Airline Ticket System Project

Freda Mensah, Kundwa Mushumba, and Clinton Watson

Informatics, Indiana University

System Implementation Info – C451

Jafrina Jabin

May 4th, 2024

Table of Contents

[Customer Problem Statement And System Requirements 3](#_Toc165588172)

[Problem Statement 3](#_Toc165588173)

[Glossary of Terms 3](#_Toc165588174)

[System Requirements 4](#_Toc165588175)

[Functional Requirements 4](#_Toc165588176)

[Nonfunction Requirements 5](#_Toc165588177)

[Functional Requirements Specification 5](#_Toc165588178)

[**Stakeholders** 5](#_Toc165588179)

[**Actors** 5](#_Toc165588180)

[**Use Cases** 8](#_Toc165588181)

[**Use Cases** 10](#_Toc165588182)

[Class Diagram 11](#_Toc165588183)

[Activity And Sequence Diagrams 11](#_Toc165588184)

[USER INTERFACE SPECIFICATION 17](#_Toc165588185)

[**Traceability Matrix** 23](#_Toc165588186)

[**Traceability Matrix** 27](#_Toc165588187)

[**System Architecture and System Design** 28](#_Toc165588188)

[**Identifying Subsystems** 28](#_Toc165588189)

[**Mapping Subsystems to Hardware:** 29](#_Toc165588190)

[**Data Storage:** 29](#_Toc165588191)

[**Network Protocol:** 29](#_Toc165588192)

[**Global Control Flow** 29](#_Toc165588193)

[**Hardware Requirements** 30](#_Toc165588194)

[**Algorithms** 30](#_Toc165588195)

[**Search algorithms** 30](#_Toc165588196)

[**A booking algorithm** 31](#_Toc165588197)

[**Payment Processing Algorithm** 32](#_Toc165588198)

[**Data Structure** 32](#_Toc165588199)

[**USER INTERFACE DESIGN AND IMPLEMENTATION** 33](#_Toc165588200)

[**Homepage Enhancement:** 33](#_Toc165588201)

[**Streamlined Ticket Purchase Without Mandatory Sign-In:** 35](#_Toc165588202)

[**Sign-In and Sign-Up Enhancement:** 37](#_Toc165588203)

[**Design of Tests** 38](#_Toc165588204)

[**Unit Testing** 38](#_Toc165588205)

[**User Registration Testing** 38](#_Toc165588206)

[**Ticket Booking Testing:** 38](#_Toc165588207)

[**Authentication Testing:** 38](#_Toc165588208)

[**Integration Testing** 39](#_Toc165588209)

[**Non-functional Testing** 39](#_Toc165588210)

[**Project Plan** 40](#_Toc165588211)

[**References** 40](#_Toc165588212)

# Customer Problem Statement And System Requirements

## Problem Statement

The current state of airline ticketing systems presents significant challenges, ranging from user interface deficiencies and inconvenient booking processes to biased airline price comparisons. Additionally, these systems lack comprehensive information about destination cities, including suggestions for activities and cost comparisons as well as the ability of users to add reviews regarding their experience. Consequently, travelers are often left unprepared and frustrated, whether embarking on a vacation or a swift business trip.

## Glossary of Terms

**Seat Availability:** Real-time information on a flight’s available seats

**Flexible Booking System:** Allows for one-way, round-trip, and multi-city reservations with customizable travel dates.

**Destination Information:** Detailed information about a destination’s popular sights and vacation places.

**System Requirements:** Users are required to have a stable internet connection, a web browser (Safari, Chrome, Microsoft Edge, or Firefox), and a PC or mobile phone.

**Flight Search:** The procedure inside the airline ticketing system by which consumers may locate and get information about available flights based on a variety of criteria such as departure and arrival locations, dates, and preferred carriers.

**Price Comparisons:** Users may compare ticket prices from multiple airlines for certain routes and periods, allowing them to make better-educated decisions.

