FlyAway (An Airline Booking Portal)

Software Development
Phase 2: Become a Back End Expert

Boigantso Mabena

Sprint Planning:

Sprint 1: Setting up the basics

Duration: 4-6 days

- User Story 1: As a user, I want to be able to enter travel details for the homepage .
- User Story 2: As a user, I want to see a list of available flights with their ticket prices based on the travel details I entered.

Tasks to be performed:

1. Create a homepage with a search form.
2. Implement the form to accept travel details (date of travel, source, destination, number of person(s).)
3. Set up the back end to process the search form and return flight options and prices.
4. Display the flight options and prices to the user.

Sprint 2: User Registration and Payment

Duration: 1 week

- User story 3: As a user, I want to be able to select a flight and book it, which should take me to a registration page.
- User story 4: As a user, I want to see the flight details on the registration page and proceed with payment via a dummy payment gateway.
- User story 5: After a successful payment, I want to see a confirmation page with booking details.

Tasks to be performed:

1. Implement flight selection and registration page.
2. Display flight details on the registration page.
3. Implement a dummy payment gateway.
4. Upon successful payment, display a confirmation page with booking details.

Sprint 3: Admin Backend and Database Integration

Duration: 1 week

- User story 6: As an admin, I want to have an admin login page and the ability to change my password.
- User story 7: As an admin, I want to manage the master lists of places, airlines, and the list of flights.

Tasks to be performed: :

1. Create an admin login page and password change functionality.
2. Develop admin panels to manage the master lists of places, airlines, and flights.
3. Integrate these admin functionalities with the database for data storage.
4. Ensure data security and access control for the admin backend.

These three sprints cover the essential features of the travel booking system, starting with the basic search and progressing to user registration, payment processing, and admin management. Additional sprints can be planned for further enhancements and optimizations.

Core Concepts Used In The Project:

1. Servlet:

A servlet is a Java class that extends the javax.servlet.http.HttpServlet class and is used to handle HTTP requests and produce HTTP responses. Servlets are the building blocks of Java web applications.

2. Request and Response Objects:

Servlets process HTTP requests and generate HTTP responses. These operations are performed through two primary objects:
- HttpServletRequest: Represents the incoming HTTP request from the client, including parameters, headers, and other request data.
- HttpServletResponse: Represents the response that will be sent to the client, allowing you to set response headers, content, and status.

3. Servlet Container:

A servlet container (e.g., Apache Tomcat, Jetty, or WildFly) is a web server or application server that manages the lifecycle of servlets. It handles the creation, initialization, execution, and destruction of servlets.

4. Lifecycle of a Servlet:

A servlet goes through various stages during its lifecycle:
- Initialization: The container loads the servlet and calls its `init` method.
- Request Handling: The container calls the `service` method for each client request.
- Destruction: The container calls the `destroy` method when the servlet is removed or the server is shut down.

5. URL Mapping:

Servlets are typically mapped to specific URLs using a web.xml configuration file or annotations (introduced in Java EE 6). This mapping defines which servlet should handle a particular request.

6. Servlet API:

The Servlet API is a set of classes and interfaces provided by Java EE for creating servlets. Key interfaces include Servlet, HttpServletRequest, HttpServletResponse, and ServletContext.

7. HTTP Methods:

Servlets can handle various HTTP methods, including GET, POST, PUT, DELETE, and others. The `doGet`, `doPost`, `doPut`, and `doDelete` methods are commonly overridden in servlets to handle specific HTTP methods.

8. Session Management:

Servlets can manage user sessions using HttpSession, allowing data to be stored across multiple requests and responses. Sessions are essential for maintaining user state in web applications.

9. Servlet Filters:

Filters are used to intercept and preprocess requests and responses before they reach the servlet. They are handy for tasks like authentication, logging, and data transformation.

10. Exception Handling:

Servlets can handle exceptions gracefully using try-catch blocks and provide custom error pages for different HTTP status codes.

11. Multithreading:

Servlet containers may create multiple threads to handle simultaneous requests. Servlets should be designed to be thread-safe to handle concurrency effectively.

12. Deployment Descriptor:

In older Java EE versions, web.xml was used to configure servlets and their mappings. In modern Java EE versions, annotations can be used to define servlet configurations.

13. Cookies:

Servlets can set and read cookies to maintain information on the client side.

The Algorithm:

Backend:

1. Set up a server (Tomcat Apache 10.1) using a suitable backend technology and create routes to handle requests from the frontend.

