A Mini Project Report

On

TRAVEL ITINERARY GENERATOR

By

K. SREE SAI CHANDANA

1602-22-733-122

S. SUSHANK

1602-22-733-125

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2024

ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Dr. M. Jithender Reddy**, Department of Computer Science and Engineering, whose guidance was unforgettable and filled with innovative ideas as well as her constructive suggestions has made the presentation of my major project a grand success.

We are thankful to **Dr. T. Adilakshmi**, Head of Department (CSE), **Vasavi College of Engineering** for the help during our course work.

Finally we express our gratitude to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete our project work successfully

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the Mini project entitled **"Travel Itinerary Generator"** being submitted by **K.Sree Sai Chandana** bearing **1602-22-733-122** and **S.Sushank** bearing **1602-22-733-125,** in partial fulfilment of the requirements of the V semester, Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

Dr.M.Jithender Reddy,
Assistant Professor,
Dept. of CSE,

Dr. T.Adilakshmi,
Professor & HOD,
Dept. of CSE

ABSTRACT

Travel Buddy is a user-friendly web application aimed at simplifying the complexities of travel planning. It empowers users to generate personalized travel itineraries based on basic inputs like destination, travel dates, and duration. By integrating real-time weather data and points of interest suggestions, it ensures a smooth and enriched travel experience.

Features:

1. Automated Itinerary Generation: Quickly compiles travel plans based on user preferences and schedules.

2. Real-Time Weather Updates: Integrates weather forecasts for travel dates to assist in activity planning.

3. Destination Highlights: Provides tailored recommendations for must-visit attractions and activities.

4. Intuitive Interface: Designed for seamless navigation, catering to travelers of all experience levels.

Technical Highlights:

- Frontend: Developed using HTML, CSS, and JavaScript for a responsive and interactive user experience.

- Backend: Powered by Flask (Python) for efficient server-side operations and API handling.

- APIs: Utilizes Google PaLM API for content generation and Visual Crossing Weather API for real-time data.

- Database: Implements Flask-SQLAlchemy (SQLite) for temporary data storage.

Travel Buddy not only saves time by automating trip planning but also enhances accessibility and convenience for users. Future enhancements include multi-destination support, integration of hotel and flight booking options, and advanced personalization tools to further optimize travel experiences.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Travel Buddy**: Simplifying Travel Planning and Enriching Experiences**

Travel Buddy is a comprehensive and innovative web application designed to tackle the common challenges faced by travellers during trip planning. From organizing itineraries to managing schedules and providing personalized recommendations, Travel Buddy serves as a one-stop solution that makes travel planning efficient, enjoyable, and accessible to all. By leveraging modern technologies, the platform ensures a seamless and interactive user experience, empowering individuals to plan their dream trips effortlessly.

The application is built to address key pain points for travellers, such as lack of centralized planning tools, limited access to accurate weather forecasts, and the difficulty of finding curated recommendations for destinations and activities. Travel Buddy bridges these gaps by automating the planning process and delivering detailed travel itineraries based on user inputs, including destination, travel dates, and trip duration. It combines weather forecasts, cost estimation, and local attraction suggestions into a single, cohesive plan, helping travelers save time and make informed decisions.

One of the standout features of Travel Buddy is its real-time weather integration, which ensures that users are equipped with accurate forecasts for their destinations. This enables travelers to prepare appropriately for their trips, from packing suitable clothing to selecting activities based on weather conditions. Additionally, the platform's itinerary generator provides detailed schedules tailored to user preferences, ensuring a balanced and enjoyable travel experience.

Travel Buddy prioritizes user convenience with its simple yet powerful interface. Built with HTML, CSS, and JavaScript, the frontend ensures a visually appealing and responsive design. The backend, powered by Python Flask, handles server-side operations efficiently, while integrated APIs like Google PaLM and Visual Crossing Weather API provide reliable content generation and weather data. Temporary data storage is managed securely through Flask-SQLAlchemy, enhancing both functionality and scalability.

Key Features and Technologies:

1. Automated Itinerary Generation: Generates comprehensive, personalized travel plans based on user inputs.
2. Real-Time Weather Updates: Delivers accurate forecasts to help users plan activities and prepare for their trips.

3. Points of Interest Recommendations: Highlights top attractions and activities at the selected destination.
4. User-Friendly Interface: A clean, intuitive design ensures seamless navigation and ease of use.
5. Tech Stack: Combines HTML, CSS, JavaScript, Python Flask, and APIs for robust performance and data handling.

Impact and Vision:

Travel Buddy is more than just a tool; it is a companion designed to revolutionize how travelers plan their journeys. By automating complex planning tasks and providing essential travel insights, it reduces the stress of trip organization and enhances accessibility for users of all experience levels.

As the platform evolves, future developments will include multi-destination support, hotel and flight booking integrations, and advanced personalization tools to cater to diverse travel needs. Real-time collaboration features will also enable groups to plan trips collectively, fostering a more interactive and engaging user experience.

With its focus on automation, personalization, and ease of use, Travel Buddy aims to set a new standard in travel planning. By simplifying the process and delivering value-driven solutions, it inspires travelers to explore the world with confidence, convenience, and joy.

2. LITERATURE SURVEY

The concept of travel itinerary planning has evolved significantly with the advent of digital tools and platforms designed to simplify the process of organizing trips. Popular platforms like Google Travel, TripIt, and Kayak have set benchmarks by offering features such as real-time booking, itinerary synchronization, and travel alerts. These platforms cater to a wide range of travelers, from casual vacationers to business professionals, providing a centralized space for managing all travel-related details.

While these platforms excel in convenience and functionality, they often cater predominantly to Western markets, with features optimized for users in those regions. This focus creates challenges for users in countries like India, where travel preferences, planning processes, and available infrastructure differ significantly. The lack of localization limits their applicability and usability in such diverse contexts.

Challenges in Existing Platforms

1. **Limited Personalization:**
   Many platforms provide generic travel plans that do not cater to specific user preferences. They lack advanced features for budget estimation, group travel considerations, or flexible itineraries tailored to individual needs.

2. **Dependency on Third-Party Services:**
   Existing tools often rely heavily on third-party integrations for flight bookings, hotel reservations, or activity scheduling, which can lead to fragmented user experiences. For users in India, these services might not always align with local travel preferences or budgetary constraints.

3. **Inadequate Weather Integration:**
   While many platforms include basic weather forecasts, they fail to provide detailed, actionable insights tailored to specific activities or travel dates. This limits users' ability to make informed decisions about their trip plans.

4. **Lack of Local Context:**
   Global platforms often overlook region-specific features such as local attractions, transportation options, or cultural nuances. This results in less effective recommendations for Indian travelers who value these localized insights.

5. **Limited Accessibility for Non-Tech-Savvy Users:**
   Most platforms assume a certain level of tech proficiency, leaving behind

users who may find navigating these tools cumbersome. This poses a barrier for first-time or casual travelers looking for simple solutions.

Uniqueness of Travel Buddy

Travel Buddy addresses these gaps by providing a platform tailored to the diverse needs of users in India and beyond. Its focus on simplicity, personalization, and local relevance sets it apart from existing solutions.

1. **Automated Itinerary Generation:**
   Travel Buddy generates personalized travel plans based on user inputs such as destination, duration, and travel dates. By leveraging AI-powered tools like Google PaLM API, it ensures itineraries are not only optimized for time and budget but also enriched with local recommendations for attractions and activities.

2. **Real-Time Weather Integration:**
   Unlike many platforms, Travel Buddy integrates detailed weather forecasts through the Visual Crossing Weather API. This ensures that users can adapt their plans based on daily weather conditions, making their travel experience more seamless and enjoyable.

3. **Points of Interest Recommendations:**
   Travel Buddy provides curated suggestions for must-visit places, tailored to the user's destination and travel preferences. This feature emphasizes local attractions, ensuring that travelers discover unique experiences.

4. **User-Friendly Design:**
   Travel Buddy's clean and intuitive interface, built with HTML, CSS, and JavaScript, ensures accessibility for users of all technical skill levels. Features such as straightforward forms and visual itineraries make the planning process hassle-free.