## System Requirements

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **No** | **Priority Weight** | **Description** |
| Req 1 | High | Flight reservation |
| Req 2 | High | Show flight comparisons |
| Req 3 | High | Notify passengers of leisure activities in their destination city based on previous travels |
| Req 4 | High | Create an account for each user |
| Req 5 | High | Searching for flights |
| Req 6 | High | Notify Passengers for any flight changes (delays , cancellations, or other changes ) |
| Req 7 | Medium | Suggest activities in other cities and include cost of living |

### Nonfunction Requirements

**Functionality**

* Secure flight checkout.
* At least 3 payment options.
* Have at least 3 airline price comparisons.
* System able to scale in case of high user traffic.
* Secure payment gateways in multiple currencies and various methods.

**Usability**

* The UI should be user-friendly and easy to use.
* Account creation and login processes should be straightforward.

**Reliability**

* The database must be backed up to prevent data loss.
* The system should be highly available with little downtime.

**Performance**

* Loading should take less than 5 seconds.
* The system should manage many users efficiently.

**Supportability**

* The system should be easy to maintain and upgrade.
* Sufficient user help and guidance should be available for users.

# Functional Requirements Specification

## **Stakeholders**

* Travelers
* Managers/Supervisors
* Administrators
* Investors
* Customer Support Rep
* Airline Reps
* Marketing Specialists

## **Actors**

* **Primary Actors**
* **Customer**: searches and books flights for personal or business travel. This actor also views real-time details of available flights, adds reviews and provides feedback on airline experiences.
* **Administrator**: This actor manages system components and configurations. They also add, modify, or remove flight information. The actor adjusts pricing rates and manages user accounts.
* **Customer Support Rep**: This actor assists users with queries and issues related to the booking process, providing guidance on system navigation and functionalities, and actively collecting user feedback to forward relevant information to administrators.
* **Secondary Actors**
* **System**: This actor assigns seats to customers based on preferences and availability. The actor also provides comprehensive information about destination cities and manages and facilitates the booking process.
* **Airline Reps:** This actor is responsible for managing and updating airline-specific information, coordinating with the system to ensure accurate flight details, and actively receiving and responding to customer reviews and feedback.
* **Database:** This actor is responsible for storing and retrieving user information securely. They facilitate the storage and retrieval of customer reviews.
* **Payment processor:** This actor validates the form of payment provided by the customer and accepts of declines

### **Use Cases**

* **Customer (14):**
* Search for flights:  Customer will look for different flights and select preferred flight (2)
* Search city: To view activities that can be done in the city (2)
* Reserve flights: To book a flight (2)
* View flight details: To view reserved flights (1)
* Add reviews: To add review (3)
* Modify/delete reviews: To modify or delete reviews (1)
* Add text in chatbot: to add text when requesting support (4)