2. Implement an admin login page with functionality for the admin to change their password.

3. Implement authentication and session management for admin login.

4. Create a database to store the following:

- Master list of places for source and destination.
- Master list of airlines.
- List of flights with source, destination, airline, and ticket price.

5. Implement API endpoints for the following actions:

- Search for available flights based on user input.
- Handle user registration and store their personal details.
- Handle payment and store booking details.

6. Ensure that the admin can manage the master lists of places and airlines through suitable admin panels.

7. Implement security measures to protect sensitive user data and payment information.

8. On successful payment, generate a confirmation page with booking details and provide an option to print or save the confirmation.

This algorithm provides a high-level overview of the necessary steps to build a web application with the mentioned features. You would need to use specific technologies and frameworks for both frontend and backend development, and you may also need to use a database system to store and retrieve data. The exact implementation details would depend on your chosen technology stack and the programming language you're comfortable with.

Frontend:

1. Create the homepage with a search form containing input fields for date of travel, source, destination, and the number of persons.
2. When the user submits the form, send an AJAX request to the server with the entered details.
3. Display a loading spinner or message to indicate that the search is in progress.
4. Receive and display the available flights with their ticket prices received from the server.

5. Allow the user to select a flight to book by clicking on it.

6. Redirect the user to the registration page.

7. On the registration page, create a form for the user to fill in their personal details, such as name, contact information, and payment details.

8. Once the user submits the registration form, send the user's information to the server.

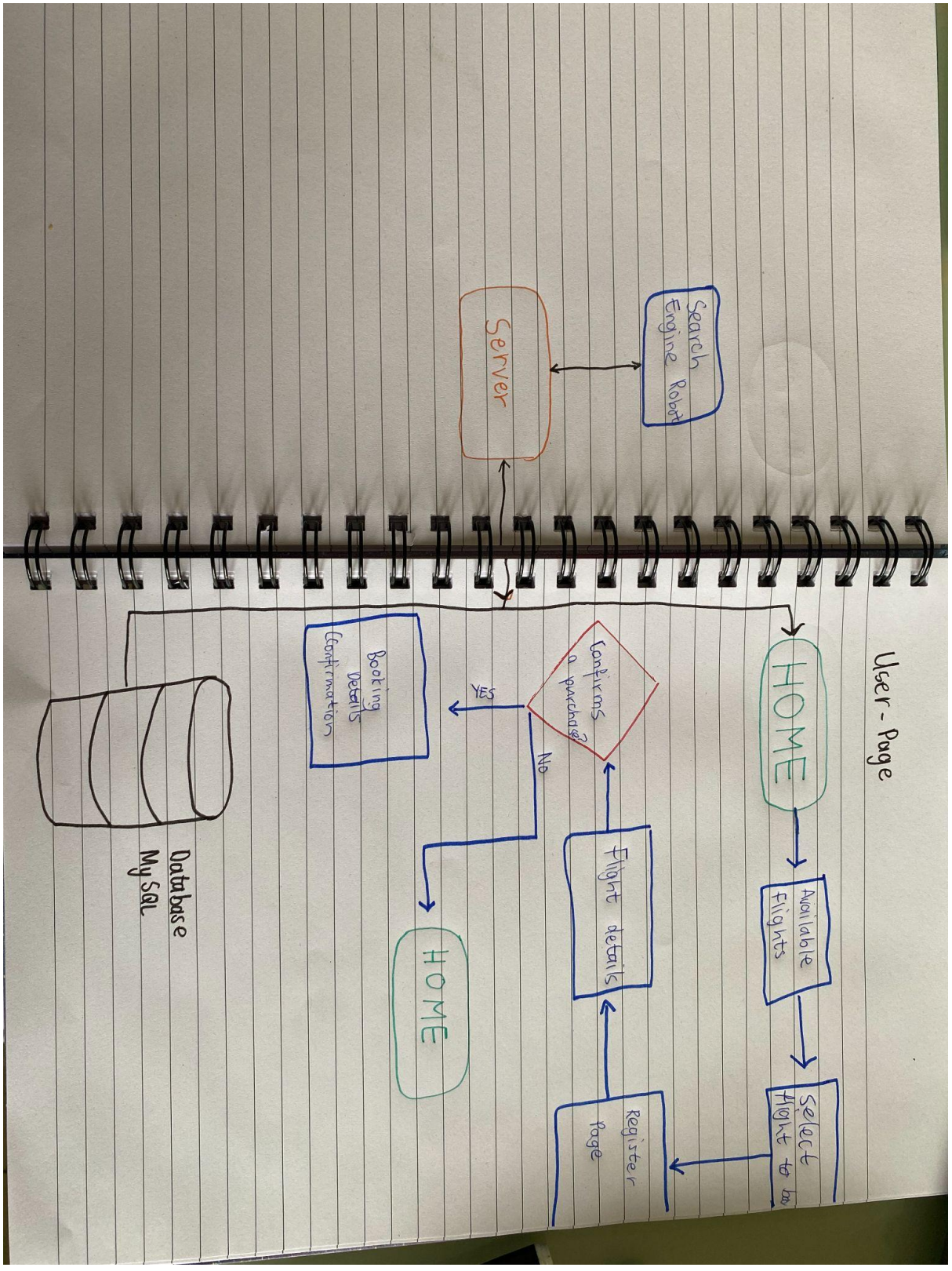
9. Display the flight details (source, destination, airline, ticket price) and the user's personal details for confirmation.