5. **Integrated Cost Estimation:**
   A standout feature of Travel Buddy is its ability to estimate travel costs, including transportation, accommodations, and activities. This helps users stay within their budget while making informed choices.

6. **API-Driven Features:**
   By utilizing robust APIs, the platform ensures reliable content generation and real-time updates. This includes both itinerary suggestions and weather forecasts, making Travel Buddy a comprehensive tool for travel planning.

Impact of Travel Buddy

By addressing the limitations of existing platforms, Travel Buddy bridges the gap between global travel solutions and the specific needs of Indian travelers. Its user-centric approach ensures that planning a trip becomes an easy, enjoyable process, regardless of the user's experience level or preferences.

Travel Buddy's localized focus, robust features, and adaptability set it apart as a leading travel planning tool. Future enhancements, including multi-destination itineraries, hotel and flight bookings, and collaborative planning tools, aim to elevate the platform further. By combining the best practices of global platforms with region-specific innovations, Travel Buddy aspires to become the preferred travel companion for users seeking efficient and enriched travel experiences.

3. BACKGROUND

Travel Buddy is a dynamic and innovative platform built to address the complexities of travel planning and deliver a seamless user experience. It leverages a modern web development stack centered on Flask for backend functionality and HTML, CSS, and JavaScript for frontend design. By integrating advanced APIs like Google PaLM and Visual Crossing Weather API, Travel Buddy automates itinerary generation and provides real-time data, helping users create personalized travel plans effortlessly.

Designed with scalability, security, and user-friendliness in mind, the platform caters to travelers of all experience levels. Its modular and robust architecture ensures adaptability, enabling the platform to evolve and incorporate additional features in the future.

Core Components of the Travel Buddy Stack

1. Flask – The Backend Framework:
   At the heart of Travel Buddy's server-side operations lies Flask, a lightweight and flexible Python framework. Flask's simplicity and power make it an excellent choice for building scalable and responsive web applications.
    - Routing: Flask ensures efficient handling of user requests, enabling smooth navigation between different pages, including itinerary generation and weather forecasts.
    - Integration of APIs: The framework manages seamless communication with external services like Google PaLM and Visual Crossing Weather APIs, ensuring real-time data retrieval and processing.
    - Session Management: Flask supports secure login and registration features, safeguarding user credentials and travel preferences.
2. HTML, CSS, and JavaScript – The Frontend Tools:
   Travel Buddy's user interface is built using HTML for structure, CSS for styling, and JavaScript for interactivity. Together, they ensure an engaging and intuitive experience for users:
    - HTML: Forms the foundation for displaying essential components such as input fields, generated itineraries, weather tables, and download options.

- CSS: Enhances the platform's aesthetic appeal by implementing clean layouts, visually appealing themes, responsive design, and animations.
- JavaScript: Drives the dynamic functionality of the platform, such as loading indicators, user feedback, and real-time rendering of itineraries and weather data.

3. Flask-SQLAlchemy – The Database Management System: Flask-SQLAlchemy acts as the database engine for Travel Buddy, handling temporary storage of user inputs and session data:
   - User Data: Includes login credentials and preferences.
   - Session Tracking: Ensures seamless transitions between user actions like itinerary generation and dashboard navigation.
   - Scalability: Supports future expansion with the ability to store more complex datasets, such as multi-destination itineraries or user history.
4. Google PaLM API – Personalized Itinerary Generation: Google PaLM API powers the platform's AI-driven itinerary generation feature. By analyzing user inputs such as destination, travel dates, and duration, the API creates optimized travel schedules that include:
   - Daily plans for activities and visits to local attractions.
   - Suggestions for balancing leisure and sightseeing based on travel duration.
   - Tailored recommendations to fit within user-specified budgets.
5. Visual Crossing Weather API – Real-Time Weather Integration: Accurate weather data is crucial for effective travel planning. Travel Buddy integrates the Visual Crossing Weather API to provide users with:
   - Comprehensive forecasts for their destination, including temperature, precipitation probability, and weather conditions.
   - Insights to help users prepare for activities, clothing, and unexpected weather changes.

Enhanced Functionality and Security

To ensure smooth operations and protect user data, Travel Buddy integrates several additional tools and features:

1. HTML2PDF.js:
   Converts itineraries into downloadable PDF files, allowing users to access their plans offline. This feature is particularly beneficial for travelers in areas with limited internet connectivity.

2. Secure Session Management:
   Flask's session management, supported by a securely configured secret key, ensures that user interactions remain private and protected from unauthorized access.

3. CORS (Cross-Origin Resource Sharing):
   Enables safe and seamless communication between the frontend and backend systems, ensuring compatibility across different devices and browsers.

4. Error Handling and Debugging Tools:
   Robust error-handling mechanisms and development tools allow the platform to identify and resolve issues efficiently, enhancing overall reliability.

Key Benefits of the Platform

1. User-Friendliness:
   Travel Buddy's intuitive design ensures ease of use for all travelers, regardless of technical expertise. The platform's clean interface allows users to input travel details, generate itineraries, and view updates effortlessly.

2. Personalization:
   By leveraging AI, Travel Buddy delivers tailored travel plans that align with individual preferences, including budget, trip duration, and destination-specific interests.

3. Real-Time Updates:
   The integration of live weather data ensures that users are always informed, allowing them to adjust their plans proactively based on conditions.

4. Scalability:
   Travel Buddy's modular architecture ensures it can accommodate growing user demand and additional features, such as multi-destination trips or advanced cost tracking.

5. Offline Access:
   The ability to download itineraries as PDFs ensures users have access to their plans even when internet connectivity is unavailable.

Localization and Accessibility

Travel Buddy is designed to cater to a wide audience, with features that enhance its accessibility and usability:

- Simplified Navigation: The platform ensures that even non-tech-savvy users can navigate its interface with ease.
- Localized Suggestions: Recommendations for activities and attractions focus on unique local experiences, ensuring travelers make the most of their destinations.

Conclusion

By leveraging a robust web development stack and integrating advanced technologies, Travel Buddy delivers a platform that is both powerful and user-friendly. The combination of AI-powered itinerary generation, real-time weather data, and dynamic features positions Travel Buddy as a transformative solution for modern travel planning.

With its focus on personalization, security, and accessibility, the platform addresses the limitations of existing tools while introducing innovative features that enhance the user experience. Travel Buddy is poised to revolutionize travel planning, empowering users to explore the world with confidence, convenience, and joy.

4. DESIGN

The architecture of the Travel Buddy platform is thoughtfully designed to ensure scalability, security, and efficient interaction between its components. The system adopts a modular structure, where each layer handles specific responsibilities, enabling smooth user interactions and reliable functionality. Below is a detailed description of its key layers and their roles:

1. Frontend Layer (Client-Side)
    - Technology: HTML, CSS, and JavaScript
    - Components:
        - User Interfaces: Home page, Login, Signup, Itinerary Generation, Weather Dashboard, and Feedback Form.
        - Interactive Elements: Forms for user inputs (destination, travel dates), dynamic itinerary rendering, and real-time weather updates.
        - Animations: CSS-based transitions to enhance the user experience.
    - Responsibilities:
        - Capture user inputs through well-designed forms and interactive fields.
        - Render dynamic and responsive views based on backend data, ensuring compatibility across devices.
        - Communicate with the backend securely via API calls to retrieve itineraries, weather data, and other travel-related information.
        - Provide offline access by enabling users to download itineraries as PDF files.

2. Backend Layer (Server-Side)
    - Technology: Flask (Python)
    - Modules:
        - API Handlers: RESTful API endpoints to handle itinerary generation, weather data retrieval, and user authentication.
        - Authentication: Secure session management using Flask’s built-in mechanisms and secret keys.

        - Data Validation: Ensures the accuracy and integrity of user inputs before processing.
        - Middleware:
            - Handles cross-origin requests securely with CORS.
            - Manages request parsing to facilitate data handling between the frontend and backend.
    - Responsibilities:
        - Process incoming user requests, validate inputs, and execute business logic (e.g., generating personalized itineraries).
        - Facilitate secure and efficient communication with external APIs like Google PaLM and Visual Crossing Weather.
        - Manage user sessions and maintain secure access controls for user accounts.

