UPSKILL CAMPUS

College name : karpaga vinayaga college of engineering and technology

Travel Planner: Development of a Web-Application

In ComputerScience

By

S.Dharshini

Table of Content

1. Project Overview
2. Features and Functionalities
3. System Architecture
4. Database Design
5. Front-end Development
6. Back-end Development
7. coding
8. Travel planner execution
9. Security and Authentication
10. Testing and Debugging
11. Deployment
12. Conclusion

Project Overview:-

Travel Planner is a full-stack web application designed to help users plan and organize their trips. The application allows users to create an account, search for destinations, plan their itinerary, book flights and hotels, and share their travel plans with friends and family.

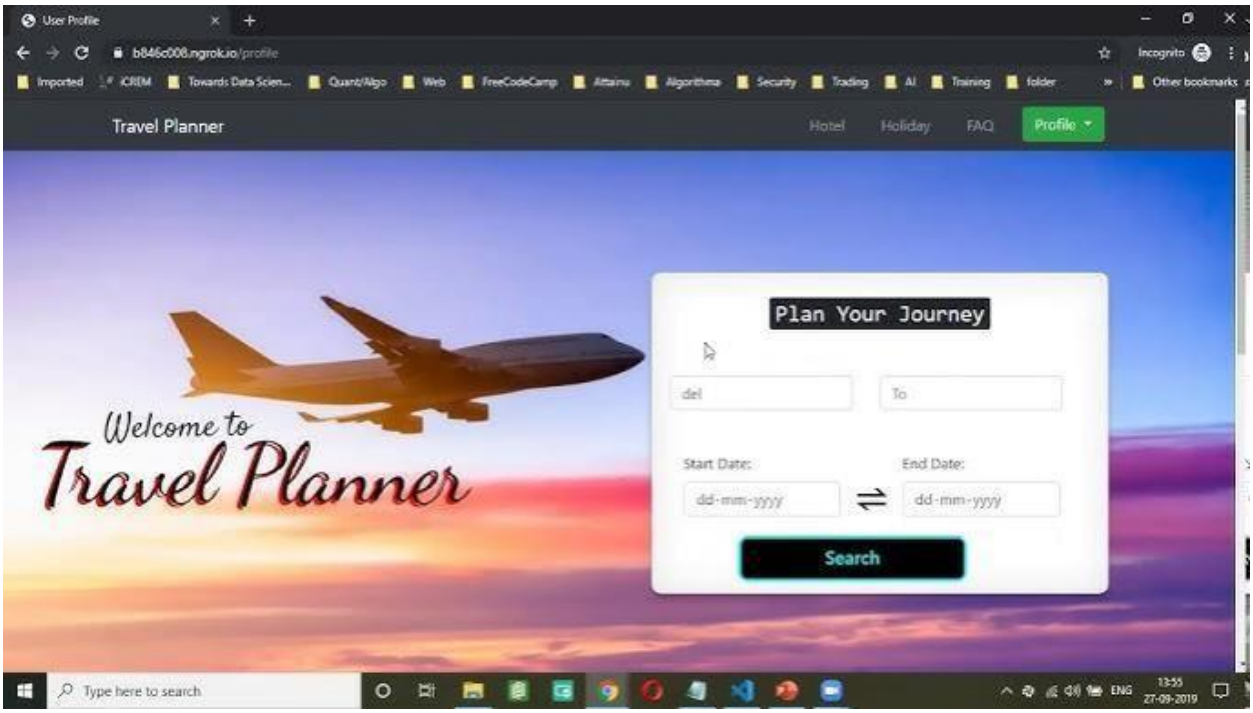
Features and Functionalities:-

1. User Registration and Login
2. Destination Search and Recommendation
3. Itinerary Planning and Management
4. Flight and Hotel Booking Integration
5. Travel Plan Sharing and Collaboration
6. Destination Reviews and Ratings
7. Personalized Travel Recommendations

System Architecture:-

The Travel Planner application will follow a microservices architecture, with the following components:

- Front-end: Java-based web application using Spring Boot and Thymeleaf
- Back-end: Java-based RESTful API using Spring Boot and Hibernate
- Database: MySQL relational database management system
- Authentication: OAuth 2.0 and OpenID Connect



Database Design:-

The database design will consist of the following tables:

- Users: stores user information, such as username, password, and email
- Destinations: stores destination information, such as name, description, and location
- Itineraries: stores itinerary information, such as destination, start date, and end date
- Bookings: stores booking information, such as flight and hotel reservations
- Reviews: stores review information, such as rating and comment

Front-end Development:-

The front-end will be developed using Java-based Spring Boot and Thymeleaf. The UI will be designed using HTML, CSS, and JavaScript.

Back end development :-

The back-end will be developed using Java-based Spring Boot and Hibernate. The RESTful API will be designed to handle requests and responses.

