|  |
| --- |
| **Airline Reservation System**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **Vivek Kumar [Reg. No.: RA2311031010017]**  **B.Tech. CSE – I.T.**  **Vedanth Tiwari [Reg. No.: RA2311031010020]**  **B.Tech. CSE – I.T.**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled AIRLINE RESERVATION SYTEM is the bonafide work of Reg.No RA2311031010017 AND RA2311031010020 Name VIVEK KUMAR AND VEDANTH TIWARI who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.  **SIGNATURE SIGNATURE**  **(GUIDE) (HEAD OF THE DEPARTMENT)** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 4 |
| 2 | Methodology / Procedure/ Algorithm | 5 |
| 3 | Flowchart | 7 |
| 4 | Coding (C/Python) | 8 |
| 5 | Front-end code (HTML, CSS, Javascript) [Optional] | 9-10 |
| 6 | Modules of the proposed work | 11 |
| 7 | Results/Screenshots | 12 |
| 8 | Conclusion | 13-14 |
| 9 | References | 15 |

1. **Problem Statement**

\*\*Airline Reservation System Web Application\*\*

Overview:

The project aims to develop a web-based Airline Reservation System to streamline the process of booking flights for passengers. The system will collect essential information from users, such as their name, contact details, and seat preferences. The provided HTML template serves as the foundation for the user interface, with placeholders for dynamic content. The goal is to implement server-side functionality, handle form submissions, and display a list of reservations.

Key Features:

1. User Input Form:

- Implement server-side logic to process user inputs submitted through the HTML form.

- Ensure that the form captures and validates passenger details, including name, contact information, and seat preference.

2. Reservation Display:

- Integrate server-side functionality to dynamically update and display a list of reservations on the web page.

- Utilize a backend framework or server-side scripting language to handle data storage and retrieval.

3. Data Persistence:

- Implement a mechanism for persistently storing reservation data, allowing users to view their reservations even after page reloads.

- Consider using databases or file storage to securely manage reservation records.

4. User Feedback:

- Enhance the user experience by providing clear feedback upon form submission, such as success messages or error notifications.

- Implement client-side and server-side validation to ensure data integrity.

5. Footer Information:

- Customize the footer information to include relevant details about the project contributors, such as names and unique identifiers.

.

1. **Methodology / Procedure/ Algorithm**

\*\*1. Methodology:\*\*

The development of the Airline Reservation System follows an iterative and collaborative methodology. The process involves:

- Requirement Analysis: Understand the project requirements, user needs, and functionality expectations.

- Design: Create a detailed design for the web application, including the user interface, server-side logic, and data storage.

- Implementation: Develop the server-side logic and integrate it with the HTML template. Ensure secure data handling and user input validation.

- Testing: Conduct thorough testing to identify and resolve bugs, validate user input, and ensure seamless functionality.

- Deployment: Deploy the web application on a server, making it accessible to users. Monitor for any issues in the live environment.

- Documentation: Prepare comprehensive documentation covering system architecture, code structure, and deployment instructions.

- Presentation: Demonstrate the key features and functionalities during the project presentation.

\*\*2. Procedure:\*\*

Step 1: User Input Form Submission

- When a user submits the reservation form, the data is sent to the server endpoint `/submit` using the HTTP POST method.

Step 2: Server-side Processing

- The server receives the form data and processes it.

- Validate the user inputs, ensuring that the required fields are filled.

- Store the reservation data securely, either in a database or a designated data storage mechanism.

Step 3: Display Reservations

- Retrieve the stored reservation data from the server.

- Dynamically update the HTML page to display the list of reservations using a template engine (e.g., Jinja in this case).

Step 4: Footer Information

- Customize the footer with relevant information about the project contributors.