3. Database Layer (Storage)
    - Technology: Flask-SQLAlchemy (SQLite)
    - Schemas:
        - User Schema: Stores user credentials, preferences, and session data.
        - Itinerary Schema: Temporarily holds generated itineraries for easy access and display.
        - Feedback Schema: Logs user feedback to improve platform performance.
    - Responsibilities:
        - Persist user session data and input details.
        - Enable quick retrieval of itineraries and travel-related information.
        - Support scalability by allowing the database to accommodate more complex structures in the future, such as multi-destination planning or user history logs.

4. API Integration Layer
    - Technology:
        - Google PaLM API for AI-driven itinerary generation.
        - Visual Crossing Weather API for real-time weather forecasting.
    - Responsibilities:

- Generate comprehensive travel plans tailored to user inputs, considering factors like destination, duration, and budget.
- Fetch accurate and detailed weather forecasts for the travel dates, including daily conditions, temperature ranges, and precipitation probabilities.

Key Benefits of the Design:

- Scalability: The layered structure allows the system to handle increased user traffic and additional features, such as multi-destination planning or hotel and flight booking integration.
- Security: Secure API communications, session management, and data validation ensure user data integrity and privacy.
- User-Friendly: The separation of frontend and backend responsibilities ensures a responsive, intuitive, and accessible interface.
- Efficiency: Modular components and APIs enable efficient data retrieval, processing, and presentation, resulting in a smooth user experience.

Conclusion

This layered architecture ensures that Travel Buddy is a secure, scalable, and efficient platform. By combining robust backend processing with dynamic frontend interactions and reliable external APIs, it provides users with a seamless and personalized travel planning experience. The modular design allows for easy enhancements, ensuring that the platform evolves to meet the growing demands of modern travelers.

4.1. SYSTEM ARCHITECTURE DIAGRAM

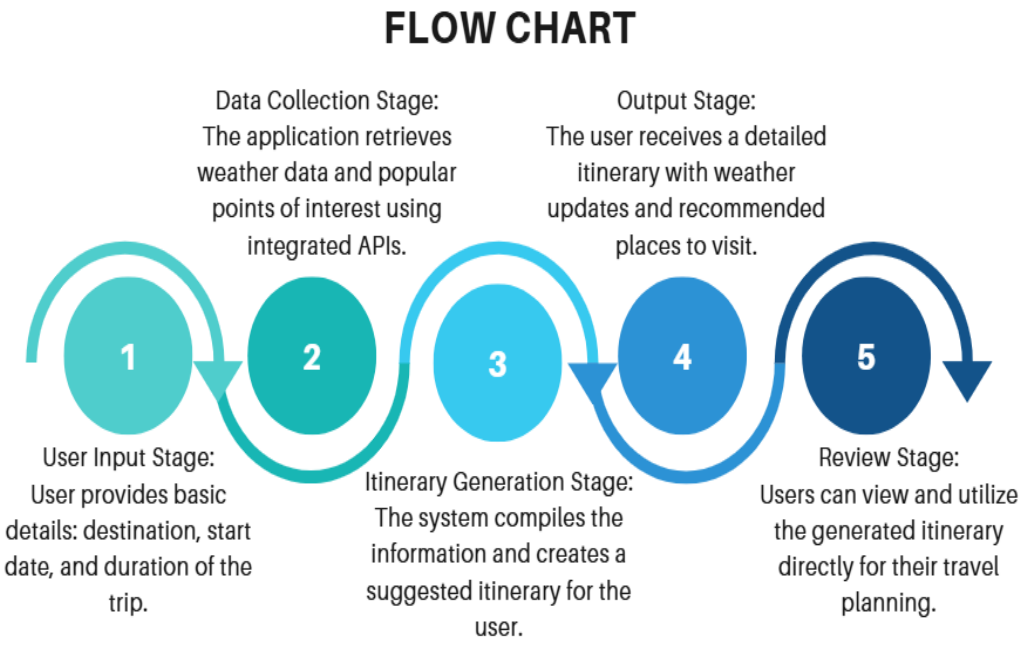


Figure 4.1. System Architecture Diagram

5. IMPLEMENTATION

1. Frontend Implementation:

HTML, CSS, and JavaScript:

- HTML: The structure of the web pages, including sections like Home, Login, Signup, Generate Itinerary, and View Itinerary, is built using HTML. Each page is designed to be simple and user-friendly, with clear calls to action.
- CSS: Custom styling is applied for responsiveness and visual appeal. Media queries are used to ensure the site works well across all screen sizes.
- JavaScript: Used for form validations (for user input such as dates, budget, etc.) and dynamic content loading. JavaScript also handles the real-time interaction with the backend, such as displaying itineraries and updating the map.

2. Backend Implementation:

Python with Flask:

- Flask Routes: RESTful API endpoints are created using Flask for functionalities like user registration, login, itinerary generation, and fetching data such as flight and hotel details.
    - POST /login: Handles user login and session creation.
    - POST /signup: Registers new users and stores their information.
    - POST /generate-itinerary: Receives travel preferences (destination, budget, number of days, etc.) and generates an itinerary.
- Database Interaction: The database is managed using Python's SQLAlchemy for data storage, including user details and generated itineraries.
    - User Data: User profiles with details like name, email, and preferences.
    - Itinerary Data: Travel plan details, including transportation options, accommodation, and activities.

- Business Logic: The backend handles complex logic, like calculating the total cost of the trip, suggesting activities based on the destination, and integrating with third-party APIs (like weather and flight information).

3. Database Implementation:

SQLite:

- User Table: Stores user credentials and preferences (destination, travel dates, budget).
- Itinerary Table: Stores generated itineraries, including destination, transportation, accommodation details, activities, and budget breakdown.
- CRUD Operations: Allows for creating, reading, updating, and deleting user data and itineraries based on user requests.

4. Security Implementation:

Password Security:

- Hashing: Passwords are securely stored using hashing techniques, such as bcrypt, to ensure user data safety.
- Session Management: Secure cookies or tokens are used to manage user sessions, ensuring that only authenticated users can access protected routes (e.g., My Itinerary).

5. Third-Party Integrations:

External APIs:

- Weather API: Retrieves real-time weather conditions for the user's destination to help users pack appropriately and plan activities.
- Flight and Hotel Booking APIs: Fetches available flights and hotel options based on the user's itinerary to make booking easier.
- Maps API: Provides interactive maps for users to visualize their destinations, transportation routes, and locations of interest.

This implementation ensures that the Travel Itinerary Generator is functional, secure, and user-friendly. Each part of the project was designed with the goal of creating a seamless experience for users from generating their travel plans to visualizing their itineraries in real-time.

.

