A logo with a city silhouette

Description automatically generated

**CIS 634 – OBJECT ORIENTED SOFTWARE ENGINEERING**

**SOFTWARE PROJECT FINAL REPORT**

**Project Title: Airlines Database Management System**

**Team members: Megha Suresh Bhat [2854046], Shivani Dineshbhai Bodara [2822188], Nishita Vijaykumar Modi [2867673]**

**Date:** **December 7th, 2023**

**Table of Contents**

1. Introduction……………………………………………………………………………………3

1.1 Purpose and scope………………………………………………………………..3

1.2 Product Overview………………………………………………………………….3

2. Project Management Plan……………………………………………………………….4

2.1 Project Organization………………………………………………………………4

2.2 Lifecycle model used……………………………………………………………..5

2.3 Risk Analysis………………………………………………………………………….6

2.4 Hardware and software resource requirement……………………..6

2.5 Deliverables and schedule…………………………………………………….7

3. Requirement specification………………………………………………………………7

3.1 Stakeholders for the system………………………………………………….9

3.2 Use cases………………………………………………………………………………11

3.3 Rationale for your use case model………………………………………..14

3.4 Non-functional requirements………………………………………………..16

4. Architecture……………………………………………………………………………….…..18

4.1 Architectural styles used………………………………………………………..18

4.2 Architectural model……………………………………………………………….19

4.3 Technology and software used………………………………………………19

4.4 Rationale for your architectural style and model…………………...19

5. Design……………………………………………………………………………………………..20

5.1 User interface design……………………………………………………………..20

5.2 Components design………………………………………………………………..22

5.3 Database design…………………………………………………………………….28

5.4 Traceability from requirements to detailed design models……..29

6. Test management……………………………………………………………………………29

6.1 A complete list of system test cases……………………………………….29

6.2 Traceability of test cases to use cases……………………………………33

6.3 techniques used for test case generation……………………………..36

6.4 Test results and assessments………………………………………………..36

6.5 Defect reports……………………………………………………………………….38

7. Conclusion………………………………………………………………………………………39

References…………………………………………………………………………………………40

**List of Figures**

1. Lifecycle model used (section 2.2)

2. Graphical use case model (Section 3.2.1)

3. Architecture Model (section 4.2)

**List of Tables**

Section 5.2

Airline information

Airlines

Flights

Flight availability

Airline Routes

Section 5.5

Requirement traceability Matrix for User Authentication System

1. **Introduction**

**1.1. Purpose and Scope**

The main objective or purpose of the project is to develop a computerized airline database system to reduce the paperwork and increase the convenience in handling the airline management operations. Implementing this will have a significant increase in the efficiency of the database software and thereby reducing human efforts to a greater extent. The project provides inform

ation about the airline record, flight record, airport record, arrival and departure timing of the flights, fare record, route record, booking option and payment methods to book a flight ticket.

The scope of the project focuses on two perspectives. One from the user perspective and the other from the administrator point of view. The users shall be able to the list of flights as per their travel itinerary and can proceed to book their tickets on the payment portal. The administrator shall be responsible for managing the flights such bringing in a new flight to the database, updating the existing one and removing a flight which is no longer in operation from the database.

**1.2. Product Overview**

Airlines database management system is a comprehensive design which is streamlined and optimized for airlines operations, enhancing efficiency, and improving the overall passenger experience.

The key features or capabilities of the software product are as follows:

* **Flight management:** Administrator has the authority to create a new flight, update the existing one and delete the one which is no more operational. The users will be able to view these flights as per their travel itinerary preference.
* **Scalability and performance:** The database can handle growing volume of data users and can optimize queries and database performance for efficient data retrieval and processing.
* **User access:** Implemented role-based access control which provides access based on the roles to ensure that only authorized personnel such as the administrator can have access to sensitive information.

**2. Project Management Plan**

**2.1. Project Organization**

* **Project goals and objectives:** The goal and objectives of the project was to build an airline database management that makes the online airline booking for the user much easier. We intended on developing a user-friendly platform to have a smooth operation between the user and the system. The project also was intended to provide a separate access for the admins to take care of the flight database as per the requirement.
* **Project Scope:** The project scope comprises of basic operations such as creation of the flight database, updating the existing one and deletion of flight records. Providing access to users and admins. Making sure that the users can view the flight list as per their travel itinerary and mange to process their booking. The project does not comprise of any real time data handling, crew management, financial reports, baggage claim domain, aircraft maintenance.
* **Project Team:** Our project team comprises of 3 members. Megha, Shivani and Nishita. Megha and Shivani were involved in the coding of the project and bringing together all the functionalities and deliverables together. Nishita was mainly involved in the documentation part of the project and acted as a significant tester to run the project as a user.
* **Project workflow:** The workflow began initially by brainstorming about the ideas we can take up for our project. Once the topic was decided, we went ahead with laying out the functionalities of the project such as the admin and user roles in the software, how they would navigate in the system, how to build a user-friendly system, what features needs to be implemented and so on by whiteboarding method.We started coding all these functionalities step by step. Once the software was ready, we tested all the operations by making one of our team members as a user and checked all the working flow.

