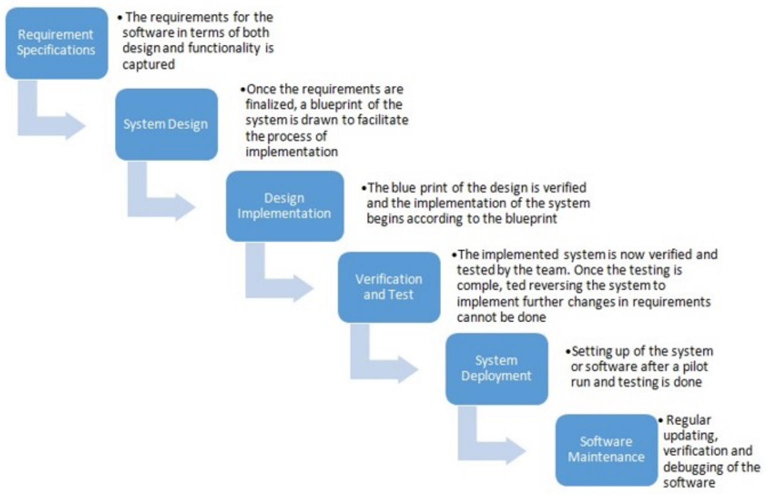

Requirement Specifications
• The requirements for the software in terms of both design and functionality is captured
System Design
•Once the requirements are finalized, a blueprint of the system is drawn to facilitate the process of implementation
Design Implementation
•The blue print of the design is verified and the implementation of the system begins according to the blueprint
Verification and Test
•The implemented system is now verified and tested by the team. Once the testing is comple, ted reversing the system to implement further changes in requirements cannot be done
System Deployment
•Setting up of the system or software after a pilot run and testing is done
Software Maintenance
• Regular updating, verification and debugging of the software

Figure 3.1: Waterfall Model Phases for Airport Feedback App Development

3.6 Business Process Model

- **Feedback Submission Module**
  - Objective: Allow users to submit feedback about their airport experience.
  - Process: Passengers access the feedback portal via the web or mobile app. They select the relevant department (e.g., security, customer service, facilities), provide feedback, and submit it through the interface. The system stores the feedback in a database for further analysis.
- **Feedback Analysis Module**
  - Objective: Analyze the feedback for insights such as satisfaction levels, recurring issues, and potential improvements.
  - Process: The submitted feedback is processed by natural language processing (NLP) algorithms, which categorize it based on sentiment (positive, negative, neutral). The system

identifies patterns in the feedback, such as common complaints or praise. The results are used to generate reports that highlight key areas for improvement.

- **User Interface Module**
  - Objective: Provide a user-friendly interface for passengers to easily submit feedback and for airport management to view analysis results.
  - Process: Passengers interact with the system via a web-based or mobile interface. The interface allows passengers to provide their feedback and view responses (if applicable). Airport management can log in to view detailed reports and summaries of feedback, including charts and metrics related to customer satisfaction.
- **Cloud Integration Module**
  - Objective: Ensure system scalability and availability using cloud services.
  - Process: System components are deployed on cloud services such as AWS or Google Cloud. User feedback is processed in real-time and stored securely in the cloud database. The system scales dynamically based on the volume of feedback submissions.

3.7 Supplementary Requirements

3.7.1 System Availability

The system should ensure 99.9% uptime by utilizing cloud services for high availability. This will be achieved through the use of load balancers, replicated databases, and scalable infrastructure.

3.7.2 Monitoring Mechanism

Monitoring tools like AWS CloudWatch should be implemented to track the system's performance, allowing early detection of issues such as high server load, resource usage anomalies, and potential security threats. Automated alerts will be configured to notify administrators of any critical problems.

3.7.3 Security Measures

The system will use encryption techniques (SSL/TLS) to secure data in transit and at rest. Cloud infrastructure will be protected by firewalls, and role-based access control (RBAC) will be used to manage user permissions.

3.7.4 Backup and Disaster Recovery

Regular automated backups will be scheduled to ensure data is recoverable in the event of a system failure. A disaster recovery plan will be in place to restore the system quickly and efficiently in the event of a critical incident.

3.7.5 Performance Metrics

The system should be able to handle feedback submissions from multiple users simultaneously without affecting performance. Key metrics to be tracked include response time, server uptime, and error rate, all of which should meet predefined thresholds.

Chapter 4

System Architecture

4.1 Client-Server Architecture

The architecture consists of three layers: the Client Layer (React app in the browser) sends HTTP requests to the Business Logic Layer (Node.js server with Express), which handles processing and communicates with the Database Layer (MongoDB). Data is retrieved from MongoDB and sent back to the client through the server.This architecture allows for efficient separation of concerns, where the client focuses on the user interface and interactions, the server handles business logic, and the database manages data storage. It ensures scalable and maintainable web applications, with each layer independently upgradable or replaceable.

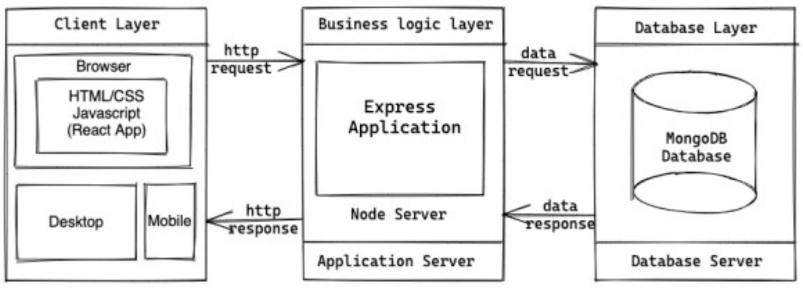


Figure 4.1: Client Server Architecture

4.2 Communications Interfaces

- Client (passenger) on the Internet will be using HTTP/HTTPS protocol.
- Client (system user) on the Internet will be using HTTP/HTTPS protocol.

Chapter 5

Design and Implementation

5.1 Product Features

– **Real-time Feedback Submission:** The system allows passengers to submit feedback in real-time via a web interface built with React. Feedback includes ratings, comments, and suggestions, providing valuable insights for airport management.

– **Data Storage and Management:** Feedback data is stored securely in MongoDB. The system ensures efficient retrieval and management of feedback, allowing the admin to view, filter, and analyze responses for better decision-making.