5.1. PSEUDO CODE

Index.html

```
{% extends "headers.html" %}
{% block content %}
   <main>
     <div class="container">
       <div class="row">
         <div class="col">
           <h1 class="text-center">Travel Itinerary Generator</h1>
         </div>
       </div>
       <section class="section-home">
         <div class="intro-text">
            <h1>Welcome to Travel Buddy</h1>
            <p>Plan your dream trip with ease</p>
         </div>
      </section>
      <form action="/" method="POST" class="form-inline">
        <div class="form-group">
          <label for="source">From:</label>
          <input
            type="text"
            class="form-control"
            name="source"
            id="source"
            placeholder="Enter your boarding point"
            required
```

```
        />
      </div>
      <div class="form-group">
        <label for="destination">To:</label>
        <input
          type="text"
          class="form-control"
          name="destination"
          id="destination"
          placeholder="Enter your destination"
          required
        />
      </div>
      <span class="form-group">
        <label for="date">Travel Date:</label>
        <input type="date" class="form-control" name="date" id="date"required/>
        <label for="return">Return Date:</label>
            <input type="date" class="form-control" name="return" id="return" required/>
      </span>
     <button style="margin-top: 5px; padding: 4px;" type="submit" class="btn btn-primary">Submit</button>
        <div  class="container d-flex justify-content-center align-items-center">
          <div class="text-center">
                                <img id="loading" style="display: none;" src="{{url_for('static',filename='loading.gif')}}" alt="Loading..." />
          </div>
        </div>
    </form>
  </main>
  <div class="floating-container">
```

```
<div class="floating-button">
   <i class="fa-solid fa-share-from-square"></i>
</div>
<div class="element-container">
                                                    <a class="float-element"
href="https://www.facebook.com/profile.php?id=61556915904075"
target="_blank">
      <span>
      <i class="fa-brands fa-facebook"></i>
    </a>
    </span>
   <a class="float-element" href="https://wa.me/9182972398" target="_blank">
    <span>
      <i class="fa-brands fa-whatsapp"></i>
    </span>
    </a>
            <a class="float-element" href="https://x.com/SirangiSushank"
target="_blank">
    <span>
      <i class="fa-brands fa-x-twitter"></i>
    </span>
    </a>
        <a class="float-element" href="https://www.linkedin.com/in/sree-sai-
chandana-kunta" target="_blank">
    <span >
      <i class="fa-brands fa-linkedin"></i>
    </span>
    </a>
    <a class="float-element" href="http://t.me/Chad131095" target="_blank">
    <span >
      <i class="fa-brands fa-telegram"></i>
```

```
        </span>
        </a>
      </div>
    </div>
    <footer
      class="jumbotron text-center text-white fixed-bottom"
      style="background-color: #030303"
    >
      <div
        class="text-center p-3"
        style="background-color: rgba(255, 255, 255, 0)"
      >
        <p>
          Made by
          <a style="color: #0ed9ddd6; text-decoration: none" href="#"
            >Sushank and Chandana</a
          >
        </p>
      </div>
    </footer>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min
.js"
                                                    integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq
46cDfL"
      crossorigin="anonymous"
    ></script>
    <script>
      setTimeout(function () {
        const alerts = document.querySelectorAll(".alert");
```

```
        alerts.forEach(function (alert) {
          alert.style.opacity = 0;
        });
        setTimeout(function () {
          alerts.forEach(function (alert) {
            alert.style.display = "none";
          });
        }, 1000);
      }, 5000);
      document.querySelector('form').addEventListener('submit', function() {
        document.getElementById('loading').style.display = 'block';
      });
  </script>
    {% endblock %}
    </body>
</html>
```

Dashboard.html

```
{% extends "headers.html" %}
{% block content %}
<main>
  <div class="container">
    <h1>Weather Information</h1>
    <h5><b>Location:</b> {{weather_data['resolvedAddress']}}</h5>
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Date</th>
          <th>Current Weather Conditions</th>
          <th>Max Temperature (in °C)</th>
```

```
        <th>Min Temperature (in °C)</th>
        <th>Precipitation Probability</th>
        <th>Humidity</th>
        <th>Weather Alerts</th>
      </tr>
    </thead>
    <tbody>
      {% for weather_data_item in weather_data['days'] %}
      <tr>
        <td>{{ weather_data_item['datetime'] }}</td>
        <td>{{ weather_data_item['conditions'] }}</td>
        <td>{{ weather_data_item['tempmax'] }}</td>
        <td>{{ weather_data_item['tempmin'] }}</td>
        <td>{{ weather_data_item['precipprob'] }}</td>
        <td>{{ weather_data_item['humidity'] }}</td>
        <td>{{ weather_data_item['description'] }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
 </div>
 <div class="container">
  <h1>Planned Itinerary</h1>
  <code><h6>
        **This is a tentative itinerary, so please be flexible, it may have
        some mistakes.
       </h6></code>
  <div id="markdown-content">{{ plan }}</div>
    <button id="download" class="btn btn-primary">Download Itinerary as PDF</button>
```

```
  </div>
  <div class="md-4 pd container">
    <h5>
      For Hotels & Flight Booking:
            <a style="text-decoration: none" href="https://www.booking.com" target="_blank">
        <button type="button" class="btn btn-light">
          Click Here
        </button>
      </a>
    </h5>
    <section id="globe-view" class="section-globe">
      <h2>View Locations on 3D Globe</h2>
      <button id="globe-btn">View 3D Globe</button>
    </section>
  </div>
</main>
<div class="floating-container">
  <div class="floating-button">
      <i class="fa-solid fa-share-from-square"></i>
  </div>
  <div class="element-container">
                                                <a class="float-element" href="https://www.facebook.com/profile.php?id=61556915904075" target="_blank">
          <span>
          <i class="fa-brands fa-facebook"></i>
      </a>
      </span>
      <a class="float-element" href="https://wa.me/9182972398" target="_blank">
      <span>
```

```
        <i class="fa-brands fa-whatsapp"></i>
    </span>
    </a>
                <a class="float-element" href="https://x.com/SirangiSushank" target="_blank">
    <span>
        <i class="fa-brands fa-x-twitter"></i>
    </span>
    </a>
            <a class="float-element" href="https://www.linkedin.com/in/sree-sai-chandana-kunta" target="_blank">
    <span >
        <i class="fa-brands fa-linkedin"></i>
    </span>
    </a>
    <a class="float-element" href="http://t.me/Chad131095" target="_blank">
    <span >
        <i class="fa-brands fa-telegram"></i>
    </span>
    </a>
  </div>
</div>
<footer class="jumbotron text-center text-white fixed-bottom" style="background-color: #030303;">
  <div class="text-center p-3" style="background-color: rgba(255, 255, 255, 0)">
    <p>
      Made By
       <a style="color: #0ed9ddd6; text-decoration: none" href="#">Sushank and Chandana</a>
    </p>
  </div>
```

```
</footer>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10.1/html2pdf.bundle.min.js"

                                                    integrity="sha512-GsLlZN/3F2ErC5ifS5QtgpiJtWd43JWSuIgh7mbzZ8zBps+dvLusV+eNQATqgA/HdeKFVgA5v3S/cIrLF7QnIg=="

  crossorigin="anonymous" referrerpolicy="no-referrer"></script>

<script>

  setTimeout(function () {

    const alerts = document.querySelectorAll(".alert");

    alerts.forEach(function (alert) {

      alert.style.opacity = 0;

    });

    setTimeout(function () {

      alerts.forEach(function (alert) {

        alert.style.display = "none";

      });

    }, 1000);

  }, 5000);

</script>

<script      src="https://cdn.jsdelivr.net/npm/markdown-it@11.0.1/dist/markdown-it.min.js"></script>

<script>

  const md = window.markdownit();

  const html = md.render(

    document.getElementById("markdown-content").textContent

  );

  document.getElementById("markdown-content").innerHTML = html;

  document.getElementById('download').addEventListener('click', function() {

    var element = document.querySelector('main');
```

```
    var opt = {
      margin: 0.5,
      filename: 'itinerary.pdf',
      image: { type: 'png', quality: 100 },
      html2canvas: { scale: 0.8 },
      jsPDF: { unit: 'in', format: 'letter', orientation: 'portrait' },
    };
      html2pdf().set(opt).from(element).toPdf().get('pdf').then(function(pdf) {
      var totalPages = pdf.internal.getNumberOfPages();
      for (var i = 1; i <= totalPages; i++) {
        pdf.setPage(i)
      }
      pdf.save('itinerary.pdf');
    });
  });
</script>
<script>
document.getElementById('globe-btn').addEventListener('click', function() {
  window.open('https://earth.google.com/web/', '_blank');
});</script>
{% endblock %}
</body>
</html>
```

App.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_sqlalchemy import SQLAlchemy
from flask_sitemapper import Sitemapper
import bcrypt
import requests
```

```
import datetime
import bard,os
from dotenv import load_dotenv
load_dotenv()
api_key = os.environ.get("WEATHER_API_KEY")
secret_key = os.environ.get("SECRET_KEY")
app = Flask(__name__)
sitemapper = Sitemapper(app=app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
db = SQLAlchemy(app)
app.secret_key = secret_key
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)
    def __init__(self, name, email, password):
        self.name = name
        self.email = email
        self.password = bcrypt.hashpw(password.encode(
            'utf8'), bcrypt.gensalt()).decode('utf8')
    def check_password(self, password):
        return bcrypt.checkpw(password.encode('utf8'), self.password.encode('utf8'))
with app.app_context():
    db.create_all()
def get_weather_data(api_key: str, location: str, start_date: str, end_date: str) -> dict:
                                                    base_url =
f"https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/{location}/{start_date}/{end_date}?unitGroup=metric&include=days&key={api_key}&contentType=json"
```