**2.2. Lifecycle Model Used**

Waterfall life cycle model has been implemented in this project.

A diagram of software development

Description automatically generated

Requirements in software such as the programming language to be used, frameworks, libraries, database were decided well in advance. With all these requirements, the software was developed phase by phase. Every step was initialized carefully as we did not have time to go back and rectify any step due to time constraint and because of the model we opted for implementation. Once the software was ready, we tested each functionality of the component which is the unit testing. The results were submitted to us as expected. In the end, we tested the entire software as a user using the system for the first time as a part of system testing. Now the system is fully operational and must be maintained for an efficient usage.

**2.3. Risk Analysis**

* **Integration issues:** Challenges arouse while integrating one functionality with another. Majorly with the booking and payment portal. We overcame by implementing a clear interface between the two and improvised our code by giving out an easy platform to navigate and avoided a clumsy output.
* **Scalability issues:** With the growing number of users, the software must be able to accommodate such huge volume and we have implemented a larger size database to store all the records.
* **Data accuracy and consistency:** Incorrect flight information can lead to a huge trouble in the future for passengers and airline staff and admins. Having a timely check on the information provided in the database is a mandatory and that is always taken care by the administrator.

**2.4. Hardware and Software Resource Requirements**

The software requirements in our project are as follows:

* **Laravel framework:** The Laravel framework has been implemented in our project. Laravel’s ORM system provides a convenient and expressive way to interact with the database. Working with the database tables and recording the PHP syntax reduces the development of complex SQL queries.
* **Programming languages:** Choosing a suitable programing language is vital. We selected HTTP, PHP, CSS as it is a web-based application and JavaScript for connecting the web pages. SQL coding for creation of the database design.
* **Integrated Development Environment (IDE):** Used visual studio code for coding, debugging and version control.
* **Version control system:** Version control system is crucial for collaborative development and code management. We have used GitHub for hosting repositories.
* **Web server:** Apache server was used to host the application in our project.

**2.5. Deliverables and schedule**

**Schedule:**

* Project initiation: Week 1 – The scope of the project was laid out and the roles and responsibilities of each team member was segregated.
* Requirement gathering and analysis: Week 2 – The requirements of the project was brainstormed, and the system installation was done accordingly.
* System design: Week 3 to 4 – The software system design, considering the water flow life cycle was designed and all the functionalities were laid out. Database schema, system flow was brought into the software.
* Development: Week 5 to 6 – The backend and front-end coding was developed, and the web page integration was implemented.
* Testing: Week 7 to 8 - Unit, integration and system testing was performed on the software to test each functionality of the project.

**3. Requirement Specifications**

1.Functional Requirements:

Functional requirements define the specific functionalities and features the system must provide. For an airline database management system, this includes tasks such as flight booking, schedule management, passenger information handling, and reporting capabilities.

1. User Interface Requirements:

User interface requirements outline the design and usability aspects of the system. This involves creating an intuitive and user-friendly interface for tasks like booking flights, checking schedules, and managing reservations.

1. Performance Requirements:

Performance requirements specify the system's responsiveness, speed, and scalability. For an airline database management system, it includes ensuring quick data retrieval, handling concurrent user interactions, and supporting a growing database size.

1. Security Requirements:

Security requirements detail measures to protect sensitive data. This involves implementing user authentication, encryption for data in transit and at rest, access controls, and ensuring compliance with industry security standards.

1. Database Requirements:

Database requirements define the structure and management of the underlying database. This includes specifications on data storage, relationships between tables (e.g., flights, passengers, airports), and mechanisms for data integrity and normalization.

1. Integration Requirements:

Integration requirements address the need for the system to work seamlessly with external components. This may involve integration with payment gateways, external databases, or other airline systems to enhance overall functionality.

1. Regulatory Compliance Requirements:

Regulatory compliance requirements ensure adherence to aviation standards, safety regulations, and reporting requirements. This involves incorporating features and processes that align with industry regulations to guarantee legal compliance.

1. Reporting and Analytics Requirements:

Reporting and analytics requirements specify the system's capability to generate reports and analyze data. This includes tools for tracking key performance indicators, revenue analysis, and providing insights for decision-making.