– **RESTful API Integration:** The system is built on a Node.js backend using Express, handling HTTP requests and responses between the client and the database, ensuring a smooth and scalable feedback collection process.

– **User-Friendly Web Interface:** The platform features a responsive web interface where passengers can easily submit feedback from desktop or mobile devices, ensuring a seamless user experience.

– **Cloud Deployment and Scalability:** The application is deployed on cloud services like AWS, ensuring high availability and scalability. Feedback data and other information are stored securely with MongoDB in the cloud.

5.2 Architecture Diagram

A web application architecture diagram is a visual representation that outlines the structure and components of a web application, illustrating how various parts interact with each other and with external systems. It serves as a blueprint for understanding the system's architecture.

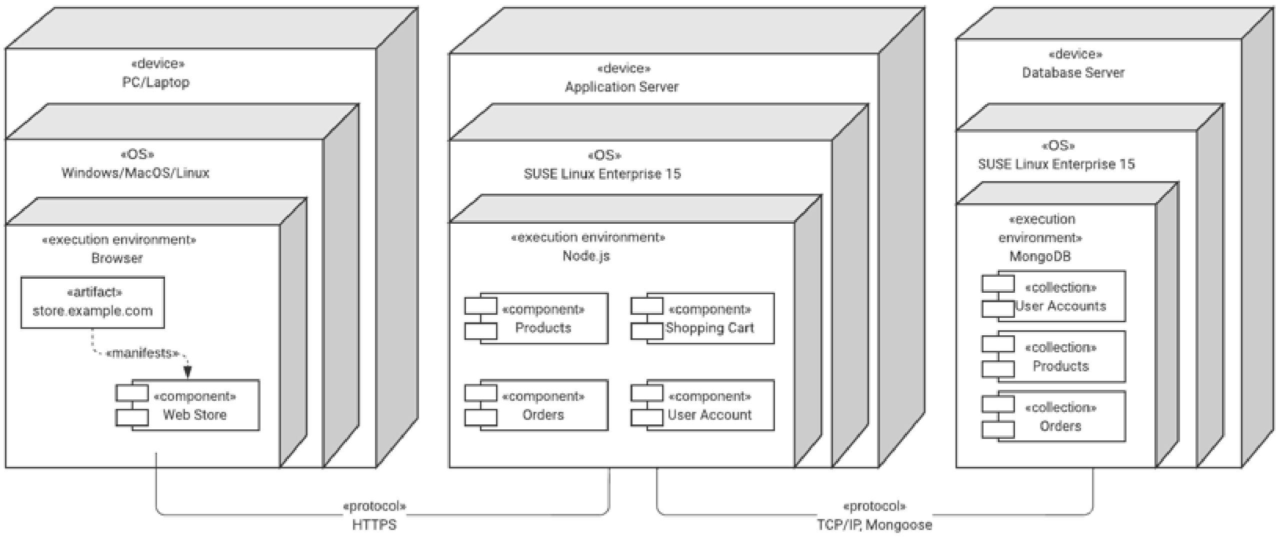


Figure 5.1: Architecture Diagram

5.3 Use-case Model Survey

The Use Case Diagram represents the interactions between the actors (users and system components) and the use cases (specific functionalities of the system).

Actors:

– **Passenger:** Passenger can give feedback regarding the different departments such as airlines, check-in counter, food court, help desk, lounge, store, etc.

– **Admin:** Admin has the authority to view or access the feedback provided by the passengers. Admin can also perform different types of operations on the feedback provided by the passengers to conclude different results.