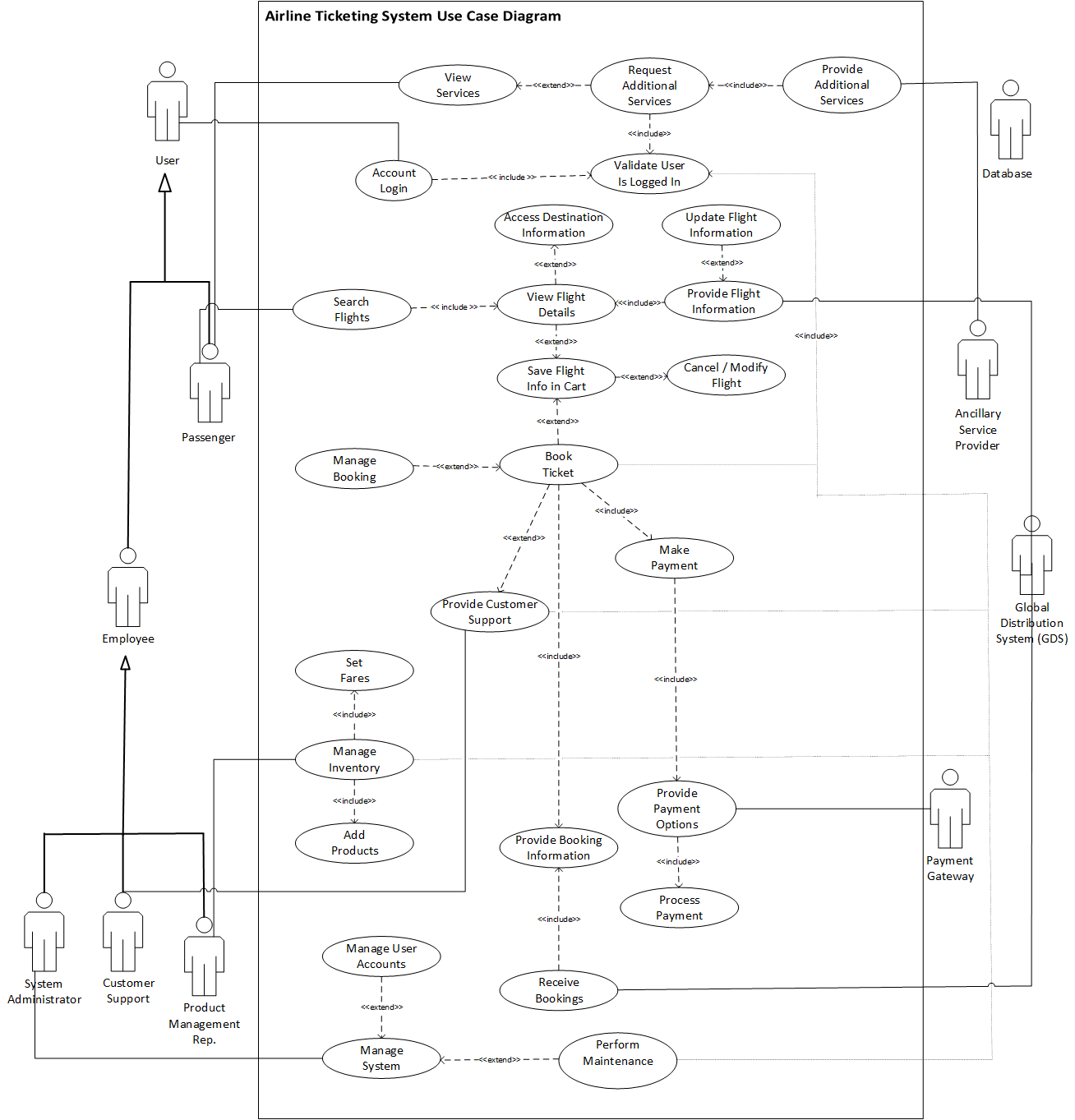
* **Administrator (7):**
* Search for flights: To search for available flights (2)
* Modify/ cancel flights: To modify or cancel flights (2)
* Add/remove user: To remove or add user information (3)

* **Customer support rep (4):**
* Search flights: to search for customer flights (2)
* Add text in chatbox: to reply to customers (2)
* **System (13):**
* Validate user login: To perform checks when user logs in (3)
* Add user: To add users to the system (3)
* Show flights: To show available flights (4)
* Show city information: To show travel city information and activities (3)

* **Database (5):**
* Retrieve data: To retrieve customer information (1)
* Store data: To store customer data (2)
* Manage access: To manage who can access the database (2)

* **Payment processor (6):**
* Validate payment: validate if information added is correct (4)
* Process payment: process correct payment (2)

### **Use Cases**



Class DiagramA diagram of a computer

Description automatically generated with medium confidence

# Activity And Sequence Diagrams

A diagram of a company

Description automatically generated

A diagram of a program

Description automatically generated

A diagram of a system

Description automatically generated

A diagram of a company

Description automatically generated

A diagram of a diagram

Description automatically generated

# USER INTERFACE SPECIFICATION

A group of people with luggage

Description automatically generated

The user account creation feature is prominently displayed on the front page, eliminating the need for users to navigate elsewhere. To create an account, users simply enter their first name, last name, and email and click the sign-up button. Upon submission, the email address undergoes verification, and users receive an on-screen notification prompting them to verify their email.