10. Provide a button for the user to proceed with the payment.

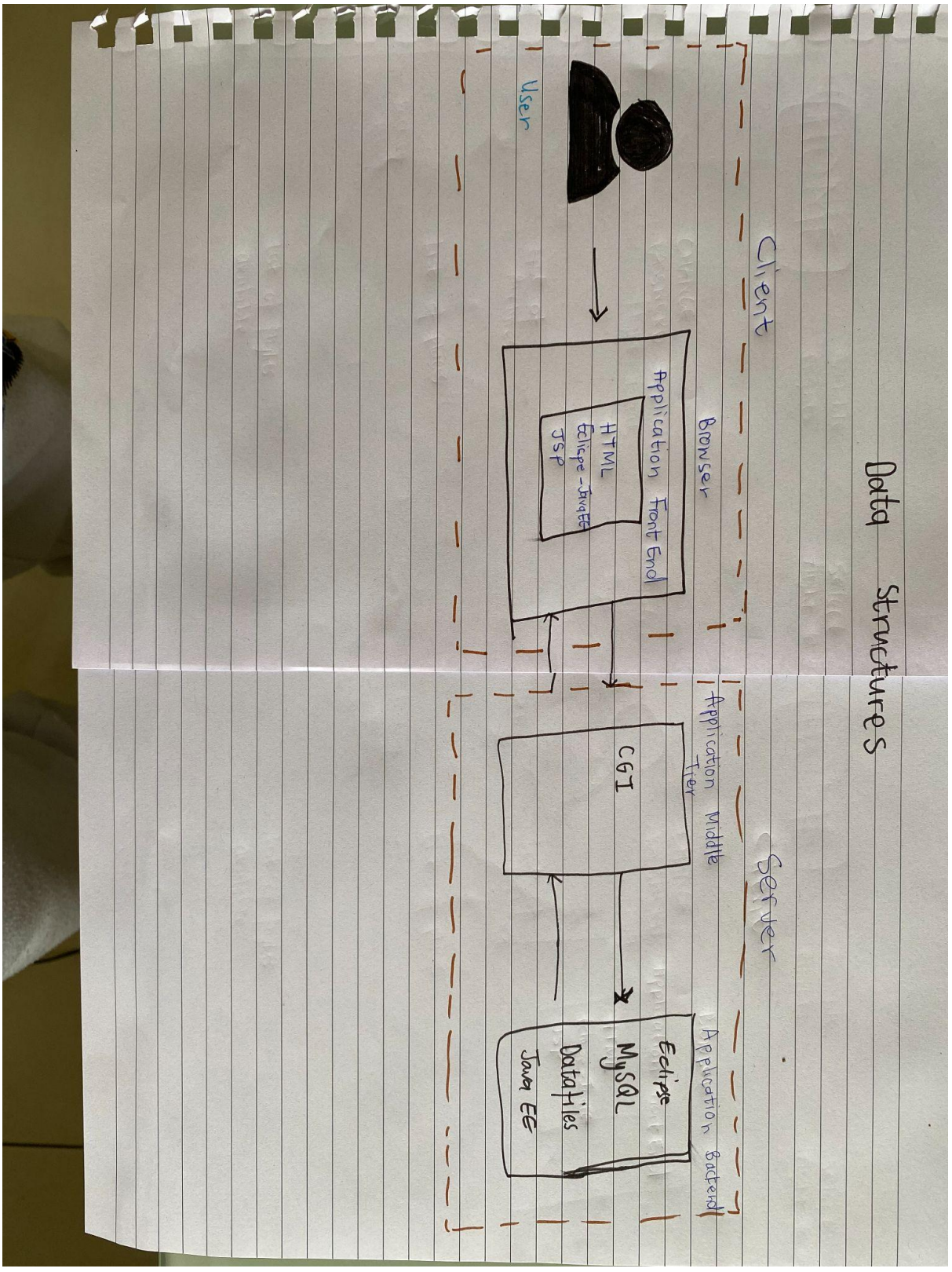
11. Implement a dummy payment gateway where users can make a mock payment.