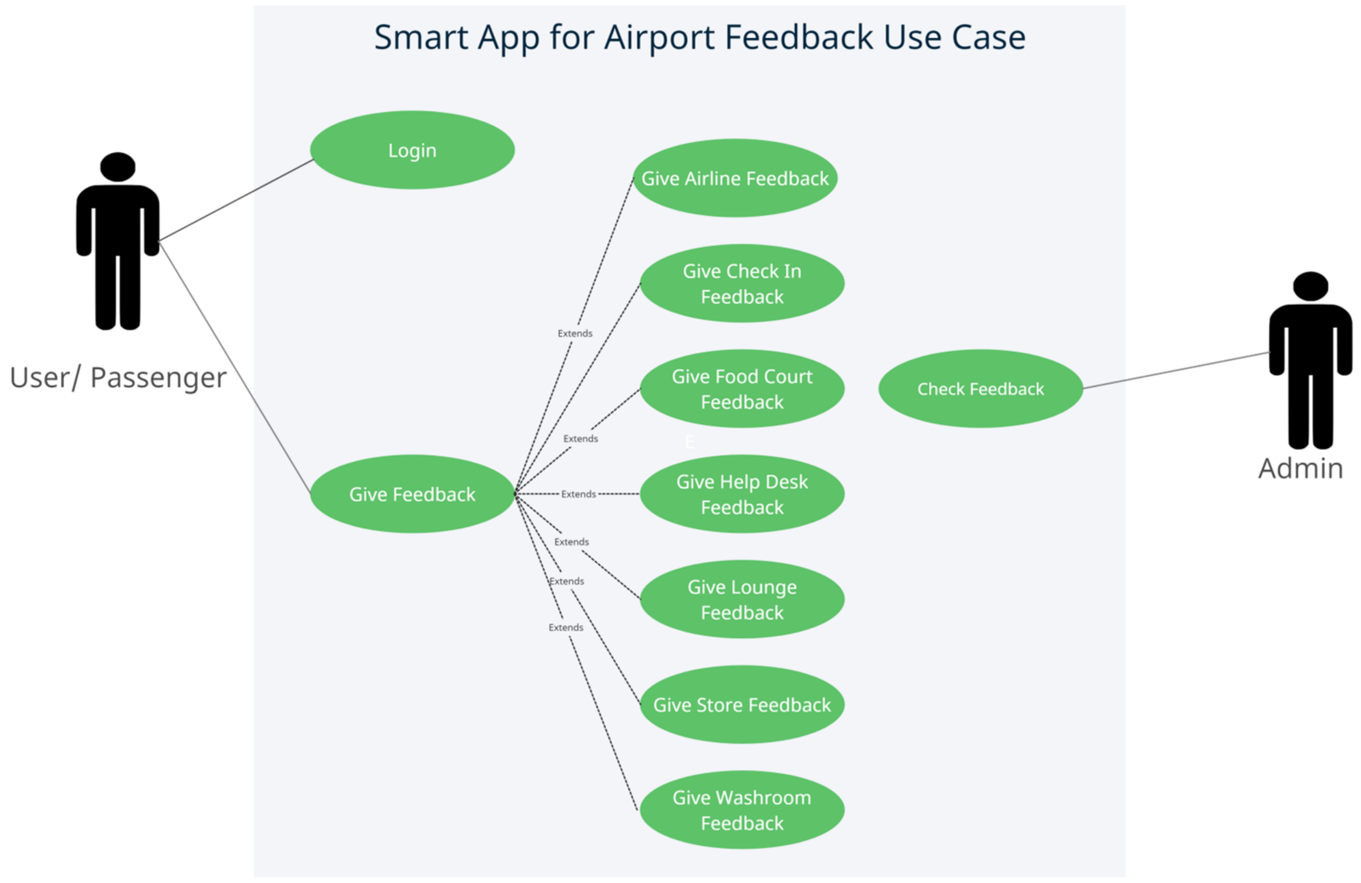


Figure 5.2: Use Case Diagram

5.4 Use Case Reports

5.4.1 Passenger Use-case Report

Use Case	Description
Enter Details	User needs to enter their personal details, including PNR number.
Give Feedback	User can provide feedback to the desired department.
View Flight Detail	User can view any flight details using the flight number.

Table 5.1: Passenger Use-case Report

5.4.2 Admin Use-case Report

Use Case	Description
Login	Admin needs to log in to access their account.
Check Feedback	Admin can review feedback and apply filters to generate the required results.
Reset Password	Admin can reset their account password.

Table 5.2: Admin Use-case Report

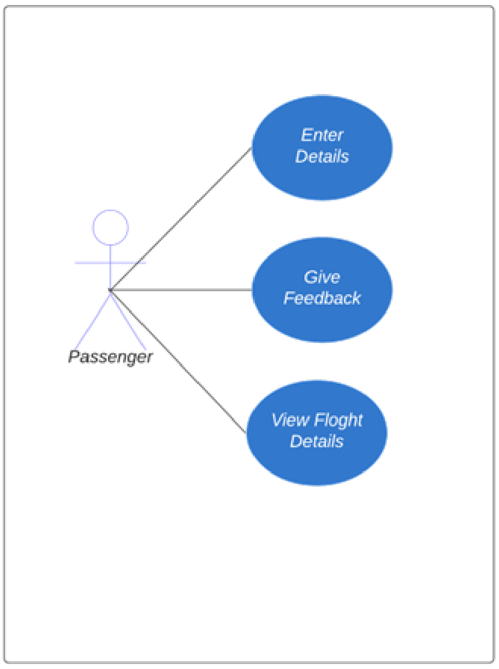


Figure 5.3: Use case diagram for Passenger

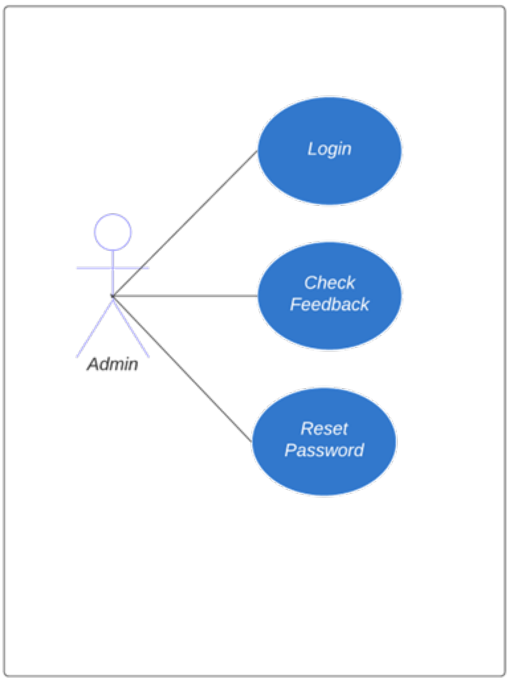


Figure 5.4: Use case diagram for Admin

5.5 ER Diagram

The diagram represents an **Entity-Relationship (ER) model** for an airport feedback system. It shows various entities involved, such as **User**, **Admin**, and different feedback types like **Airline Feedback**, **Check-In Feedback**, **Food Court Feedback**, etc.

– The **User** entity collects details like name, email, mobile number, and PNR, and provides feedback across multiple categories.

– **Admin** can check the feedback and has attributes like name, email, and password.

– Each type of feedback (e.g., **Airline Feedback**, **Food Court Feedback**, **Lounge Feedback**) captures specific attributes such as *ratings*, *service*, *staff*, *cleanliness*, *value for money*, and *recommendations*.

– The **ISA** (Integrated Feedback System) centralizes or aggregates feedback data from multiple areas, facilitating easy feedback management.

This ER diagram outlines a comprehensive feedback system allowing both passengers and airport administrators to interact with and manage feedback across various airport services.