**Navigation:** Front page>Create Account > Account creation form > enter first name, last name, and email address > sign up button. Check on-screen notification for email verification.

User Effort Estimation: Interacting with “Create account” feature (1 click). Enter first name, last name, and email address(Keystrokes varies with user input). Click on “sign up” button(1 click).

#**Clicks:** 2 + Keystrokes (varies with user input)

A screenshot of a website

Description automatically generated

After the user successfully registers for an account or returns to the homepage, the user clicks on the “sign in” button.

The system then displays a login form with a field for username/email and password requesting for input from user.

The user then enters their username/email and password used to create the account.

If the user leaves one field empty or enters incorrect information, the system displays an error message: “Error: Account or username is incorrect. The user then clicks on the “Login” button. If the login is successful, the system directs the user to their account dashboard.

Navigation: Homepage>Sign In button>Login form>correct errors>login button>(if successful)user dashboard

User Effort Estimation: Click on “sign in” button (1 click). Enter username/email and password(keystrokes varies with user input). Click on “login” button (1 click).

Total clicks: 2 clicks + Keystrokes (varies with user input)

A screenshot of a chat

Description automatically generated

A "Support" button is accessible via the navigation links. When users click this button, a page for chatting with support will open. Users can then enter their inquiries and engage in real-time conversation with support staff.

**Navigation :** Navigation Bar > support button. Which takes you to the Chat Modal or the Support Chat

User Effort Estimation: Click on “support” button (1 click). Engage in conversation with customer care or customer support bot (Keystrokes varies with user input).

#**Clicks :** 1 + Keystrokes (varies with user input)

A screenshot of a computer

Description automatically generated

After logging in, the user clicks on the “Book your flight” option

The system displays a booking form with fields for “Enter origin” and “Enter Destination” where the user enters the origin and destination cities.

If the user leaves any of the fields empty, the system displays an error message: “Error: insert text in missing field”.

The user clicks on the “search” button after inserting text.

The system then shows search results, including a list of available flights, prices, and airline options.

Navigation: User Dashboard>Book Your flight option>Booking form>Enter origin and destination>search button. Search results with flight options and prices are displayed.

User Effort Estimation: Click on “book you flight” option (1 click). Enter origin and destination cities(3 keystrokes). Click on “search” button (1 click).

Total clicks: 2 clicks + 3 Keystrokes.

A screenshot of a credit card

Description automatically generated

After selecting their flights, users will proceed to enter their payment information, either debit card or PayPal. For debit card payments, users must input their name, card number, expiration date, security code, and billing ZIP code. If all information is correct, users will receive a notification confirming payment acceptance. If there are errors, users will be prompted to correct the information before proceeding.

**Navigation:** Search results > Select flights > Proceed to payment button > Payment information form > Select debit card > Enter card details > Submit Payment button. System sends a notification of payment acceptance.

User Effort Estimation: Click on “proceed to payment” button (1 click). Select payment option (1 click). Enter name, card number, expiration date, security code, and billing ZIP code(Keystrokes varies with user input). Click on “submit payment” button (1 click).