```
    try:
        response = requests.get(base_url)
        response.raise_for_status()
        data = response.json()
        return data
    except requests.exceptions.RequestException as e:
        print("Error:", e.__str__)
@sitemapper.include()
@app.route('/', methods=["GET", "POST"])
def index():
    if request.method == "POST":
        global source, destination, start_date, end_date
        source = request.form.get("source")
        destination = request.form.get("destination")
        start_date = request.form.get("date")
        end_date = request.form.get("return")
            no_of_day = (datetime.datetime.strptime(end_date, "%Y-%m-%d") -
datetime.datetime.strptime(start_date, "%Y-%m-%d")).days
        if no_of_day < 0:
            flash("Return date should be greater than the Travel date (Start date).",
"danger")
            return redirect(url_for("index"))
        else:
            try:
                    weather_data = get_weather_data(api_key, destination, start_date,
end_date)
            except requests.exceptions.RequestException as e:
                flash("Error in retrieving weather data.{e.Error}", "danger")
                return redirect(url_for("index"))
        try:
```

```
            plan = bard.generate_itinerary(source, destination, start_date, end_date, no_of_day)
        except Exception as e:
            flash("Error in generating the plan. Please try again later.", "danger")
            return redirect(url_for("index"))
        if weather_data:
            return render_template("dashboard.html", weather_data=weather_data, plan=plan)
    return render_template('index.html')
@sitemapper.include()
@app.route("/about")
def about():
    return render_template("about.html")
@sitemapper.include()
@app.route("/contact")
def contact():
    user_email = session.get('user_email', "Enter your email")
    user_name = session.get('user_name', "Enter your name")
    message = "
            return render_template("contact.html", user_email=user_email, user_name=user_name, message=message)
@sitemapper.include()
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form.get("email")
        password = request.form.get("password")
        user = User.query.filter_by(email=email).first()
        if user and user.check_password(password):
            session["user_id"] = user.id
            session["user_name"] = user.name
```

```
            session["user_email"] = user.email
            flash("Login successful.", "success")
            print(session["user_email"])
            return redirect(url_for("index"))
        else:
            flash("Wrong email or password. Please try again or register now.", "danger")
            return redirect(url_for("login"))
    else:
        return render_template("login.html")
@sitemapper.include()
@app.route("/logout")
def logout():
    session.clear()
    flash("Logged out.", "info")
    return redirect(url_for("login"))
@sitemapper.include()
@app.route("/register", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        name = request.form.get("name")
        email = request.form.get("email")
        password = request.form.get("password")
        password2 = request.form.get("password2")
        if password == password2:
            existing_user = User.query.filter_by(email=email).first()
            if existing_user:
                flash("User already exists. Please log in.", "danger")
                return redirect("/login")
            else:
```

```
            user = User(name=name, email=email, password=password)
            db.session.add(user)
            db.session.commit()
            return redirect("/login")
        else:
            flash("Passwords do not match.", "danger")
            return redirect("/register")
    else:
        return render_template("register.html")
@app.route('/robots.txt')
def robots():
    return render_template('robots.txt')
@app.route("/sitemap.xml")
def r_sitemap():
    return sitemapper.generate()
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404
@app.context_processor
def inject_now():
    return {'now': datetime.datetime.now()}
```

bard.py

```
import google.generativeai as palm
import os
from dotenv import load_dotenv
load_dotenv()
palm_api_key = os.environ.get("PALM_API_KEY")
palm.configure(api_key=palm_api_key)
```

```
model = palm.GenerativeModel(model_name="gemini-1.5-flash-8b-exp-0924")

def generate_itinerary(source, destination, start_date, end_date, no_of_day):

    prompt = f"Generate a personalized trip itinerary for a {no_of_day}-day trip from {source} to {destination} on {start_date} to {end_date}, with an optimum budget (Currency:INR)."

    response = model.generate_content(prompt)

    return(response.text)
```

.env:

```
WEATHER_API_KEY='MWAX9JCWMTJDPMTGVX4G6WW3X'

PALM_API_KEY='AIzaSyC69BwP0tbRMQyPpYhAsgMMfUM7tcNNNFc'
```

Wsgi.py

```
from app import app  # Import your Flask app instance

if __name__ == "__main__":