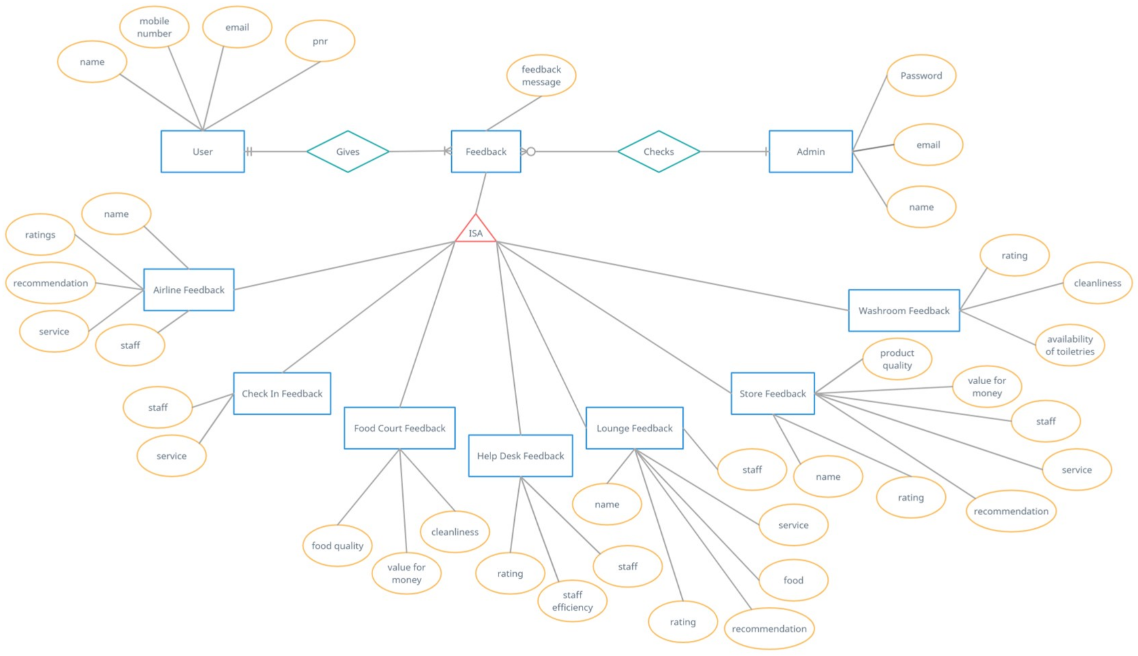


Figure 5.5: ER Diagram

5.6 Databse Design

Database design is the process of defining the structure, organization, and constraints of a database. It involves creating a blueprint for how data will be stored, accessed, and managed within a database management system (DBMS).

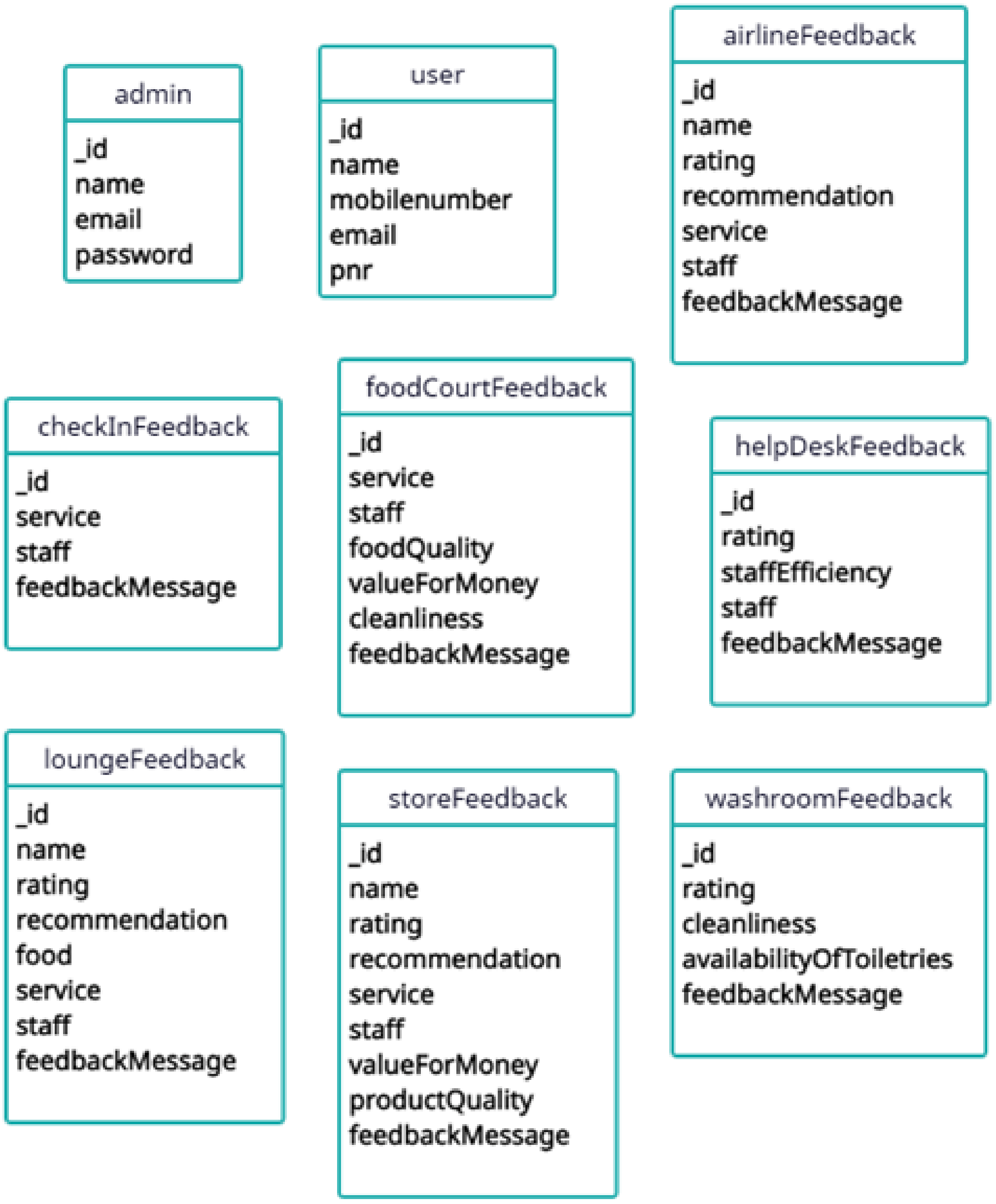


Figure 5.6: Database Design Diagram

Tech Titans	Version: 1.1
Software Requirements Specification	Date: 10-12-2024

5.7 Behavior Diagrams

5.7.1 Activity Diagram

Passenger Registration Activity

First of all, the user enters his details. If all the details are correct, he is redirected to the home page. If any detail is incorrect, then an error message is shown, and the user needs to correct the details to redirect to the home page.