#**Clicks :** 3 + Keystrokes(varies with user input)

# **Traceability Matrix**

|  |  |  |
| --- | --- | --- |
| **No.** | **Priority Weight** | **Description** |
| REQ 1 | 5 | User authentication and authorization |
| REQ 2 | 2 | Search and display available flights |
| REQ 3 | 5 | Admin and Customer supports can log in/out from the system |
| REQ 4 | 4 | Booking and reservation functionality |
| REQ 5 | 2 | Payment processing and integration |
| REQ 6 | 1 | Notification system for booking confirmation |
| REQ 7 | 2 | Seat selection and customization |
| REQ 8 | 1 | Integration with third-party APIs for flight data |
| REQ 9 | 1 | Multi-language support for the user interface. |

|  |  |
| --- | --- |
| **No** | **Description** |
| UC1 | Passengers can **search flights** to see and choose from what is available based on desired destination |
| UC2 | Passengers can **view services** to see options available for additional checked luggage, select preferred seat, or access admirals club at various airports |
| UC3 | **Account login** allows users access to the system. This applies particularly to the instances where employees need to carry out their various job functions, and for passengers to secure products and services beyond searches |
| UC4 | **User logon validation** ensures users are logged into the system |
| UC5 | **Request additional services** allow users to select and pay for these services |
| UC6 | **View Flight Details** provide passengers with the details of flights that may wish to book |
| UC7 | **Access destination information** provide users with available amenities and activities available in the various cities |
| UC8 | **Cancel/Modify flights** provide passengers with options to change the information on flights saved in their cart |
| UC9 | **Save flight info in cart** allow the passenger the options of booking, modifying, or canceling the flight at a later |
| UC10 | **Book ticket** require passengers to pay for their tickets based on the flight info saved in cart. The booking process can also be done by the Customer Support Agent |
| UC11 | **Make payment** provide passengers with the available options for payment |
| UC12 | **Provide payment options** is the acceptable forms of payment the gateway provides passengers |
| UC13 | **Process payment** is the payment gateway’s verification and processing of the passenger’s payment information |
| UC14 | **Manage booking** allow the passenger or the customer support agent to make modification to tickets that have been booked |
| UC15 | **Provide customer support** is the general function of the agent to assist passengers with booking or managing previous booking |
| UC16 | **Receive bookings** is the Global Distribution System’s responsibility to record all booking information provided by the airline |
| UC17 | **Provide flight information** is the information made available to passengers by the Global Distribution System (GDS) |
| UC18 | The GDS also has the responsibility to u**pdate flight information** |
| UC19 | **Provide booking information** is the details of bookings supplied to the GDS by the airline |
| UC20 | **Manage inventory** is the Product Management Rep’s responsibility for setting fare and adding products |
| UC21 | **Manage System** is the System Administrator’s responsibility for the maintenance of the system, and user accounts |
| UC22 | **Provide additional services** is the availability of seat selection, additional checked luggage, and admirals club access by Ancillary Service Provider |

## **Traceability Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req't | PW | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | UC7 | UC8 | UC9 | UC10 | UC11 | UC12 | UC13 |
| REQ1 |  |  |  | X | X |  |  |  |  |  | X |  | X | X |
| REQ2 |  | X |  |  |  |  | X |  |  |  |  |  |  |  |
| REQ3 |  |  |  | X |  |  |  |  |  |  |  |  |  |  |
| REQ 4 |  |  |  |  |  | X |  |  |  | X | x |  | X | X |
| REQ 5 |  |  |  |  |  | X |  |  |  |  | x | X | X | x |
| REQ 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 7 |  |  | X |  |  |  |  |  | X |  |  |  |  |  |
| REQ 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Max PW | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Total PW | | 5 | 5 | 5 | 5 | 9 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

# **System Architecture and System Design**

**Architectural Styles:** The client-server architectural approach was utilized to build the airline ticketing system. In this type, a client, which is the user’s computer or device, communicates with a server to do different functions such as searching for flights, purchasing tickets, and retrieving flight and destination information. The client does not need to understand the server’s internal workings, and it communicates with it simply by making requests and getting replies. We used WAMP (localhost) to run the program on a local webserver on our computers.

**Identifying Subsystems:**

*User Interface (UI):* In charge of displaying information to the user and receiving user input. It provides features for creating an account, logging in, searching for flights, booking, and chatting with support.

