

Florida International University
Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: Travel Buddy

Team Members: Alejandro Morales, Gabriel Rodriguez, Haadiya Khan, Elena Sotolongo, Argelio Rodriguez

Product Owner(s): Haadiya Khan

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Table of Contents

Abstract	4
Introduction	5
Current System	5
Purpose of the New System	5
User Stories	6
Project Plan	8
Hardware and Software Resources	8
Sprints Plan	8
System Design	9
User Interface (UI)	9
Frontend Duties	10
Backend Duties	11
Combined Team Duties	11
Installation Guide	13
User Manual	15
Future Enhancements	17
Challenges and Lessons Learned	18
Appendix	19
A1. Home Page – Featured Destinations	19
A2. Flights Search – Expedia Integration	20
A3. Hotel Search Interface	21
A4. Things To Do – Events Filter	22
A5. Attractions Finder	23
A7. Login / Register Pages	24
A8. Poster and Demo Preparation	25

Abstract

This document outlines the development and implementation of **TravelBuddy**, a modular, responsive, full-stack web application designed to simplify travel planning. TravelBuddy brings together various fragmented functionalities—such as flight search, hotel discovery, attractions, events, and weather forecasting—into one unified platform.

Built using the Flask web framework, HTML/CSS, and third-party APIs (Amadeus, Expedia, Google Places, Ticketmaster, EventBrite, OpenWeather), TravelBuddy provides a scalable solution that improves the user experience and paves the way for future AI-driven travel assistance and smart itinerary generation. This report covers the system's technical design, sprint development cycle, implementation phases, user interface design, testing efforts, and future vision.

Introduction

TravelBuddy was designed to alleviate the complexities of travel planning by offering an all-in-one solution. The application enables users to search for flights, view and book hotels, explore local attractions, find nearby events, and get real-time weather—all through a single interface. It is scalable, modular, and serves as a foundation for more intelligent travel planning through AI.

Current System

Currently, users rely on disparate services (Expedia for flights, Booking.com for hotels, Google Maps for attractions, etc.). This fragmentation leads to a disjointed and time-consuming travel planning experience. None of these platforms offer holistic, customizable planning across multiple travel dimensions.

Purpose of the New System

TravelBuddy aims to:

- Centralize major travel planning services.
- Provide an intuitive, responsive user experience.
- Reduce cognitive overload during trip planning.
- Support future AI integrations for personalized travel assistance.

User Stories

User Story 1: As a developer, I want to set up the development, testing, and production environments so that the team can begin coding, testing, and deploying efficiently.

User Story 2: As a developer, I want to finalize and document the technology stack, including frontend, backend, database, and APIs to ensure team alignment.

User Story 3: As a developer, I want to research third-party APIs for functionality, limitations, and cost to choose the most appropriate ones for our app.

User Story 4: As a developer, I want to create a basic user interface for the application that includes the main layout, homepage, and navigation to key modules.

User Story 5: As a developer, I want to integrate a simple weather feature so that users can view current weather forecasts for their destinations.

User Story 6: As a developer, I want to implement an accommodations feature so that users can search for hotels or rentals in their travel locations.

User Story 7: As a developer, I want to integrate a flight search feature so that users can search for and view available flights using Expedia's redirect system.

User Story 8: As a developer, I want to integrate a nearby attractions feature using Google Places API so that users can discover interesting places around their destination.

User Story 9: As a developer, I want to integrate a nearby events feature using Ticketmaster or Eventbrite so users can find and book tickets to local events.

User Story 10: As a developer, I want to build a sign-up page so that users can create accounts to access personalized features.

User Story 11: As a developer, I want to build a sign-in page so that users can securely log in to their accounts using session authentication.

User Story 12: As a developer, I want to build a verification system so that user email accounts can be verified through a secure process.

User Story 13: As a user, I want to filter events by date, category (e.g., music, sports, art), and location, and view detailed information about each event (e.g., description, time, location, ticket availability), so that I can find and evaluate events that match my interests and schedule.

User Story 14: As a user, I want to see recommendations for trending or popular events in my destination so that I can discover new and interesting activities.

User Story 15: As a team member, I want to design and finalize the showcase poster so that we can effectively communicate our project's purpose and value at presentations.

User Story 16: As a team member, I want to record and edit the project demo video to demonstrate the functionality of our application in a professional way.

User Story 17: As a team, we want to prepare a final deliverable package (including source code, documentation, video link, and project report) so we can submit a complete and polished project.

Project Plan

Hardware and Software Resources

- **Technologies used:** Python, HTML, CSS, Jinja2, Flask, JavaScript, REST APIs
- **Development environment:** Visual Studio Code
- **Version control:** Git and GitHub
- **Design tools:** Figma, Canva (for poster and mockups)
- **API Services:** Amadeus, Booking.com, OpenWeather, Google Places, Expedia (redirect), Ticketmaster, EventBrite.

Sprints Plan

1. Sprint 1: Initial Planning and Environment Setup

Set up development, testing, and deployment environments. Finalize the tech stack, project scope, and architectural structure. Begin API research and basic interface layout planning.