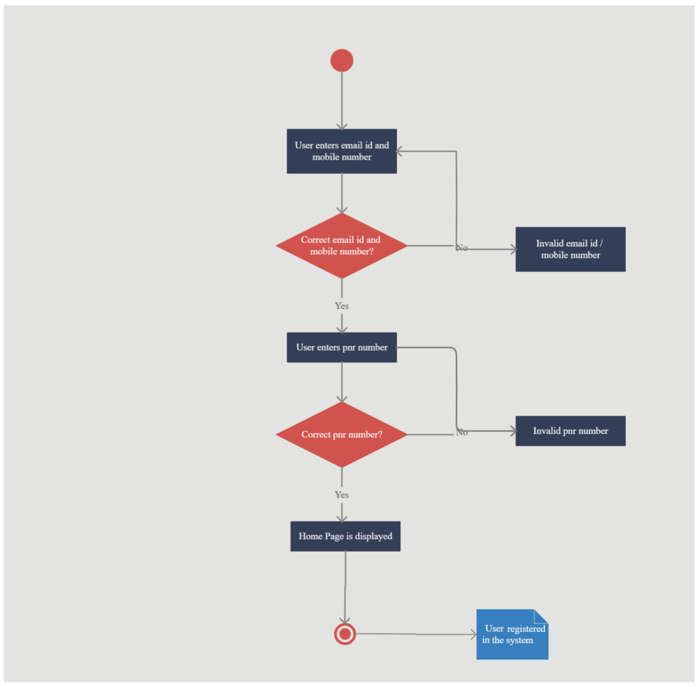
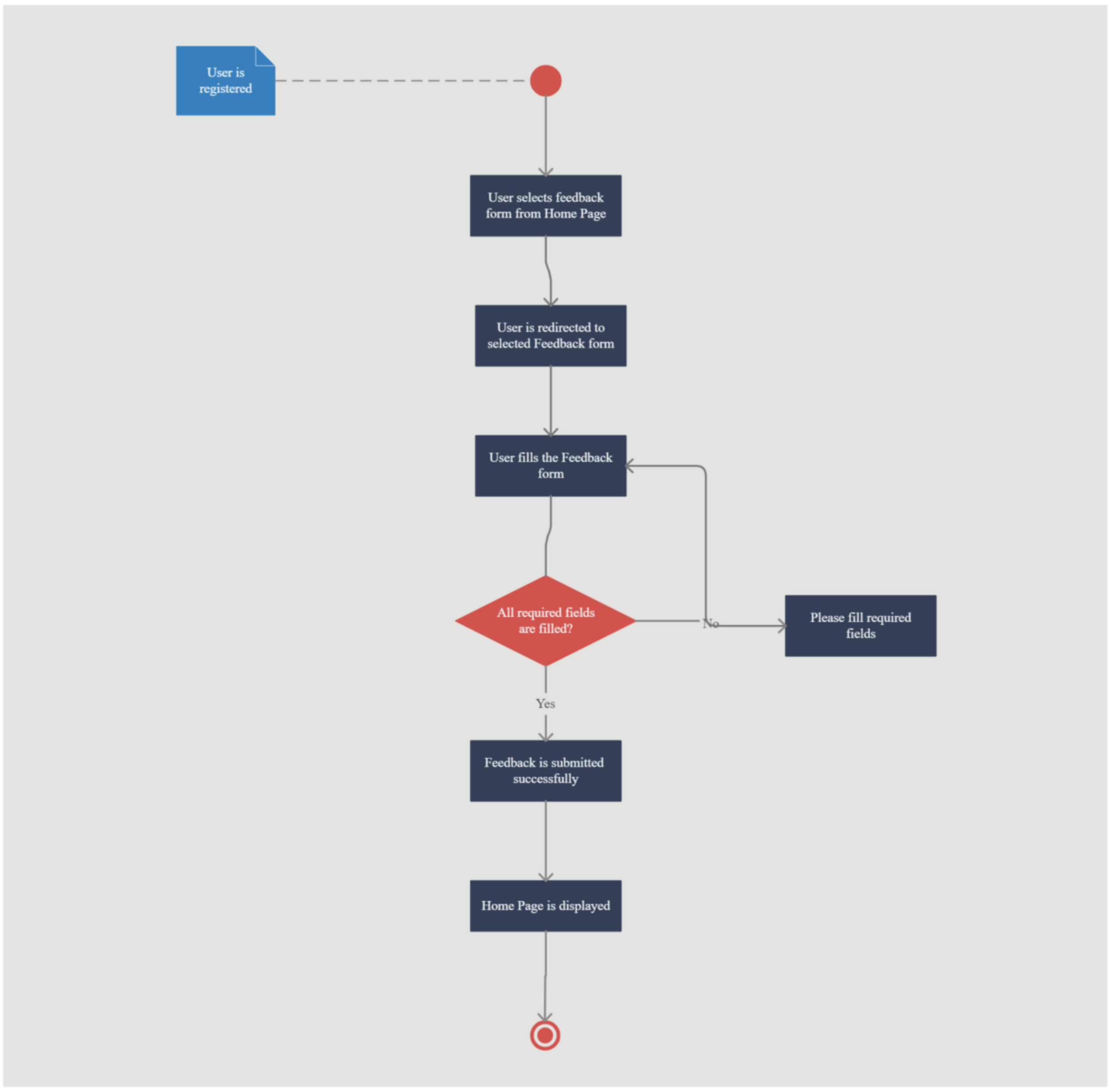

User enters email id and mobile number
Correct email id and mobile number?
No
Invalid email id / mobile number
Yes
User enters pnr number
Correct pnr number?
No
Invalid pnr number
Yes
Home Page is displayed
User registered in the system

Figure 5.7: Passenger Registration Activity Diagram

Passenger Feedback Activity

The user selects the feedback he wants to give and enters all the feedback details. Entering all the feedback details is mandatory for the user to submit the feedback. If any detail is not entered by the user, he will get an alert message indicating that he needs to give the specified rating to proceed. Entering all the feedback details is mandatory for the user to submit the feedback