    app.config['TEMPLATES_AUTO_RELOAD'] = True

    app.run()
```

6. RESULTS

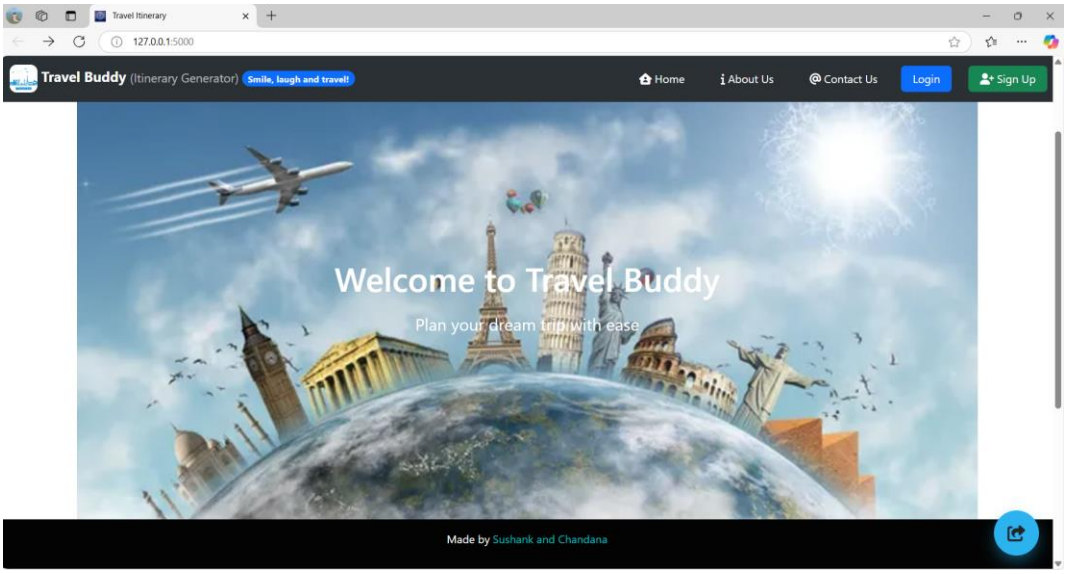


Figure 6.1. Home Page

When the website is started, this page is displayed with a navbar and basic information about the website.

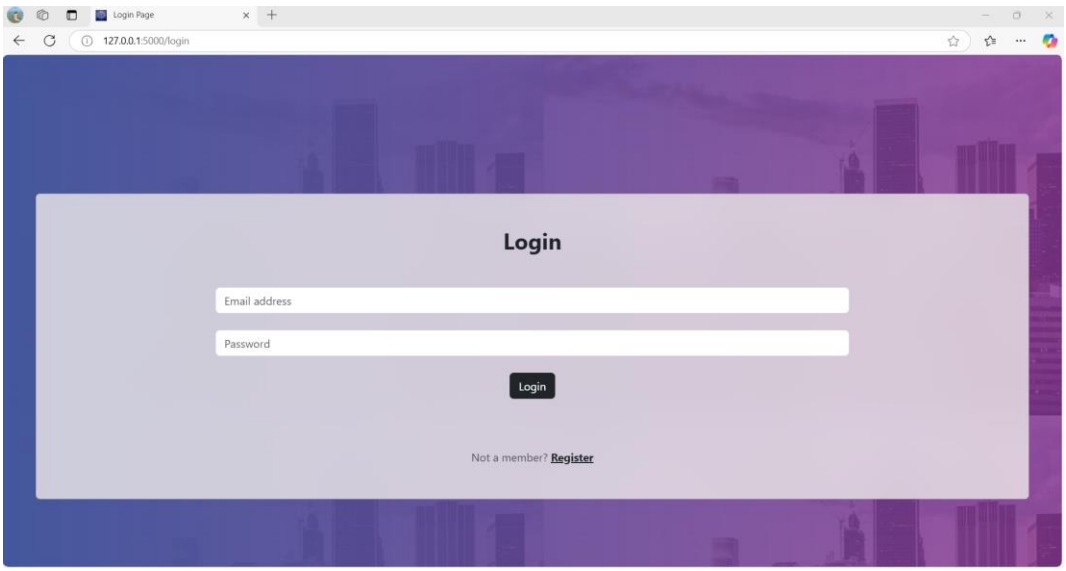


Figure 6.2. Login page

This is the Login Page,where existing users can login.

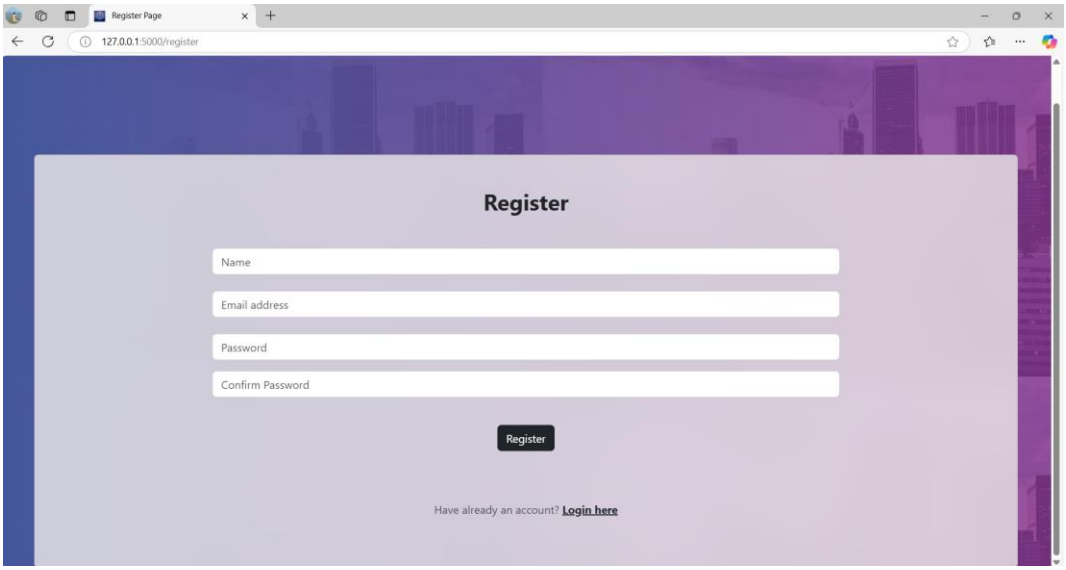
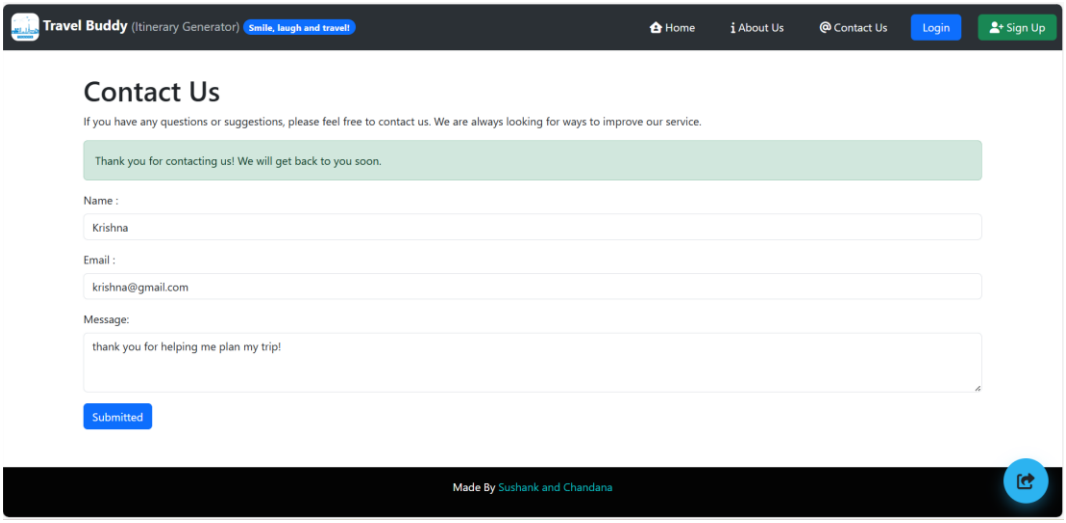


Figure 6.3. Sign Up Page

This is the Sign-Up page, where users can sign up by entering a name, email, and password. The password is encrypted and stored in the database.



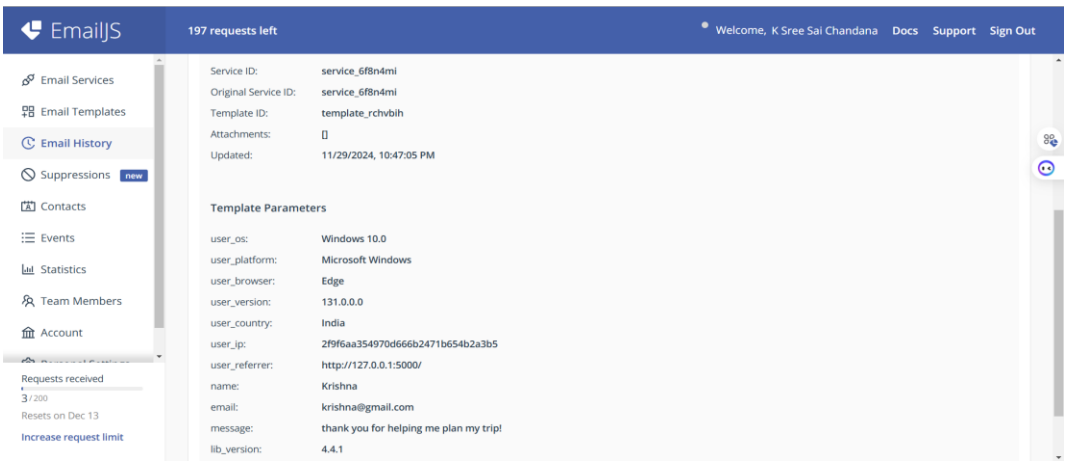


Figure 6.4. Contact Page

This is the Contact page, where users can Contact us by an email. The messages will reach me through emailjs platform.

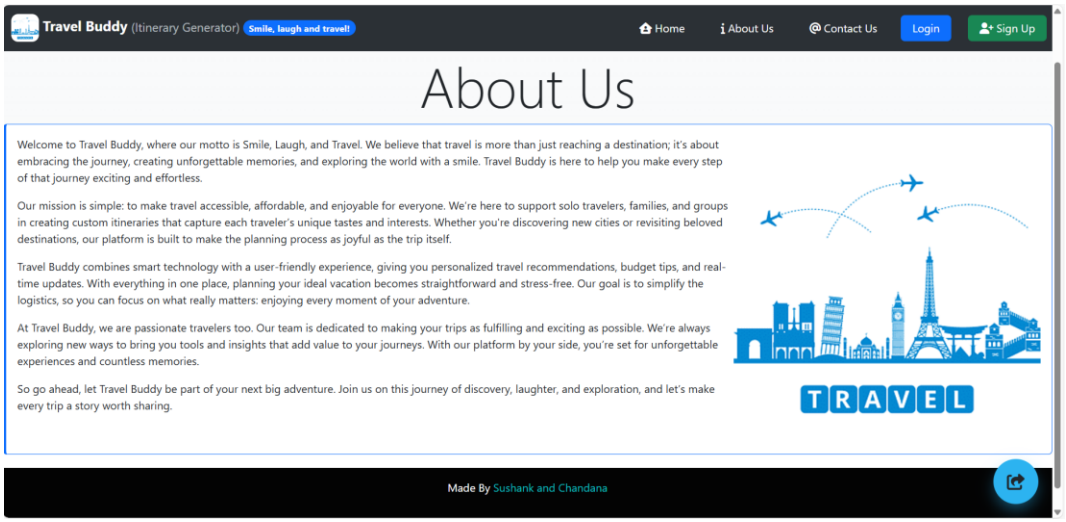


Figure 6.5. About Us Page

In this page user can get to know about our website and its purpose.

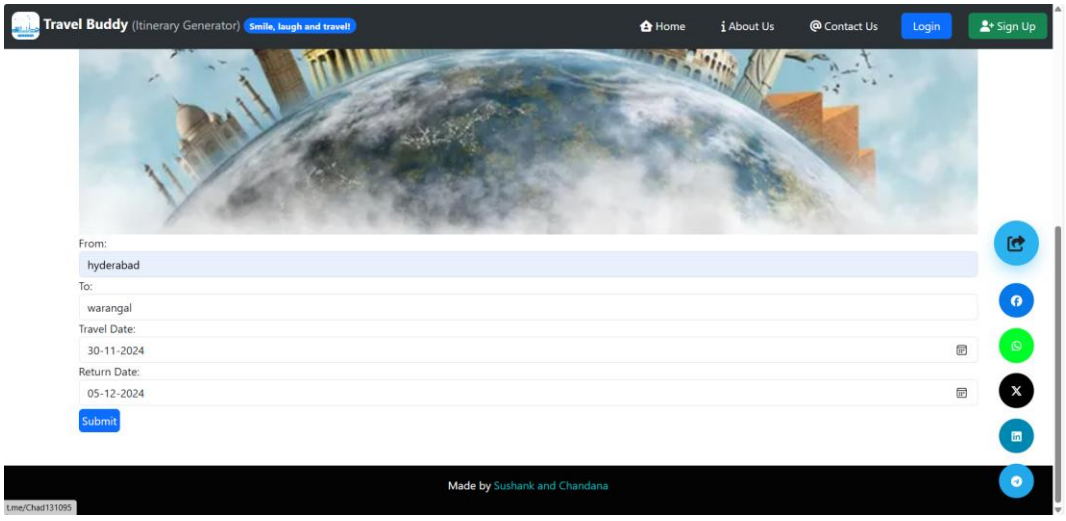


Figure 6.6. User input

This is a part of home page i.e. after scrolling down. Here the user can enter the source place, final destination, date of starting the journey and return date.

When he submits that form by clicking button, it will load and give the trip plan.

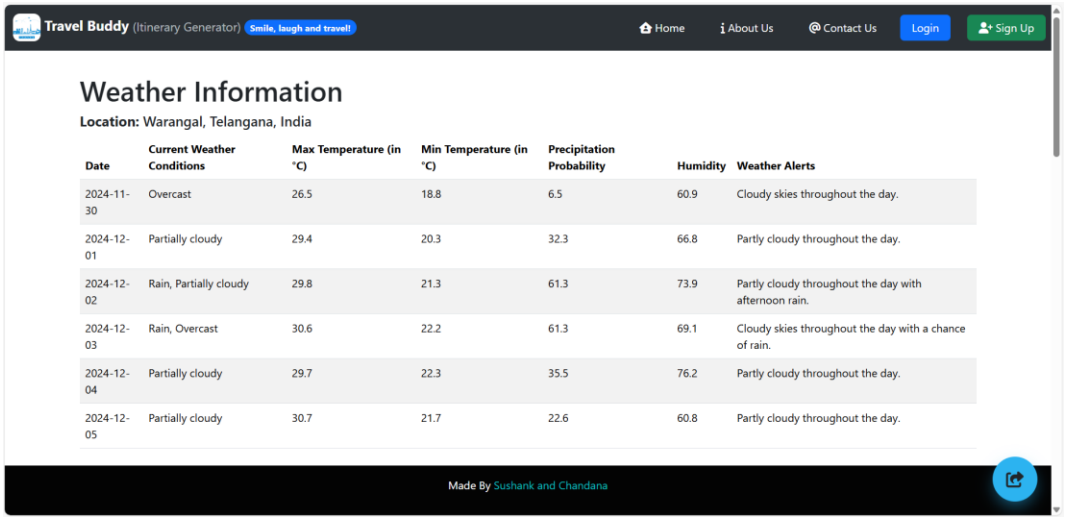
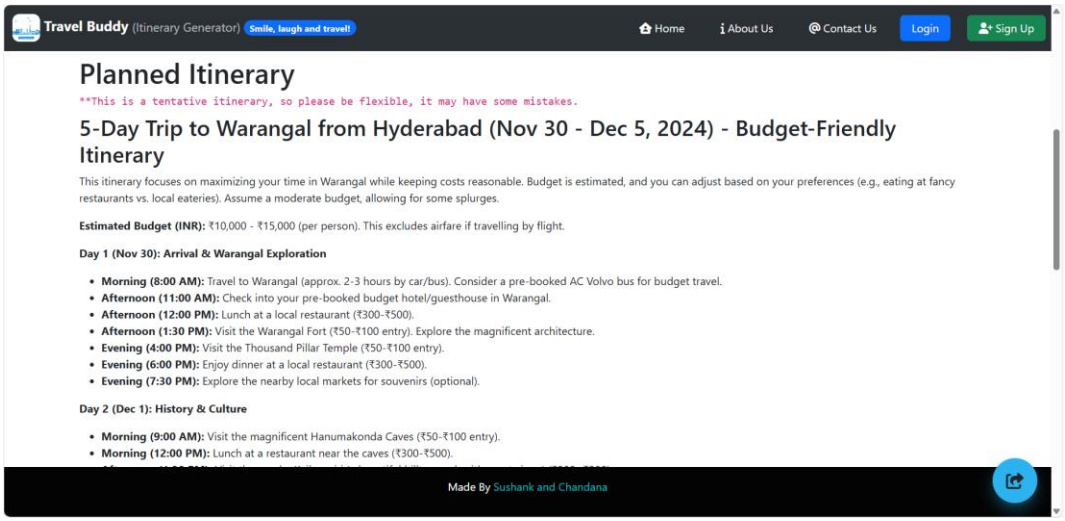
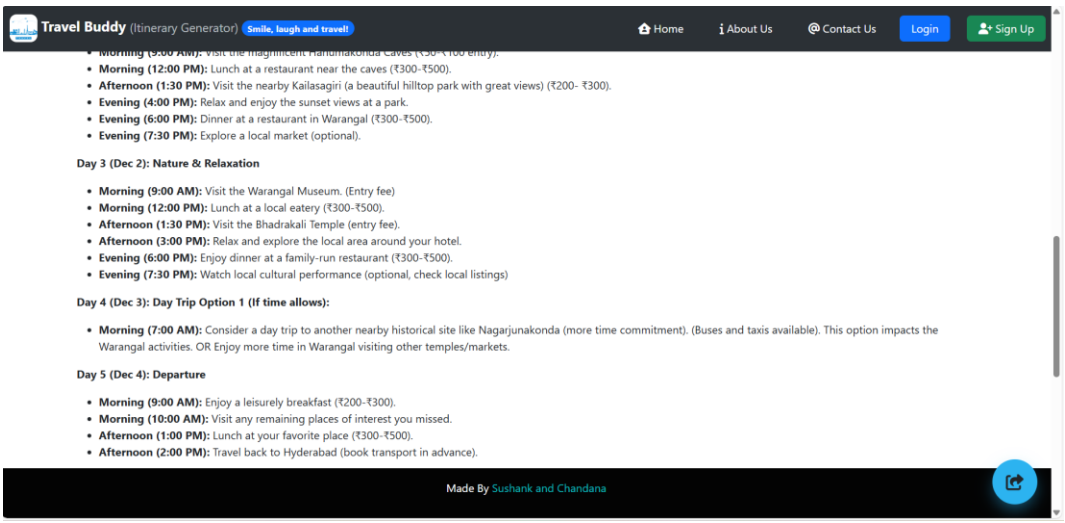


Figure 6.7. Weather information

This is a part of dashboard page and it’s showing the weather forecast for the place user would like to go .





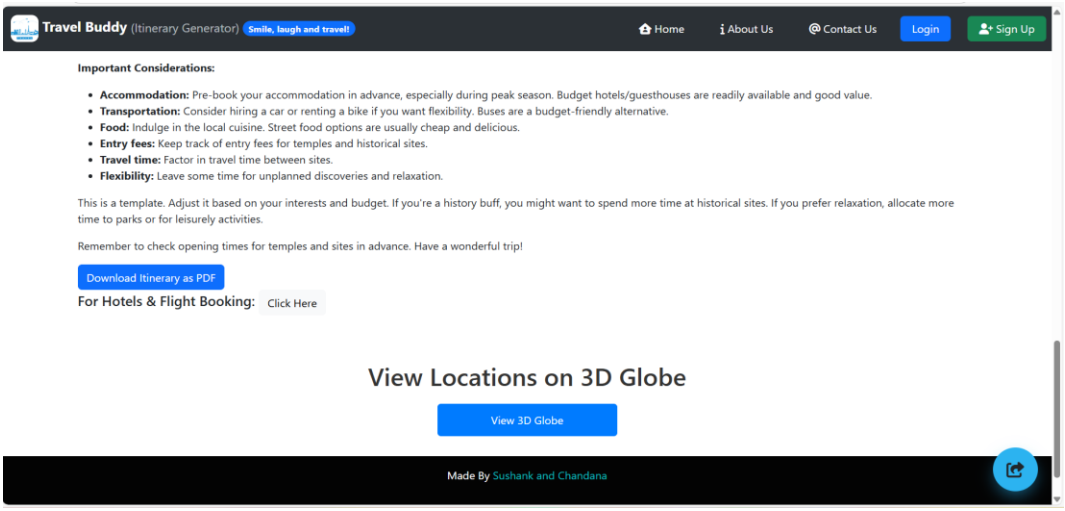
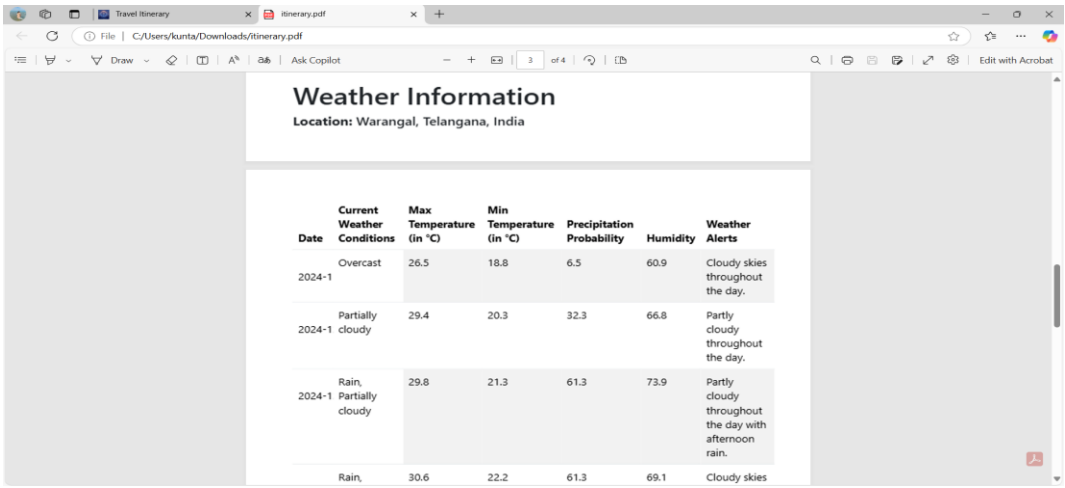


Figure 6.8. Itinerary generated

This part of dashboard page shows the day wise itinerary and few important considerations for the trip user wants to plan.



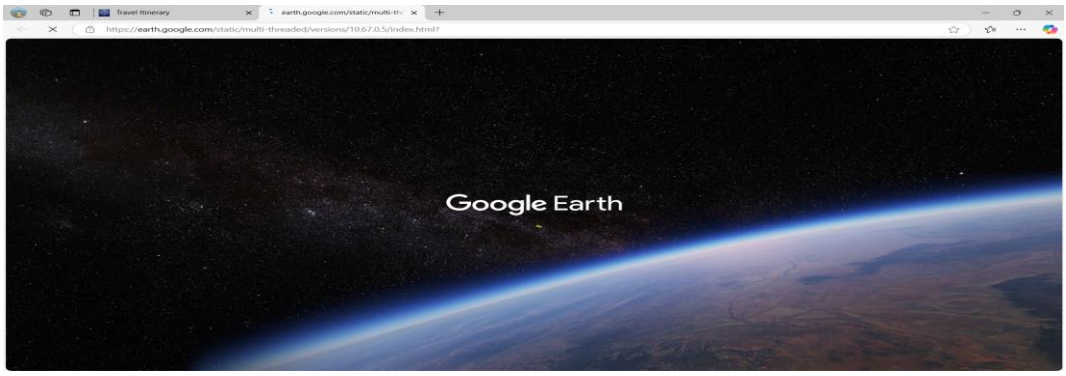