1. Documentation Requirements:

Documentation requirements outline the need for comprehensive system documentation. This includes user manuals, developer guides, and any other documentation necessary for system understanding, maintenance, and future development.

1. Testing and Quality Assurance Requirements:

Testing and quality assurance requirements define the testing processes and standards. This involves creating test cases, performing unit testing, integration testing, and ensuring overall system reliability and quality.

These requirements collectively provide a clear and structured framework for the development of a robust airline database management system.

**3.1. Stakeholders for the system**

1. Airline Management:

Airline management represents the leadership overseeing the entire airline database management system. Their concerns include operational efficiency, regulatory compliance, and strategic decision-making based on system-generated insights.

2. IT Administrators:

IT administrators are responsible for the technical aspects of the airline database management system. Their focus includes system deployment, configuration, security, and ongoing maintenance to ensure seamless and secure operations.

3. Developers:

Developers play a key role in creating and maintaining the airline database management system. Their responsibilities encompass coding, debugging, and implementing features to meet functional requirements outlined in the development plan.

4. Passengers:

Passengers are end-users of the system, relying on it for tasks like flight booking, schedule checking, and obtaining accurate information. Their satisfaction depends on the system's user-friendly interface and reliability.

5. Regulatory Authorities:

Regulatory authorities set and enforce industry standards. Their interest in the airline database management system lies in ensuring compliance with aviation regulations, safety protocols, and reporting requirements.

6. Ground Crew and Staff:

Ground crew and staff involved in day-to-day airline operations depend on the system for tasks like check-ins and managing flight schedules. Their concerns include access to real-time information for efficient ground operations.

7. Marketing Team:

The marketing team utilizes system data for market analysis, customer preferences, and promotional campaigns. Their stake involves leveraging the system to inform marketing strategies and enhance customer engagement.

8. Security Analysts:

Security analysts are tasked with safeguarding sensitive data within the system. Their responsibilities include implementing robust security measures, conducting risk assessments, and ensuring the system adheres to industry security standards.

9. Customer Support:

Customer support stakeholders focus on assisting passengers using the system. Their concerns encompass system usability, addressing user queries, and ensuring a positive overall customer experience with the system.

**3.2. Use cases:**

3.2.1. Graphic use case model

A diagram of a user

Description automatically generated

A diagram of a person

Description automatically generated

**3.2.2. Textual Description for each use case**

Use Case 1: Flight Booking

Actor: Passenger

Description: The Passenger initiates the process of searching, selecting, and booking a flight. They provide necessary details, choose seats, and finalize the booking, with the system updating the database and confirming the reservation.

Use Case 2: Check Flight Schedule

Actor: Passenger

Description: The Passenger accesses the system to check the schedule of available flights. They can view details such as departure and arrival times, flight numbers, and seat availability, enhancing their ability to plan and make informed decisions.

Use Case 3: Manage Reservations

Actor: Passenger

Description: Passengers can manage their reservations, including modifying flight details, changing seats, or canceling bookings. The system ensures that the database is updated accordingly and reflects the latest passenger preferences.

Use Case 4: System Configuration

Actor: System Administrator

Description: The System Administrator configures the system settings, manages user accounts, and ensures overall system functionality. This includes database maintenance, security configurations, and monitoring system health.

Use Case 5: User Authentication

Actor: System Administrator

Description: The System Administrator oversees user authentication processes, managing access control and ensuring that only authorized personnel can access sensitive functionalities within the airline database management system.

Use Case 6: Flight Management

Actor: Airline Staff

Description: Airline Staff use the system to manage flights, including updating schedules, assigning crews, and making necessary adjustments. The system updates the database to reflect the latest flight information.

Use Case 7: Booking Agent Support

Actor: Booking Agent

Description: Booking Agents interact with the system to assist passengers in making reservations. They can check seat availability, book flights on behalf of passengers, and manage bookings to provide efficient customer support.

**3.3. Rationale for your use case model**

The use case model for the airline database management system is designed to capture key interactions and functionalities that address the diverse needs of stakeholders, including passengers, airline staff, system administrators, and booking agents. The rationale for this use case model is grounded in several considerations:

User-Centric Approach:

The model prioritizes user interactions to ensure that the system is user-friendly and meets the needs of different stakeholders. Passengers can easily book flights, check schedules, and manage reservations, while staff and administrators have tools for efficient system management.

Comprehensive System Coverage:

Actor Identification:

The identified actors (Passenger, Airline Staff, System Administrator, and Booking Agent) represent distinct roles within the system. This actor-based approach helps in defining specific user responsibilities and functionalities associated with each role.