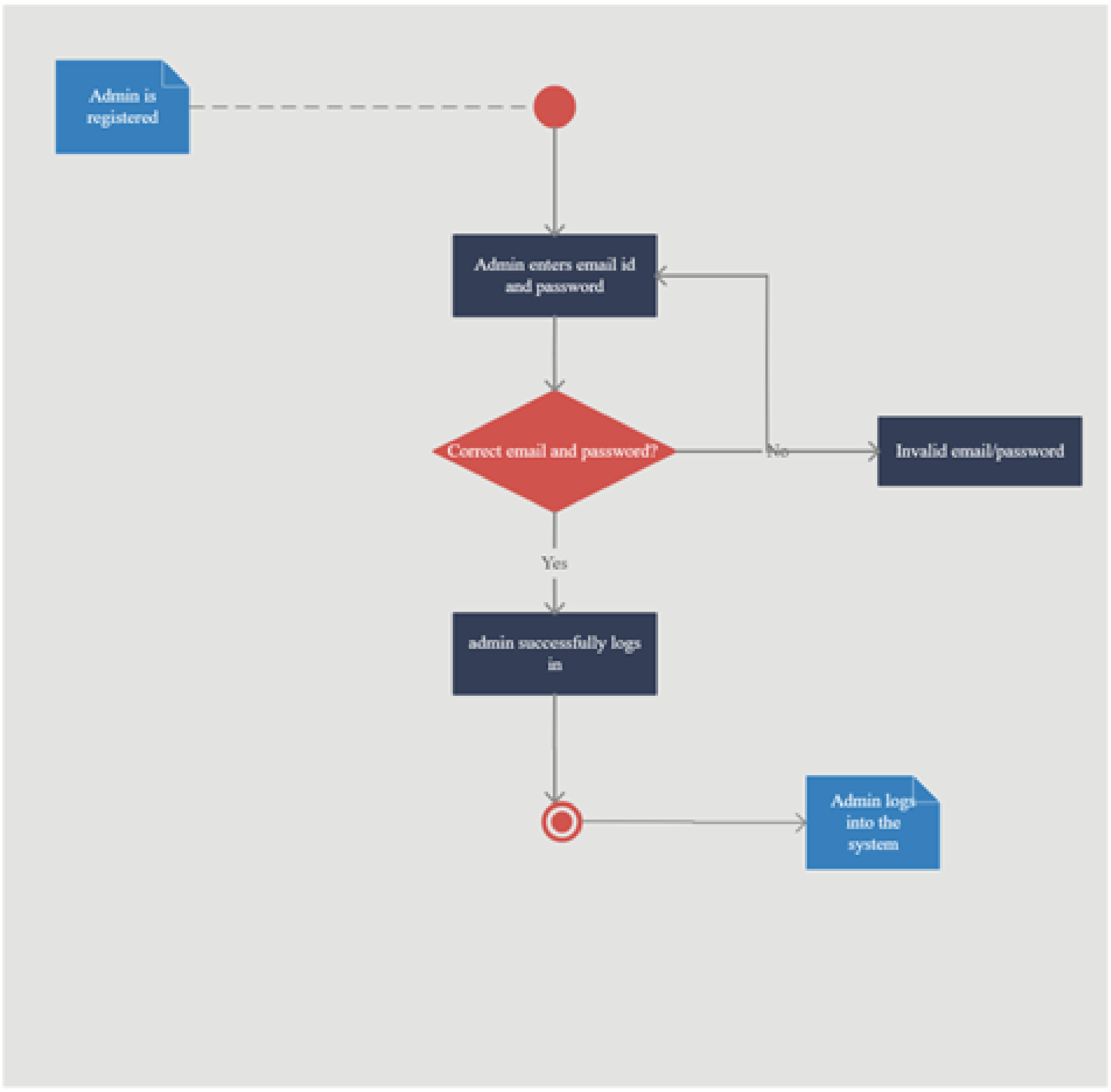


User is registered
User selects feedback form from Home Page
User is redirected to selected Feedback form
User fills the Feedback form
All required fields are filled?
No
Please fill required fields
Yes
Feedback is submitted successfully
Home Page is displayed

Figure 5.8: Passenger Feedback Activity Diagram

Admin Login Activity

The admin needs to log in to check the feedback given by the users. If the entered credentials are correct, then he will be redirected to the feedback page; otherwise, he would be shown an alert message to enter the correct details.



Admin is registered
Admin enters email id and password
Correct email and password?
No
Invalid email/password
Yes
admin successfully logs in
Admin logs into the system

Figure 5.9: Admin Login Activity Diagram

Admin View Feedback Activity

The admin can select the feedback he wants to check at the dashboard, and he can apply desired filters to retrieve the specific feedback.It is very necessary as main work of the entire application is data gathering which it does efficiently