12. On successful payment, store the booking details in the database, and show a confirmation page with all the booking information.

Flowchart:


User - Page
HOME
Available flights
Select flight to be
Register Page
Flight details
Confirms a purchase?
YES
NO
Booking Details Confirmation
HOME
Server
Search Engine Robot
Database MySQL

Data Structures:



Git Repository:

Boigantso/FlyAway.com: Phase 2 - Become a Back End Expert (github.com)

Source Code:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">

<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,400;0,500;1,600&dis
play=swap" rel="stylesheet">

 <link rel="stylesheet" href="style.css">

<title>Home Page</title>
<link type="/text/css" rel="stylesheet" href="CSS/AdminHome.css">
<style>
body{
margin:0;
background-image: url(CSS/Airline_marketing.jpg);
background-size: cover;
background-repeat: no-repeat;
background-position: center top;
background-size: 1350px 650px;

}
.head{
background-color : Grey;
color : #fff;
height : 75px;
}
h2{
margin: 15px;
            font-family: 'Montserrat', sans-serif;
            text-align : center;
            color: white;
```

```
            font-size: 35px;
}
.btn{
    background: none;
    border:2px solid white;
    font-family: "montserrat", sans-serif;
    text-transform: uppercase;
    padding: 12px 20px;
    margin:10px;
    cursor: pointer;
    transition: color 0.4 linear;
    }


a{
margin : 20px 20px;
padding : 5px;
text-decoration : none;
}

button{
 background-color: white;
    color: black;
    font-size: 16px;
    font-width: bold;
    padding: 10px 15px;
    border: 3px solid grey;
    border-radius: 8px;
    text-align: center;
    position: relative;
    center: 10px;
    bottom: 10px;
    }

</style>

</head>
<body>
<center>
            <div class="head"><h2>FlyAway.com</h2>
            </div>
</center>
<div><h2>
   <%  String email=(String)session.getAttribute("email");

             out.println("Welcome aboard, book now & travel in luxury.");
   %>
</h2></div>
```

```
<hr>
<center>
<div class="topnav">
  <h3><br><a href="changePassword.jsp"style="text-decoration:
none;color:white";>CHANGE PASSWORD</a></br>
  <br><a href="Places.jsp"style="text-decoration: none;color:white";>LIST OF
DESTINATIONS</a></br>
  <br><a href="AirlineList.jsp"style="text-decoration: none;color:white";>LIST OF
AIRLINES</a></br>
 <br> <a href="FlightsList.jsp"style="text-decoration: none;color:white";>LIST OF
FLIGHTS</a> </br>
 <link rel="stylesheet" href="style.css">

 </h3>

</div>
<br>

 <button ><a href="Home.html"  style="text-decoration:none">Login</a></button>
</center>
</body>
</html>
```

Conclusions:

The implementation of a comprehensive travel booking system that includes a search form for travel details, flight selection, user registration, payment processing, and a robust admin backend is a multifaceted and dynamic endeavour. It seamlessly integrates the user's journey from travel planning to payment confirmation.

The user experience begins with a user-friendly homepage where travellers can input their travel details, such as date, source, destination, and the number of passengers. This information is then processed to present a list of available flights and their respective ticket prices, offering a convenient and informative platform for travellers.

Upon selecting a preferred flight, users are seamlessly directed to a registration page, where they are required to provide personal details. Subsequently, they are presented with a detailed summary of their flight booking, ensuring transparency and accuracy.

The payment process is executed through a dummy payment gateway, mimicking a real-world transaction. The secure and efficient handling of payment data assures users of a safe and trustworthy experience.

To manage and maintain this system, the admin backend offers essential functionality. It includes an admin login page with the option to update the password, ensuring security. The admin backend also facilitates the maintenance of a master list of places for source and destination, a master list of airlines, and a list of flights, each with associated details.

So lastly, this travel booking system offers a comprehensive and user-centric solution for travellers, seamlessly guiding them from search to confirmation, while empowering administrators with the necessary tools to manage the system effectively. Its success depends on efficient front-end design, robust back-end functionality, and a focus on user security and satisfaction.