Adaptability to Industry Standards:

The use case model aligns with industry standards and best practices in airline management. By incorporating essential functionalities such as flight management, reservation handling, and user authentication, the model ensures that the system complies with aviation regulations and meets the industry's operational requirements.

Flexibility and Scalability:

The model allows for flexibility and scalability in system functionalities. For instance, the "Flight Management" use case accommodates changes in flight schedules and crew assignments, demonstrating the system's adaptability to evolving airline operations.

Efficient Customer Support:

The inclusion of the "Booking Agent Support" use case recognizes the importance of providing efficient customer support. Booking agents have specific functionalities to assist passengers, enhancing the overall experience and addressing customer queries and needs effectively.

System Security and Administration:

The model addresses system security and administration through use cases like "System Configuration" and "User Authentication." System administrators can configure the system settings, manage user accounts, and ensure secure access, contributing to the overall reliability and integrity of the system.

Balanced System Operation:

By encompassing both passenger-centric and administrative use cases, the model maintains a balance in addressing the operational needs of different stakeholders. This ensures that the system not only provides a positive experience for passengers but also facilitates effective management and control for airline staff and administrators.

**3.4. Non-functional requirements**

Usability:

The system should have an intuitive and user-friendly interface for both passengers and staff.

The user interface should be responsive and accessible, ensuring a positive user experience.

Performance:

Response times for critical operations, such as flight booking and schedule checking, should be within acceptable limits to provide a seamless experience.

The system should handle concurrent user interactions efficiently, especially during peak booking times.

Reliability:

The system should be available 24/7, with minimal downtime for maintenance.

Data integrity and consistency should be maintained even during system upgrades or modifications.

Security:

User authentication and authorization mechanisms should be robust to prevent unauthorized access.

Sensitive data, including passenger information and flight schedules, should be encrypted both in transit and at rest.

Scalability:

The system should scale gracefully to accommodate a growing number of users and an expanding database.

Performance should not degrade significantly as the system load increases.

Maintainability:

The system should be easily maintainable, allowing for updates, patches, and bug fixes without disrupting regular operations.

Documentation for system administrators and developers should be comprehensive and regularly updated.

Compatibility:

The system should be compatible with various web browsers to ensure accessibility for a wide range of users.

Integration with external systems, such as payment gateways and regulatory databases, should be seamless.

Regulatory Compliance:

The system must comply with aviation industry regulations and standards.

Regular audits and assessments should be conducted to ensure ongoing compliance.

Performance Monitoring and Logging:

The system should have robust monitoring capabilities to track performance metrics and identify potential issues.

Logging mechanisms should capture relevant events and errors for effective troubleshooting.

Data Backup and Recovery:

Regular automated backups of the database should be performed to prevent data loss.

Efficient data recovery mechanisms should be in place in case of system failures or data corruption.

Training and Support:

Adequate training materials and sessions should be provided for system users and administrators.

A responsive support system should be in place to address user queries and issues promptly.

**4. Architecture**

**4.1. Architectural style(s) used.**

The chosen architectural style is Microservices, where the system is divided into independent, modular services, each responsible for specific functionalities. This approach enhances scalability, flexibility, and ease of maintenance by allowing individual services to evolve independently.

**4.2. Architectural model (includes components and their interactions)**

A diagram of a software process

Description automatically generated

**4.3. Technology and software used.**

Technology Stack:

Programming Languages: PHP, JavaScript (for front-end interactivity).

Framework: Laravel (PHP framework for back-end development).

Web Technologies: HTML, CSS, JavaScript.

Database: MySQL (Relational Database Management System).

Containerization: Docker for packaging and deploying microservices.

API Design: RESTful APIs for communication between microservices and external interfaces.

**4.4. Rationale for your architectural style and model**

Architectural Style Rationale:

Microservices architecture is chosen for its ability to provide modularity, scalability, and independent development of services. This aligns with the dynamic nature of airline operations, where different aspects (booking, schedules, authentication) can evolve independently.

Architectural Model Rationale:

The chosen model emphasizes modularity and separation of concerns. User interfaces interact with specific microservices, ensuring a clear and efficient flow of data. The centralized database simplifies data management, and external interfaces enhance system capabilities.

Technology Rationale:

The technology stack, including Laravel for back-end development and Docker for containerization, is selected for its balance of performance, ease of development, and scalability. MySQL is chosen for its reliability and compatibility with the system's relational data requirements.

Hardware Rationale:

Cloud-based infrastructure is preferred for its scalability, enabling the system to adapt to varying workloads and ensuring high availability. This choice aligns with modern trends in scalable and flexible deployment architectures.