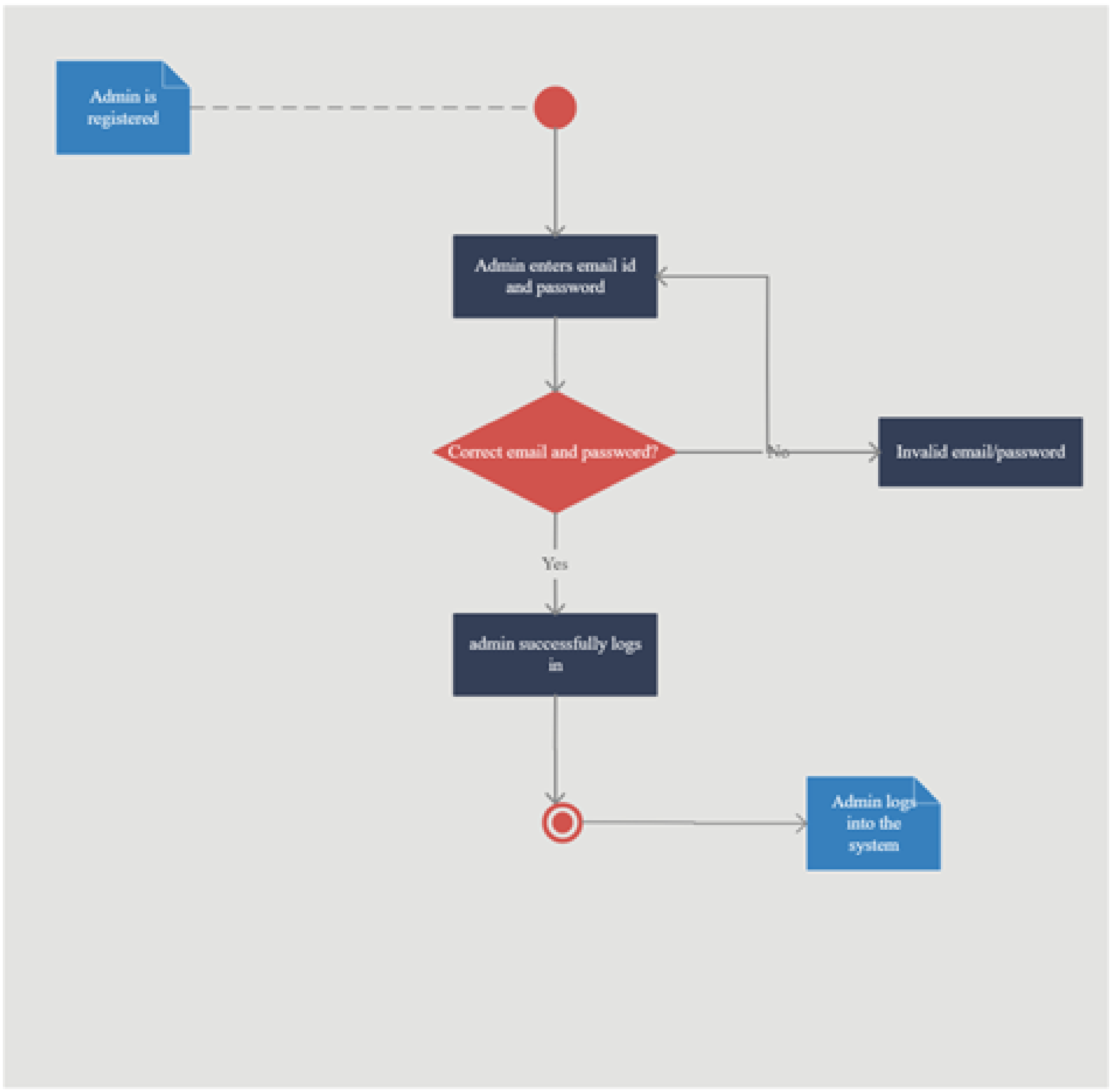

Admin is registered
Admin enters email id and password
Correct email and password?
No
Invalid email/password
Yes
admin successfully logs in
Admin logs into the system

Figure 5.10: Admin View Feedback Activity Diagram

5.8 Sequence Diagram

A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that shows how objects interact in a particular scenario of a use case. It emphasizes the time order of messages exchanged between objects and represents the sequence of events that occur as part of a specific process or operation.It emphasizes the time order of messages exchanged between objects and represents the sequence of events that occur as part of a specific process or operation.