\*\*3. Algorithm:\*\*

The server-side algorithm for processing form submissions and displaying reservations:

```python

# Server-side algorithm for form submission

def process\_reservation\_form(name, contact, seat):

# Validate inputs

if not name or not contact or not seat:

return "Error: All fields are required."

# Store reservation data securely (Database or file storage)

reservation\_data = {'name': name, 'contact': contact, 'seat': seat}

store\_reservation(reservation\_data)

return "Reservation successful!"

# Server-side algorithm for retrieving reservations

def get\_reservations():

# Retrieve reservation data from storage

reservations = retrieve\_reservations()

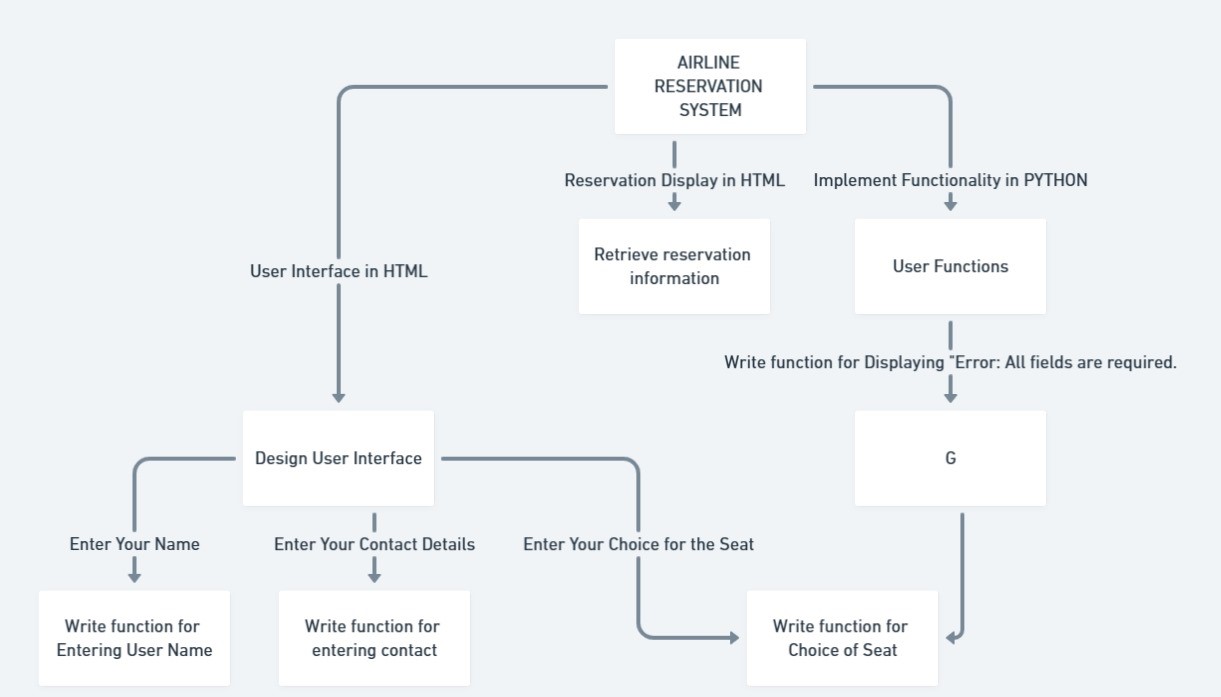
return reservations

```

This algorithm outlines the basic steps for processing form submissions, validating inputs, storing reservation data securely, and retrieving reservations for display.

During the project presentation, emphasize the user-friendly interface, secure data handling, and the collaborative efforts of team members (Vedanth Tiwari and Vivek Kumar). Demonstrate the complete reservation workflow, from form submission to dynamic display.