**5. Design**

**5.1. User Interface design**

**Flight Booking Interface Design**

**Layout and Structure**:

1. **Home Page**:
   * **Top right corner**: Includes links to **Login** and **Register** for user login and registration.
   * **Central Booking Section**: Features a simple form with fields for ‘From', 'To’, ‘Departure’, 'Return' (optional for one-way trips).
   * **Submit Button**: A prominent button labeled ‘Submit' initiates the search process.
2. **Search Results Page**:
   * **Flight Listings**: There are two lists for departure and arrival flights. Each listing shows key flight details such as airline name, flight number, and flight model with associated book button with each listing.
3. **Flight Details and Booking Confirmation**:
   * After selecting a flight, users are taken to a login page if they are not logged in.

Otherwise, they are shown user detail page.

* Users can proceed to book by entering passenger details and payment information.
  + A confirmation page displays a confirmation message that says thank you for booking with us.

**Design Elements**:

1. **Color Scheme**: A calming palette of blue and white, evoking a sense of trust and professionalism.
2. **Typography**: Clear, legible fonts for easy reading.

**Usability Features**:

1. **Responsive Design**: The interface is fully responsive, ensuring a consistent experience across devices.
2. **Autocomplete**: Autocomplete functionality for airport codes to speed up the booking process.

**User Flow**:

1. User enters travel details on the home page.
2. User views and filters search results, selects a flight.
3. User completes the booking process with passenger and payment details.
4. User can see the booking confirmation message.

**5.2. Components design**

1. **Flight Information**:

* Stores details about flights, including flight numbers, airline name and flight model for departure and arrival flights.

**Static Model of Flight Information**

**Example**:

1. **Flight Route Information**:
   * Departure Airport: SAN
   * Arrival Airport: (LAX)
   * Aircraft: Boeing 777
2. **Schedule Template**:
   * Days of Operation: Monday,Wednesday,friday
   * Departure date: 12/10/2023
   * Arrival Date: 12/22/2023
3. **Passenger Records**:

* Personal information of passengers such as name, contact details, and credentials.

**Static model of passenger details:**

* **Name**: John Doe
* **Email id**: johndoe@gmail.com

1. **User Management and Access Control**:

* User accounts for employees with different access levels based on roles.

**Static Model of User Management and Access Control**

* + **Name:** Jane Smith
  + **Role:** Administrator
  + **Email Address:** [janesmith@gmail.com](mailto:janesmith@gmail.com)

**Role Definitions:**

**System Administrator:**

Full access to the entire system.

* + - Can modify user roles, manage access controls, and oversee system security.

**Security Settings**:

**Data Permissions:**

* Sensitive Data Access: Only accessible by authorized roles (e.g., HR data is only accessible by system administrators).
* Read-Only Access: For users who need to view but not alter data.

1. **Airline information:**

**Static Model of Airline Information**

Table: Airlines

| Field Name | Data Type | Description |
| --- | --- | --- |
| AirlineID | Integer / UUID | Unique identifier for each airline. |
| Name | Varchar(255) | The official name of the airline. |
| Email | Varchar(255) | Contact email address for the airline. |
| CreatedAt | Timestamp | The date and time when the airline record was created. |
| UpdatedAt | Timestamp | The last date and time the record was updated. |

Example Record:

* AirlineID: 1
* Name: "United Airlines"
* Email: unitedairlines@gmail.com
* CreatedAt: 2020-01-15 08:30:00
* UpdatedAt: 2023-03-10 14:45:00

Explanation:

* AirlineID: A unique identifier (such as a sequential integer or a UUID) that distinguishes each airline in the database.
* Name: The official name of the airline, which is a static piece of information and does not change frequently.
* Email: A contact email for the airline. This may change occasionally but is generally static.
* CreatedAt: The timestamp indicating when the airline's information was first entered into the system. This is a fixed value and does not change.
* UpdatedAt: The timestamp of the most recent update to the airline's record. While this field can change, it is a part of a static model as it doesn't represent ongoing or real-time operations but rather the last point of data modification.

1. Flight information:

**Static Model of Flight Information**

**Table**: **Flights**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| FlightID | Integer / UUID | Unique identifier for each flight. |
| AirlineID | Integer / UUID | Identifier for the airline operating the flight. |
| FlightNumber | Varchar(20) | The flight number as assigned by the airline. |
| FlightModel | Varchar(50) | The model of the aircraft used for this flight. |

**Example Record**:

* **FlightID**: 7
* **AirlineID**: 2
* **FlightNumber**: "DL100"
* **FlightModel**: "Airbus-250"

**Explanation:**

* **FlightID**: A unique identifier (such as a sequential integer or a UUID) for each flight record in the database.
* **AirlineID**: A reference to the airline operating the flight, linking to the airline's unique identifier in the Airlines table.
* **FlightNumber**: The official flight number, a combination of letters and numbers, used by the airline for scheduling and tracking.
* **FlightModel**: The specific model of the aircraft used for this flight, which provides important information for maintenance, scheduling, and capacity planning.