Coding:-

Step 1: Set up the project structure

Create a new directory for the project and create the following subdirectories:

- `travel-planner`
  - `src`
    - `main`
      - `java`
      - `resources`
    - `test`
      - `java`
      - `resources`
  - `pom.xml` (if using Maven) or `build.gradle` (if using Gradle)

Step 2: Set up the database

Create a new MySQL database for the application:

- Database name: `travel_planner`
- Username: `travel_planner_user`
- Password: `travel_planner_password`

Create the following tables:

- `users`
  - `id` (primary key)
  - `username`
  - `password`
  - `email`
- `destinations`
  - `id` (primary key)
  - `name`
  - `description`
  - `location`
- `itineraries`
  - `id` (primary key)
  - `user_id` (foreign key referencing `users.id`)
  - `destination_id` (foreign key referencing `destinations.id`)
  - `start_date`
  - `end_date`

- `bookings`
  - `id` (primary key)
  - `itinerary_id` (foreign key referencing `itineraries.id`)
  - `flight_id` (foreign key referencing `flights.id`)
  - `hotel_id` (foreign key referencing `hotels.id`)
- `flights`
  - `id` (primary key)
  - `airline`
  - `departure_airport`
  - `arrival_airport`
  - `departure_time`
  - `arrival_time`
- `hotels`
  - `id` (primary key)
  - `name`
  - `location`
  - `rating`

Step 3: Set up the backend

Create a new Java class `TravelPlannerApplication.java` in the `src/main/java` directory:

```
Import org.springframework.boot.SpringApplication;
Import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
Public class TravelPlannerApplication {

  Public static void main(String[] args) {
    SpringApplication.run(TravelPlannerApplication.class,  args);
  }
}
```

Create a new Java class `User.java` in the `src/main/java` directory:

```
Import javax.persistence.Entity;
Import  javax.persistence.GeneratedValue;
Import  javax.persistence.GenerationType;
Import javax.persistence.Id;

@Entity
Public class User {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  Private Long id;

  Private String username;
```

```
  Private String password;

  Private String email;

  // getters and setters
}
```

Create a new Java class `Destination.java` in the `src/main/java` directory:

```
Import javax.persistence.Entity;
Import  javax.persistence.GeneratedValue;
Import  javax.persistence.GenerationType;
Import javax.persistence.Id;

@Entity
Public class Destination {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  Private Long id;

  Private String name;

  Private String description;
```

```
  Private String location;

  // getters and setters
}
```

Create a new Java class `Itinerary.java` in the `src/main/java` directory:

```
Import javax.persistence.Entity;
Import  javax.persistence.GeneratedValue;
Import  javax.persistence.GenerationType;
Import javax.persistence.Id;
Import javax.persistence.ManyToOne;

@Entity
Public class Itinerary {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  Private Long id;

  @ManyToOne
  Private User user;
```

```
    @ManyToOne
    Private Destination destination;

    Private Date startDate;

    Private Date endDate;

    // getters and setters
}
```

Step 4: Set up the frontend

Create a new HTML file `index.html` in the `src/main/resources` directory:

```
<!DOCTYPE html>
<html>
<head>
    <title>Travel Planner</title>
</head>
<body>
    <h1>Travel Planner</h1>
    <form>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Password:</label>
```

```
    <input type=”password” id=”password” name=”password”><br><br>
    <input type=”submit” value=”Login”>
  </form>
</body>
</html>
```

Create a new CSS file `styles.css` in the `src/main/resources` directory:

```
Body {
  Font-family: Arial, sans-serif;
}

Form {
  Width: 50%;
  Margin: 40px auto;
  Padding: 20px;
  Border: 1px solid #ccc;
  Border-radius: 10px;
  Box-shadow: 0 0 10px rgba(
```

Travel planner execution:-

Step 1: User Registration

- The user opens the Travel Planner application and clicks on the “Register” button.

- The user is redirected to the registration page where they enter their details such as username, password, and email.

- The user clicks on the “Register” button to submit their details.

- The application creates a new user account and stores the user’s details in the database.

Step 2: User Login

- The user opens the Travel Planner application and clicks on the “Login” button.

- The user is redirected to the login page where they enter their username and password.

- The user clicks on the “Login” button to submit their credentials.

- The application verifies the user’s credentials with the stored data in the database.

- If the credentials are correct, the application logs the user in and redirects them to the dashboard.

Step 3: Destination Search

- The user clicks on the “Search Destinations” button on the dashboard.

- The user is redirected to the destination search page where they can enter their search criteria such as destination name, location, and dates.

- The user clicks on the “Search” button to submit their search criteria.

- The application searches for destinations that match the user’s search criteria and displays the results on the page.

Step 4: Itinerary Planning

- The user selects a destination from the search results and clicks on the “Plan Itinerary” button.

- The user is redirected to the itinerary planning page where they can enter their travel dates, accommodation preferences, and activities.

- The user clicks on the “Save” button to save their itinerary.

- The application creates a new itinerary for the user and stores it in the database.

Step 5: Booking Flights and Hotels

- The user clicks on the “Book Flights and Hotels” button on the itinerary planning page.

- The user is redirected to the booking page where they can select their flight and hotel options.

- The user clicks on the “Book” button to book their flights and hotels.

- The application integrates with a third-party booking API to book the user’s flights and hotels.

Step 6: Sharing Itinerary

- The user clicks on the “Share Itinerary” button on the itinerary planning page.

- The user is redirected to the sharing page where they can enter the email addresses of the people they want to share their itinerary with.

- The user clicks on the “Share” button to share their itinerary.

- The application sends an email to the specified email addresses with a link to view the user’s itinerary.

Security and Authentication:-

The application will use OAuth 2.0 and OpenID Connect for authentication and authorization.

Testing and Debugging:-

The application will be tested using Junit and Selenium. Debugging will be done using Eclipse and Chrome DevTools.

Deployment:-

The application will be deployed on a cloud platform, such as AWS or Google Cloud.

Conclusion:-

The Travel Planner application will be a comprehensive full-stack web application that allows users to plan and organize their trips. The application will be developed using Java-based Spring Boot and Hibernate, and will follow a microservices architecture.