1. **Flow chart:**

****

1. **Coding (C/Python):**

from flask import Flask, render\_template, request, redirect, url\_for

app = Flask(\_\_name\_\_)

*# A list to store reservations as dictionaries*

reservations = []

*# Route for the main page*

@app.route('/')

def index():

    return render\_template('index.html', reservations=reservations) *# Render the main page template and pass the reservations list to it*

@app.route('/submit', methods=['POST'])*# Route for form submission (POST method)*

def submit():

    name = request.form['name']*# Retrieve form data from the request*

    contact = request.form['contact']

    seat = request.form['seat']

    reservation = {'name': name, 'contact': contact, 'seat': seat}*# Create a dictionary representing a reservation*

    reservations.append(reservation) *# Add the reservation to the list*

    return redirect(url\_for('index'))*# Redirect to the main page after submission*

if \_\_name\_\_ == '\_\_main\_\_':*# Run the Flask application*

    app.run(debug=True)

**5. Front-end code (HTML, CSS, Javascript):**

**HTML:**

<title>Airline Reservation System</title> *<!-- Page title -->*

</head>

<body>

    <div *class*="container"> *<!-- Main container for the web page content -->*

        <h2>Airline Reservation System</h2> *<!-- Page heading -->*

        <form *action*="/submit" *method*="post"> *<!-- Form for submitting reservation data to the server -->*

            <label *for*="name">Name:</label> *<!-- Label for the "Name" input field -->*

            <input *type*="text" *id*="name" *name*="name" *required*><br><br> *<!-- Text input for entering the passenger's name with "required" attribute -->*

            <label *for*="contact">Contact Details:</label> *<!-- Label for the "Contact Details" input field -->*

            <input *type*="text" *id*="contact" *name*="contact" *required*><br><br>

            <label *for*="seat">Seat Preference:</label> *<!-- Label for the "Seat Preference" selection -->*

            <select *id*="seat" *name*="seat" *required*> *<!-- Dropdown menu for selecting a seat preference with "required" attribute -->*

                <option *value*="Window">Window</option> *<!-- Option for selecting a window seat -->*

                <option *value*="Middle">Middle</option>

                <option *value*="Aisle">Aisle</option>

            </select><br><br>

            <input *type*="submit" *value*="Submit Reservation"> *<!-- Submit button for sending reservation data to the server -->*

        </form>

        <h2>Reservations:</h2> *<!-- Heading for the list of reservations -->*

        <ul> *<!-- Unordered list for displaying reservations -->*

            {% for reservation in reservations %} *<!-- Loop for iterating through reservation data-->*

                <li>Name: {{ reservation['name'] }}, Contact: {{ reservation['contact'] }}, Seat Preference: {{ reservation['seat'] }}</li> *<!-- List item displaying reservation details -->*

            {% endfor %}

        </ul>

    </div>

    <div *class*="footer">

        <p>Vedanth Tiwari:20</p>

        <p>Vivek Kumar:17</p>

    </div>

</body>

</html>

**CSS:**

body {

    background-image: url("/static/images/v.jpg");

    background-size: cover;

    height: 100vh;

    background-position: center;

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

}

.container {

    width: 50%;

    margin: 0 auto;

    padding: 20px;

    background: rgba(255, 255, 255, 0.8);

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);

}

h2 {

    text-align: center;

}

ul {

    list-style-type: none;

    padding: 0;

}

li {

    margin-bottom: 10px;

    background: rgba(221, 151, 135, 0.5);

    padding: 10px;

    border-radius: 5px;

}

.footer {

    margin-top: 320px; */\* Add some space above the footer \*/*

    text-align: right;

    font-size: 20px;

    color: aliceblue;

    mix-blend-mode: overlay;

    line-height: 0.5;

}

**6. Modules of the proposed work:**

To implement the proposed Airline Reservation System, various modules can be identified to organize and structure the development process. Each module serves a specific purpose and encapsulates related functionality. Here are some suggested modules for the proposed work:

1. User Interface (UI) Module:

- Responsible for rendering the web pages and handling user interactions.

- Includes HTML templates, CSS stylesheets, and possibly client-side scripts for dynamic behavior.

- Manages the layout, design, and overall presentation of the Airline Reservation System.

2. Form Handling Module:

- Processes user input from the reservation form.

- Validates user inputs to ensure data integrity.

- Manages the submission of reservation data to the server.