**Flight availability:**

**Static Model of Flight Information**

**Table**: **Flight Availability**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| FlightID | Integer / UUID | Unique identifier for each flight. |
| AirlineID | Integer / UUID | Identifier for the airline operating the flight. |
| FlightNumber | Varchar(20) | The flight number as assigned by the airline. |
| FlightModel | Varchar(50) | The model of the aircraft used for this flight. |

**Example Record**:

**FlightID**: 456

**AirlineID**: 101 (Assuming this refers to "Global Air" from the previous example)

**FlightNumber**: "GA123"

**FlightModel**: "Boeing 787 Dreamliner"

**Explanation:**

**FlightID**: A unique identifier (such as a sequential integer or a UUID) for each flight record in the database.

**AirlineID**: A reference to the airline operating the flight, linking to the airline's unique identifier in the Airlines table.

**FlightNumber**: The official flight number, a combination of letters and numbers, used by the airline for scheduling and tracking.

**FlightModel**: The specific model of the aircraft used for this flight, which provides important information for maintenance, scheduling, and capacity planning.

**Airline routes:**

**Static Model of Airline Routes**

**Table**: **Airline Routes**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| RouteID | Integer / UUID | Unique identifier for each route. |
| SourceAirport | Varchar(50) | The IATA/ICAO code or name of the source airport. |
| DestinationAirport | Varchar(50) | The IATA/ICAO code or name of the destination airport. |

**Example Record**:

**RouteID**: 789

**SourceAirport**: "LHR" (London Heathrow)

**DestinationAirport**: "JFK" (New York JFK)

**Explanation:**

**RouteID**: A unique identifier for each route record in the database. This can be a sequential integer or a UUID.

**SourceAirport**: The departure point of the route, typically indicated by its IATA (e.g., LHR for London Heathrow) or ICAO (e.g., EGLL for London Heathrow) code. The source airport can also be described by its full name.

**DestinationAirport**: The arrival point of the route, identified in the same format as the SourceAirport.

**5.3. Database design**

**Entities Present:**

1. Airlines
2. Flights
3. Users
4. Flight availability
5. Airline\_routes

**Relationships between entities:**

1. An airport manages multiple flights. An airport may or may not have a flight landing/takeoff. Hence, it’s optional. Whereas for flight details, it is mandatory to have one airport association.

**Key Design:**

1. Each airline can have multiple flight details associated with its respective source and destination.
2. There can be multiple flights flying for a particular source and destination.

**5.5. Traceability from requirements to detailed design models.**

**Requirement Traceability Matrix for User Authentication System**

| **Requirement ID** | **Requirement Description** | **Design Component** | **Test Case ID** |
| --- | --- | --- | --- |
| REQ-01 | User registration with secure password. | User Registration Module | TC-01, TC-02 |
| REQ-02 | User login with ID and password. | Login Interface | TC-03, TC-04 |
| REQ-03 | Role-based access control (RBAC). | Role in database column | TC-09, TC-10 |

**6.0 Test Management**

**6.1. A complete list of system test cases**

**Admin part:**

Test name: Login as an admin.

In the database, we have a person that is registered as admin so when they login using their credentials, it will take them to admin page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 1 | Name  Email address  Password  Confirm password | Click register and enter name, email id, password, confirm password and click register | Can see your name on the upper right corner to confirm an admin has been registered successfully | Can see your name on the upper right corner to confirm an admin has been registered successfully |

1. Test name: Create a new flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 2 | Flight number  Eg. UN345  Flight model  Eg. Airbus-350  Airline name  Eg. United | Click the create button and enter a flight number, model and airline name and click submit | Flight saved successfully | Flight saved successfully |

1. Test name: Read flight information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 3 | Credentials to login to a portal | Enter email id and password to see the list of flights | Can see the list of flights on flights web page | Can see the list of flights on flights web page |

1. Test name: Update(edit) a flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 4 | Flight number  Flight model  Airline name | Click edit and provide flight number, model and airline name and click update | Redirect to the same page and can see the updated flight record | Redirect to the same page and can see the updated flight record |

1. Test name: Delete a flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 5 | Select a flight and click the delete button | Select a flight and click the delete button | Redirect to the same page and see the message that says “Successfully deleted the flight” | Redirect to the same page and see the message that says “Successfully deleted the flight” |