5.8.1 Passenger Sequence Diagram

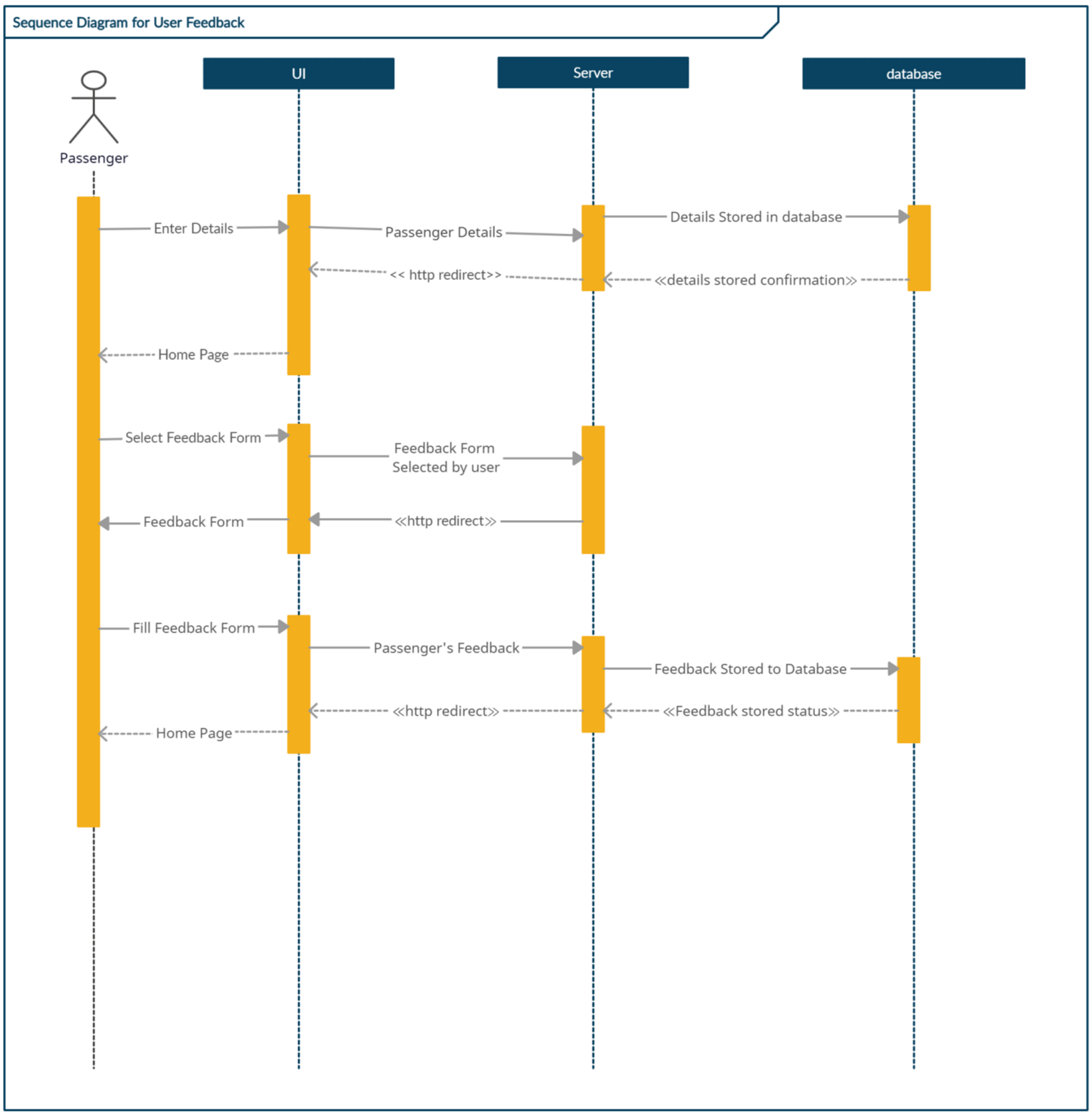


Figure 5.11: Passenger Sequence Diagram

5.8.2 Admin View Feedback Sequence Diagram

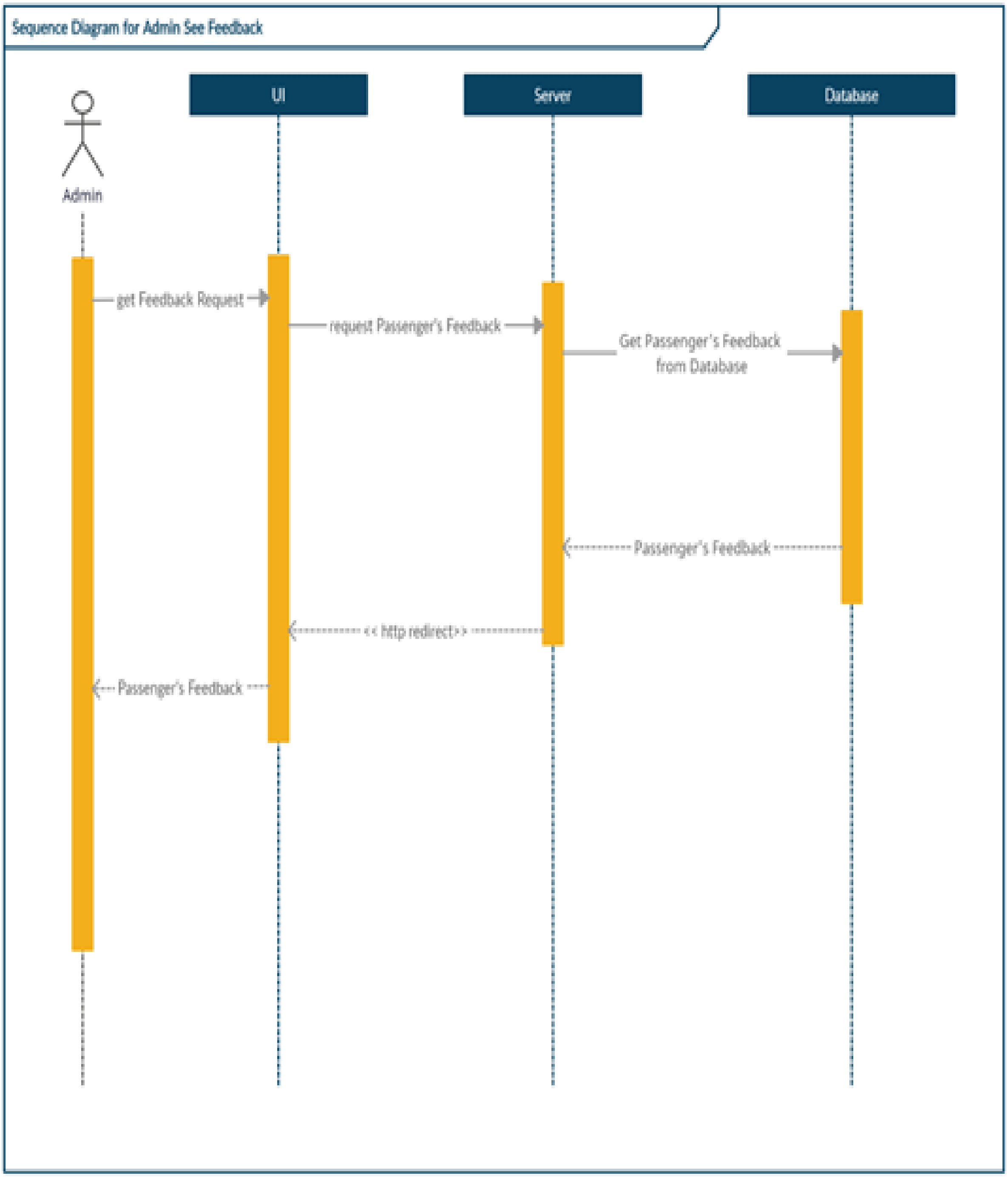

Sequence Diagram for Admin See Feedback
Admin
UI
Server
Database
get Feedback Request
request Passenger's Feedback
Get Passenger's Feedback from Database
Passenger's Feedback
<< http redirect>>
Passenger's Feedback

Figure 5.12: Admin View Feedback Sequence Diagram

5.9 Assumptions and Dependencies

5.9.1 Assumptions

- **Internet Connectivity:** It is assumed that users will have a stable and fast internet connection to access the feedback app, as submitting feedback and retrieving responses rely on consistent connectivity.
- **User Technical Knowledge:** It is assumed that users (travelers, airport staff, etc.) possess basic knowledge of using mobile applications or web interfaces to submit feedback and understand the feedback process.
- **Availability of Mobile and Web Services:** The app assumes that mobile and web services, such as hosting and backend APIs, are available and functioning correctly. Any downtime may impact users' ability to submit feedback or view responses.
- **Supported Feedback Formats:** It is assumed that users will provide feedback in standard formats (text, ratings, etc.) that the app can process. Users should follow guidelines for providing constructive feedback to enhance the system's effectiveness.

5.9.2 Dependencies

- **Feedback Processing Modules:** The app relies on modules for analyzing and categorizing user feedback. Any disruptions in these services may affect the app's ability to provide timely responses to user submissions.
- **Data Storage Solutions:** The app depends on cloud storage solutions to securely store user feedback and related data. The performance and reliability of the app are contingent upon the availability of these storage services.
- **Notification Services:** The app uses notification services to inform users of updates or responses to their feedback. Any issues with these services could lead to delays in communication and user engagement.
- **Browser and Device Compatibility:** The app's functionality depends on compatibility with various devices and browsers, ensuring a seamless experience across platforms, including iOS, Android, Chrome, and Safari.

Chapter 6

Supporting Information

6.1 List Of Figures

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The Airport Feedback App effectively facilitates the collection and analysis of user feedback, providing airport authorities with valuable insights to enhance passenger experience. By leveraging cloud technologies, the app ensures scalability, high availability, and secure data storage. The platform streamlines the feedback process for travelers and airport staff by automating feedback collection and analysis, ultimately improving service quality and operational efficiency.

7.2 Future Scope

– **Support for Multilingual Feedback:** Expanding the app's capabilities to support feedback submission in multiple languages would increase accessibility for international travelers.

– **Real-Time Analytics Dashboard:** Developing a real-time analytics dashboard would allow airport authorities to monitor feedback trends and respond promptly to emerging issues, enhancing decision-making.

– **Mobile App Development:** Creating a dedicated mobile app version would enable users to easily submit feedback and receive notifications directly from their smartphones, improving user engagement.

– **Integration with Airport Services:** Future enhancements could include integrating the app with other airport services (e.g., flight status updates, baggage tracking) to provide a more comprehensive experience for users.

Chapter 8

Concerns/Queries/Doubts if any

8.1 Current Constraints

- **Limited Customization for Feedback Responses:**
  - Concern: Users have expressed the need for more customization options in how their feedback is acknowledged and addressed by airport staff.
  - Impact: The lack of flexibility in personalizing responses may hinder user satisfaction and engagement with the feedback process.