*Backend Services:* Manages business logic and interactions with other systems like payment gateways and airline databases. It provides components for comparing flights, sending notifications, managing users, and integrating with external APIs.

*Database Management System (DBMS):* Controls the storage and retrieval of data about users, flights, reservations, and destination information. It has components for data storage, retrieval, and management.

**Mapping Subsystems to Hardware:** The system will be executed on a single computer, usually a web server, that houses both the client and server-side components. There is no dispersal of components among numerous computers.

**Data Storage:** Data persistence beyond a single execution is required by the system to maintain user information such as user account, flight details, reservations, and destinations. Persistent items contain user profiles, flight records, booking information, and destination data. The storage management technique entails effectively storing structured data and ensuring data integrity using a relational database management system like Postgresql.

**Network Protocol:** HTTP (Hypertext Transfer Protocol) is used to communicate over the Internet. HTTP is a popular protocol for communicating between clients and servers in web-based applications. It permits data interchange between the client’s web browser and the server that hosts the airline ticketing system.

## **Global Control Flow**

* *Execution Order:* The system is event-driven, where user actions such as searching for a flight, booking a flight, and interactions with user interface trigger events that drive the system’s flow.
* *Time Dependencies:* The system does not have strict real-time constraints. However, it may utilize timers for periodic tasks such as checking if a flight is available, updating, and sending notifications to users.
* The system may use multiple threads to handle simultaneous user requests and background tasks. Synchronization mechanisms such as locks and signals may be employed to manage shared resources and ensure data consistency.

## **Hardware Requirements**

* Users require a device with a screen display to interact with the user interface.
* A stable internet connection is necessary for users to access the system and communicate with the server.
* The system requires a web server with sufficient processing power, memory, and storage to host the application and manage user requests efficiently.

# **Algorithms**

**Search algorithms** will be implemented to allow passengers to find available flights based on their desired destination, travel dates, and other preferences. This algorithm might be a simple or more complex algorithm based on how flights will be sorted based on the criteria.

A diagram of a flowchart

Description automatically generated

**A booking algorithm** will be implemented when a passenger decides to book a flight. This algorithm involves checking seat availability, reserving the seat, and updating the database with the information.

A diagram of a flowchart

Description automatically generated

**Payment Processing Algorithm** will be implemented to process payments for the passenger, and it handles the secure processing of payment transactions, verifying payment details and updating booking status.

# **Data Structure**

There are arrays and objects/JSON used in the airline ticket system. There are arrays to store lists of cities for both origin and destination options in the select cities form. These arrays hold the available options for passengers to choose the cities they are originating from and the cities they are originating to.

The objects/JSON are there to store information about users, flights, bookings, etc., in a structured form. They allow for easy manipulation and access to data through methods and properties.

When it comes to performance arrays and JSON offer fast access to data index and lookup making them suitable for access to data and elements.

Arrays are memory-efficient for storing homogeneous data types in contiguous memory blocks. Objects consume memory proportional to the size of the data they store.

JSON provides high flexibility as it allows for the dynamic addition, removal, and modification of properties. Arrays provide moderate flexibility for adding, removing, and modifying elements.

# **USER INTERFACE DESIGN AND IMPLEMENTATION**

After creating the initial screen mock-ups in Module 7, we revisited the designs and realized it was tacky and not what most flight ticket reservation applications, systems, or websites are like. In the screen mock-ups that were created in Module 7, we used the Canva designing platform. However, for the main implementation, we utilized Express.js for the front-end and Node.js for the back-end.  The comparison helped us identify areas for improvement regarding user engagement and interaction flow. Below are some of the key changes we made and how they impact the user experience:

**Homepage Enhancement:** Initially, our homepage had a brief introduction to our system, inviting users to register via an embedded link. Unlike standard ticketing systems, which provide quick flight search capabilities, our homepage merely displayed an introductory paragraph. To improve user engagement, we removed unnecessary content and added new forms. Users can now easily search for airplane tickets immediately from the homepage without having to sign in, which simplifies the ticket purchasing process.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Streamlined Ticket Purchase Without Mandatory Sign-In:** To improve user experience and lower barriers to purchasing tickets, we redesigned our system so that customers can explore and pick tickets without having to sign in. This modification allows consumers to freely explore event and ticket alternatives, providing a more smooth and user-friendly experience. At checkout, people can choose to sign in, create an account, or continue as guests. This flexibility enables a variety of user choices, allowing those who desire a rapid transaction to buy tickets without creating an account while still providing account-based benefits to those who wish to join up. By eliminating the requirement for sign-in, we want to improve access and speed the ticket purchase process.

A screenshot of a computer

Description automatically generated

**Sign-In and Sign-Up Enhancement:** In our screen mock-ups and midterm demos, the sign-up and sign-in areas had an out of place design for an airline ticketing system with mismatched colors and a cluttered appearance. To solve this, we redesigned the style and visuals to mimic a professional airline ticketing system, resulting in a more consistent and user-friendly experience. We reduced the forms by deleting unneeded fields and design components, resulting in a more streamlined, user-friendly experience.

A screenshot of a login form

Description automatically generated

A screenshot of a login form

Description automatically generated

# **Design of Tests**

## **Unit Testing**

Unit Testing is important in ensuring that individual modules and components function as expected. The test cases we plan to program and use for the unit testing are below:

**User Registration Testing:** We will test the registration process with valid and invalid input. This should result in successful registration with valid inputs and error messages sent when user enter with invalid inputs.

**Ticket Booking Testing:** We will test the ticket booking process, select seats, create payments, and ticket issuance. This should also result in successful booking if the input is valid and error messages if the input is invalid.

**Authentication Testing:** We will test the sign-in and sign-up process with valid and invalid information which can either be successful if the information is valid or generate error messages if information is invalid.

## **Integration Testing**

The integration testing ensures that different system modules work together as expected. Our strategy is to test by progressively combining system modules to verify their interactions. This will help us to identify issues early and ensure seamless communication between these components.

The first phase is a test between the user authentication and sign in or sign up verifying that sign-in or sign-up was successful and user information is stored in database.

The second phase is a test between booking and payment processing to ensure that booking requests trigger appropriate payment workflows.

The third phase is a test between the user and the support chatbot to ensure that support is indeed of help to the user and there is no issue.

## **Non-functional Testing**

For non-functional testing, we will test the performance, usabilit, and security to ensure it is safe for users.

In performance testing, we will test to see the response time and stability of our system under high loads to see if it will maintain acceptable response times and stability under different conditions.

In Usability testing, we will try to test using real people to evaluate the ease of use of the user interface and gather feedback on the overall experience to see if the system is easy to use, with a positive user experience.

For security testing, we will test the system’s security features, like data protection, and authentication, conducting vulnerability assessments to ensure that system is secure for users.

# **Project Plan**

* Week 1: Define project requirements and assign roles to different team members.
* Week 2: Set up and test the programming environment to figure out the system’s framework and design concepts.
* Week 3: Start the front-end development using Node.js.
* Week 4: Develop the user identification for login and account creation and interface layout.
* Week 5: Initiate the backend development using Java.
* Week 6: Implement UI features for checking ticket prices, booking flights, Viewing, and adding reviews about different airlines.
* Week 7:  Implement MySQL for Database and create systems for complete destination information.
* Week 8: Test features, record, and submit what we have for midterm presentation.
* Week 9: Implement a system for comparing prices.
* Week 10: Debug the application and perform final requirements.
* Week 11: Prepare for system implementation by improving performance, preparing reports and usage manuals, and user readiness testing. Gather feedback from users for potential enhancements.
* Week 12: Review the overall success. Test and record the demo for final presentation.

# **References**