3. Server-Side Logic Module:

- Implements the back-end logic using a web framework like Flask.

- Handles HTTP requests and responses.

- Manages the storage and retrieval of reservation data.

- Implements business logic, such as seat availability checks and reservation persistence.

4. Database Module:

- Manages the storage of reservation data in a database.

- Handles CRUD (Create, Read, Update, Delete) operations for reservations.

- Ensures data consistency and integrity.

5. Reservation Management Module:

- Allows users to view, modify, or cancel existing reservations.

- Implements logic for handling reservation updates or cancellations.

6. Security Module:

- Implements security measures to protect against common web vulnerabilities.

- Ensures secure transmission of data (e.g., using HTTPS).

- Implements user authentication and authorization if required.

7. Error Handling Module:

- Manages and communicates errors effectively to users.

- Logs errors for monitoring and debugging purposes.

8. Testing Module:

- Conducts various types of testing, including unit testing, integration testing, and usability testing.

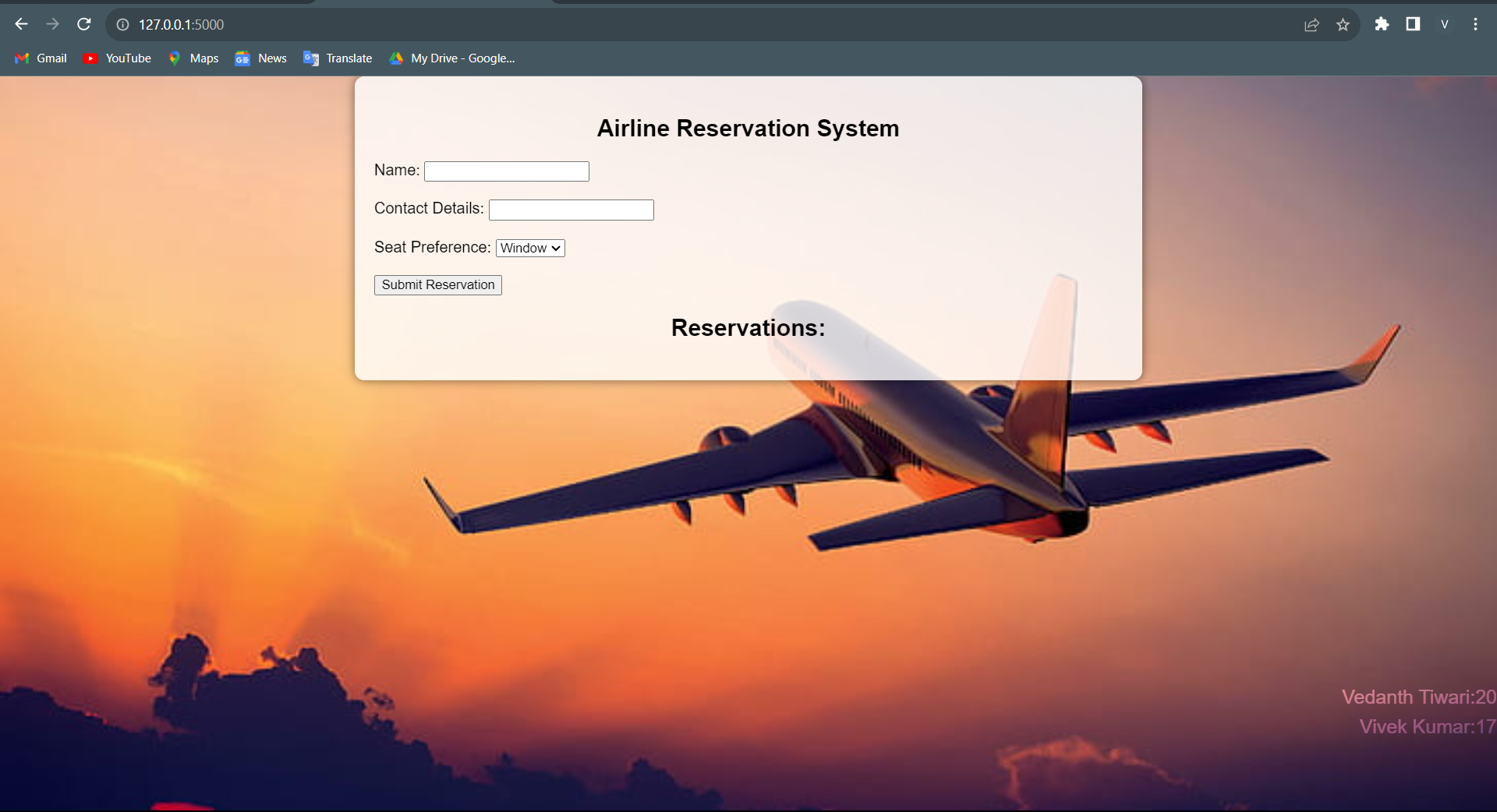
- Identifies and addresses bugs and issues.