Figure 6.9. Converted to pdf and google earth

The itinerary generated can be converted into a pdf by clicking “download itinerary as pdf” and also a link to google earth is provided in the end.

7. CONCLUSION

The Travel Itinerary Generator project represents a significant step forward in simplifying travel planning for users. By combining effective front-end design with a robust back-end system, it provides a seamless and efficient platform for users to generate detailed, personalized itineraries based on their preferences.

Using a combination of HTML, CSS, and JavaScript for a responsive user interface, along with Python and Flask for secure backend functionality, the platform ensures that users can easily input their travel details and receive comprehensive itineraries that include transportation, accommodation, and activity recommendations. The integration of third-party APIs for weather updates, flight details, and maps further enhances the overall user experience.

This project stands out by offering a simple yet powerful solution for travelers, making it easier for them to plan their trips efficiently. By focusing on the needs of users, the Travel Itinerary Generator fosters a more streamlined, enjoyable travel planning process, bridging the gap between digital tools and the excitement of travel preparation.

7. FUTURE SCOPE

To further enhance the Travel Itinerary Generator and broaden its capabilities, the following improvements are proposed:

- Integration with Travel Agencies and Hotels: Collaboration with agencies and hotel booking platforms to offer direct booking options within the app, making the trip planning process more seamless.
- Mobile Application: Development of a cross-platform mobile app to provide users with on-the-go access to itinerary management, weather updates, and travel assistance, ensuring convenience while traveling.