1. Test name : Create a new airline.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 6 | Airline name and airline email id | Click the create button on the airlines web page and enter an airline name and email id and click submit | Redirect to the same page with a new airline and see the message that says “Airline created successfully” | Redirect to the same page with a new airline and see the message that says “Airline created successfully” |

1. Test name : Read airline information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 7 | Login using email id and password on the airlines page | Login using email id and password on the airlines page to see the list. | Can see the airline list | Redirect to the same page with a new airline and see the message that says “Airline created successfully” |

1. Test name : Update(edit) airlines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 8 | Airline name and airline email id | Click on the edit button on the airlines page and update the necessary item and click update | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” |

1. Test name: Delete airlines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 9 | Click the delete button on the airlines page | Click on the delete associated with button on the airlines page and update the necessary item and click update | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” |

**User part:**

1. Test name: Search for available flights

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 10 | Departure and arrival airports, Departure, and arrival dates | On home page enter the necessary details and click the submit button | Can see the list of flights available for that date and route from the database. |

1. Test name: Book available flights.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 11 | Click book button | On home page enter the necessary details and click the submit button | It will prompt the user to login if they are not logged in else they are directed to fill the user details and select the payment method. |

1. Test name: Continue as guest.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 12 | Fill in the details.  e.g., Names , and email addresses, of the people for whom a user wants to book a flight for. Additional details such as payment method, phone number and home address to book a flight | Click on continue as guest button to fill in the details and click continue to payment | Redirect to card details page and display the card details to be filled. | Redirect to card details page and display the card details to be filled. |

1. Test name: Invalid credentials.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 13 | Wrong email id and/or wrong password | Enter wrong email id and/or password | Should see the message that says,  “Invalid email id or password” | see the message that says,  “Invalid email id or password” |

1. Test name: user login while booking.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 14 | Email id and password to login to book a flight | Click on the book button associated with each flight to proceed. | Login page should appear for users to login. |

1. Test name: Confirmation message.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 15 | Card number, holder’s name, cvv, expiration date | Click on “Book Ticket” after filling in card details. | Should see the message that says,  “Thank you for booking with us” | See the message that says,  “Thank you for booking with us” |

**6.2 Traceability of test cases to use cases.**

**Use Case UC-01: User Login**

* Description: Customers can log into their personal accounts to manage their flight bookings and preferences.
* Test Cases:
* TC-01: Verifies that the login functionality works with valid credentials.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 01 | User’s correct email id eg. [shivani@gmail.com](mailto:shivani@gmail.com)  And password | Click on “submit” after filling in correct details. | Direct users to fill in the details if they clicked on book button. If they login on home page, direct them to search flights page. | Users are directed to user details page if they click on “book” button otherwise to search flight page. |
|  |  |  |  |  |

* TC-02: Verifies that the login functionality should not work with invalid credentials.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 02 | User’s incorrect email id eg. [shivani@gmail.com](mailto:shivani@gmail.com)  And/or password | Click on “submit” after filling in incorrect details. | Should display the error message that says, “these credentials do not match our records.” | Is displaying the error message that says, “these credentials do not match our records.” |

TC-03: user login while booking.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 03 | Email id and password to login to book a flight | Click on the book button associated with each flight to proceed. | Login page should appear for users to login. |

**Use Case UC-02: Book a Flight**

**Test cases:**

**TC-0**4: Search for available flights

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 04 | Departure and arrival airports, Departure, and arrival dates | On home page enter the necessary details and click the submit button | Can see the list of flights available for that date and route from the database. |

TC-05: Book available flight.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output |
| 05 | Click book button | On home page enter the necessary details and click the submit button | Can see the list of flights available for that date and route |

**Use case UC-03: System Administration for airline and flight CRUD operations.**

**Test cases:**

**TC-06: Admin login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 1 | Name  Email address  Password  Confirm password | Click register and enter name, email id, password, confirm password and click register | Can see your name on the upper right corner to confirm an admin has been registered successfully | Can see your name on the upper right corner to confirm an admin has been registered successfully |

TC-07 : Create a new flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 2 | Flight number  Eg. UN345  Flight model  Eg. Airbus-350  Airline name  Eg. United | Click the create button and enter a flight number, model and airline name and click submit | Flight saved successfully | Flight saved successfully |

TC-08: Read flight information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 3 | Credentials to login to a portal | Enter email id and password to see the list of flights | Can see the list of flights on flights web page | Can see the list of flights on flights web page |

TC-09: Update(edit) a flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 4 | Flight number  Flight model  Airline name | Click edit and provide flight number, model and airline name and click update | Redirect to the same page and can see the updated flight record | Redirect to the same page and can see the updated flight record |