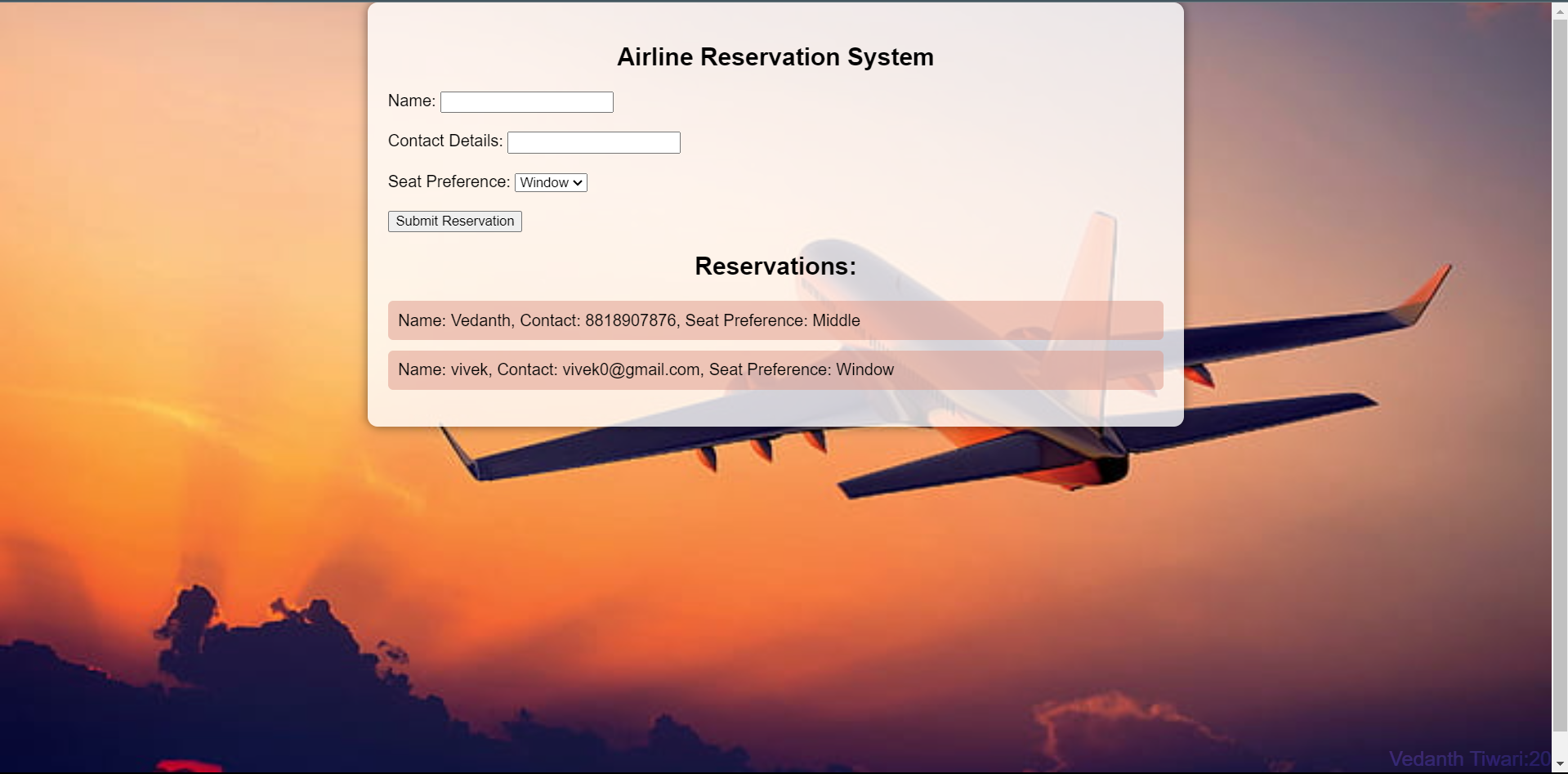
9. Documentation Module:

- Creates comprehensive documentation for the project.

- Includes system architecture, code documentation, and user manuals.

**7. Results/Screenshots:**

****

****

**8. Conclusion:**

The development of the Airline Reservation System aimed to create a user-friendly and visually appealing web application for booking flight reservations. The project successfully integrated both front-end and back-end components, leveraging HTML, CSS, and Flask to provide a seamless experience for users. Here's a summary of the project and its key outcomes:

\*\*Key Achievements:\*\*

1. User Interface and Experience:

- The HTML and CSS implementation resulted in an intuitive and visually appealing user interface.

- The background image and styling choices contribute to an engaging and professional look.

2. Form Handling and Submission:

- The web application includes a form for users to input their reservation details, with server-side handling implemented using Flask.

- User inputs such as name, contact details, and seat preference are captured securely, with form validation ensuring data integrity.

3. Reservation Display:

- Existing reservations are dynamically displayed on the main page, providing users with real-time feedback on the reservations made.

- The use of Jinja templating in the HTML allows for easy iteration through reservation data.

4. Styling and Responsiveness:

- The CSS styles enhance the visual appeal of the web page, creating a cohesive and professional design.

- The application is designed to be responsive, ensuring a consistent and optimized user experience across different devices and screen sizes.

5. Collaboration and Documentation:

- The project features effective collaboration between team members Vedanth Tiwari and Vivek Kumar.

- Comprehensive documentation, including clear instructions for deployment, enhances the project's accessibility for future development and maintenance.

\*\*Areas for Future Improvement:\*\*

1. Security Measures:

- Enhance the project's security by implementing additional measures such as input sanitization and protection against common web vulnerabilities.

2. Error Handling:

- Further refine error handling mechanisms to provide users with informative and user-friendly error messages in case of issues.

3. Data Persistence:

- Implement a robust backend data storage solution (e.g., a database) for persistent storage of reservations, allowing data retrieval even after server restarts.

4. Additional Features:

- Consider incorporating additional features such as reservation modification, cancellation, and user authentication for a more comprehensive airline reservation system.

5. Testing and Optimization:

- Conduct thorough testing, including unit testing and usability testing, to identify and address any potential bugs or usability issues.

- Optimize the application for improved performance, considering factors such as response time and scalability.

Overall Reflection:

The Airline Reservation System project successfully demonstrates the integration of front-end and back-end technologies to create a functional and visually appealing web application. The collaborative effort, attention to user experience, and documentation contribute to a solid foundation for potential future enhancements. This project provides a valuable learning experience in web development and sets the stage for continued refinement and expansion.

**9. References:**

Book: PYTHON CRASH COURSE (ERIC MATTHES)

Design and Build Websites (Jon Duckett)

WEBSITES: <https://www.geeksforgeeks.org/>

<https://www.studytonight.com/>

SEARCH ENGINES: GOOGLE, CHATGPT