- Personalized AI Travel Recommendations: Leveraging AI to recommend personalized destinations, accommodations, and activities based on user preferences, past trips, and real-time data.
- Real-Time Notifications: Implementation of push notifications and email alerts for users to receive updates on flight statuses, booking confirmations, or changes in weather conditions.
- Geolocation and Local Guide Integration: Adding location-based services to suggest nearby attractions, restaurants, or guides based on the user's itinerary and current location.
- Collaboration with Local Businesses: Partnering with local tour operators, restaurants, and activity providers to offer exclusive deals and promotions to users of the platform.
- Enhanced User Analytics: Incorporating an analytics dashboard to track user preferences, popular destinations, and common travel patterns, helping to improve future recommendations and marketing efforts.
- Social Media Integration: Enabling users to share their trip details, itineraries, and experiences on social media platforms, encouraging others to plan their trips using the generator.

These improvements will ensure that the Travel Itinerary Generator becomes an all-encompassing platform for travelers, providing not only trip planning but also personalized recommendations, bookings, and real-time assistance, making travel experiences more enjoyable and efficient.

9. REFERENCES

[1] "Travel Itinerary Generator: A Comprehensive Guide," Udemy Course, 2024.

[2] https://www.html5rocks.com/en/

[3] https://www.javascript.info/

[4] https://www.w3schools.com/js/

[5] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

[6] "Creating a Responsive Web Application with HTML, CSS, and JavaScript," W3Schools, 2024.

[7] "Building Scalable Web Applications," O'Reilly Media, 2020.