TC-10: Delete a flight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 5 | Select a flight and click the delete button | Select a flight and click the delete button | Redirect to the same page and see the message that says “Successfully deleted the flight” | Redirect to the same page and see the message that says “Successfully deleted the flight” |

TC-11: Create a new airline.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 6 | Airline name and airline email id | Click the create button on the airlines web page and enter an airline name and email id and click submit | Redirect to the same page with a new airline and see the message that says “Airline created successfully” | Redirect to the same page with a new airline and see the message that says “Airline created successfully” |

TC-12: Read airline information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 7 | Login using email id and password on the airlines page | Login using email id and password on the airlines page to see the list. | Can see the airline list | Redirect to the same page with a new airline and see the message that says “Airline created successfully” |

TC-13: Update(edit) airlines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 8 | Airline name and airline email id | Click on the edit button on the airlines page and update the necessary item and click update | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” |

TC-14: Delete airlines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Input | Description | Expected Output | Actual Result |
| 9 | Click the delete button on the airlines page | Click on the delete associated with button on the airlines page and update the necessary item and click update | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” | Can see the airline list with the updated record and display a message that says “ Airline updated successfully” |

**6.3. Techniques used for test case generation.**

**Use Case Testing:**

* We have used use case testing techniques in which test cases are generated based on use cases that define system interactions from the user's perspective. Each use case typically has a happy path (expected scenario) and alternative paths (exception or error conditions).

**6.4 Test Results and Assessments**

**Overview**

Our testing strategy for the airline database management system has been rigorously centered around use case testing to ensure the functionality aligns precisely with user expectations and business requirements. This section evaluates the effectiveness of our test cases and the quality of our software based on the test outcomes.

**Test Case Effectiveness**

The test cases derived from use case scenarios have been exceptionally comprehensive, covering a wide range of user interactions with the system. Each use case was dissected into its constituent parts, allowing for detailed test scenarios that methodically assessed both standard operations and edge cases.

**Software Quality Insights**

Throughour rigorous testing efforts, the system has demonstrated high reliability and robustness. We observed the following key outcomes:

**Functionality**: The system met all the functional requirements as outlined in the use cases. All core features, such as flight booking, user and admin login and authentication, payment information, payment methods, flights list operated within the expected parameters.

**Usability:** Our website is extremely easy to use.

**Performance:** Our web app is highly responsive with response times well within acceptable limits.

**Assessment of Test Cases:**

Our test cases have proven to be highly effective in simulating real-world usage and detecting potential issues before deployment. The granularity of our use case-derived tests allowed for targeted assessments, resulting in a well-vetted system. The iterative refinement of test cases post-initial test runs led to increased coverage and a deeper understanding of system behavior.

**Software Assessment**

Overall, the software's current state reflects a mature and user-oriented system. The rigorous testing and subsequent improvements have culminated in a system that not only meets the specified requirements but also provides a stable, secure, and efficient environment for end-users and administrators alike.

**6.5. Defect reports.**

**Defect Categorization**

Defects have been categorized into the following severity levels:

* **Critical**: Defects that cause system outages or significant malfunctions, leading to a halt in operations or data integrity issues.
* **High**: Defects that significantly impair functionality or performance but do not cause complete system failure.
* **Medium**: Defects that affect non-critical system functionality and moderately impact user experience or performance.
* **Low**: Minor defects that have a negligible impact on system performance and user experience.

**Critical Defects**:

* A concurrency issue in the flight booking module that could lead to overbooking.
* An authentication bypass in the admin module that allowed access to sensitive data.
* We have not yet implemented validation of card details.

**Medium Severity Defects**

* Occasional slow response times

**Low Severity Defects**

* Typos in the admin panel.

1. **Conclusion**

**7.1 Outcomes of the project**

The user was able to successfully book airline tickets and a user-friendly platform was created. Admin was able to perform the CRUD operations. Therefore, all the initial plan of our project was successfully executed and obtained the required outputs.

**7.2 Lesson Learned**

Gained knowledge about database, web page development both from front end and backend.

**7.3 Future development**

Sending mails to users after flight booking. More navigation options

**References:**

1. Allan Leake, (2000), Definition of Database,

<https://searchsqslserver.techtarget.com/definition/database>

1. W3 schools, <https://www.w3schools.com/sql/>
2. Stack Overflow, <https://stackoverflow.com/>
3. Textbook: “Database system concepts” by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
4. Textbook: Ramez Elmasri and Shamkant B. Navathe, ‘Fundamentals of Database Systems’ Addison Wesley Publishing Company