2. Sprint 2: Integration of Core Modules

Start developing core application features including the hotel and flight search components, and user authentication pages. Lay out the homepage and navigation.

3. Sprint 3: Frontend Development and API Implementation

Implement front-end interfaces for weather, hotel listings, flight search, and attractions. Connect APIs for live data retrieval and integrate results into responsive UI components.

4. Sprint 4: UI Refinement and Interactive Filtering

Improve interactivity by adding event filters and attraction category search. Refine UX/UI for event cards, attraction displays, and hotel booking components.

5. Sprint 5: Authentication and Final Feature Implementation

Complete sign-in/sign-up logic, verification page, and event recommendation module. Finalize core functionality and prepare application for testing.

6. Sprint 6: Developer Testing / User Testing / Feedback

Conduct manual testing across devices and browsers. Gather user feedback from mock showcase sessions. Adjust UI design, address bugs, and optimize functionality for a smoother experience.

7. Sprint 7: Deployment / Documentation / Final Presentation

Prepare final deliverables: demo video, showcase poster, and project documentation. Complete final deployment of the app and present the project at the class showcase on April 18, 2025.

System Design

User Interface (UI)

The user interface of TravelBuddy focuses on creating an intuitive and interactive experience that enables users to easily plan trips by navigating across multiple travel modules. The application emphasizes clean navigation, engaging visuals, and functional consistency.

Core Features:

- **Search & Explore:** Users can explore flights, hotels, weather, events, and attractions.
- **Interactive Forms:** Dynamic input forms for date, location, filters, and user preferences.
- **Responsive Cards:** Grid and card-based display for attractions, hotels, and events.
- **Session-based UI:** Personalized navigation once the user is logged in.
- **Visual Feedback:** Alerts, placeholder messages, and validation prompts guide users throughout their journey.

Destination & Attraction Details

Each major module displays relevant information for user decision-making:

- **Flights:** Departure and arrival location, date, link to Expedia search.
- **Hotels:** Hotel name, rating, price, location, and booking links.
- **Attractions:** Name, type (museum, park, etc.), address, rating, and image.
- **Events:** Event name, category (music, sports, etc.), time, date, location, image, and link to buy tickets.
- **Weather:** Real-time temperature, icon-based condition, location-based forecast (In Development).

Filtering & Personalization

The following features provide a custom experience for users:

- **Event Filters:** Users can filter events by category, date range, and location.
- **Attractions Filter:** Filter by type (e.g., zoo, park, museum) and rating threshold.
- **Trending Suggestions:** Smart event recommendations based on location (static logic or API-driven).
- **User Status:** UI adapts based on whether the user is logged in or not (session-based visibility).

Data Management

The backend manages interaction with live API data, simulates user data persistence, and handles request routing.

- **Flights, Events, Hotels, Attractions, Weather:** Fetched from third-party APIs and passed to templates.
- **City Validation:** Data is validated against `worldcities.csv` before API queries are processed.
- **User Session:** Basic session management with login/logout state maintained.
- **Data Persistence (*Future*):** Intended to expand into database-backed user data (e.g., saved trips, search history).

Frontend Duties

Designing the User Interface

- Develop consistent layout with `base.html` template.
- Create responsive components such as:
 - Search forms (flights, hotels, events, attractions)
 - Card layouts for listings
 - Navigation menu and sidebar
 - Reusable buttons and input fields

Implementing Interactivity

- Implement filters and form submission logic for:
 - Event types, dates, cities
 - Attraction categories and ratings
- Add client-side form validation and error messages for invalid inputs.
- Handle dynamic loading of user-dependent pages (e.g., toggle login/signup buttons).

Responsive Design

- Ensure layout functions smoothly across all screen sizes:
 - Desktop: Full-width grid views for hotels and events
- Use consistent styling through CSS and scalable units.
- Follow accessible design practices (color contrast, form labels).

Backend Duties

Develop the API & Routing Logic

- Define Flask routes for all major pages:
 - /, /flights, /hotels, /things-to-do, /map, /login, /register
- Process form input via POST/GET methods and pass cleaned parameters to API requests.
- Render returned API data using Jinja2 templates.
- Implement redirect logic for external platforms (e.g., Expedia).

Data Handling & Integration

- Connect to APIs using requests library:
 - Google Places, OpenWeather, Ticketmaster, Expedia (URL-based)
- Clean and standardize response data (e.g., missing ratings, photos, duplicates).
- Validate city inputs using worldcities.csv.

Security

- Perform input validation to avoid API errors and malformed requests.
- Sanitize user data before using in URL/query params.
- Use Flask session to store logged-in state securely.
- *(Future)* Integrate authentication tokens, database login, and CORS for production hosting.

Combined Team Duties

Documentation & Communication

- Maintain up-to-date sprint planning, retrospective summaries, and final report.
- Clearly outline API logic and expected input/output for frontend/backend synchronization.
- Document which templates consume which routes and which APIs power each page.

Coordination on API and Data Flow

- Backend must expose all necessary data to the frontend templates via Jinja variables.
- The front-end team ensures the proper presentation and user input to support API calls.
- Shared testing of final application for functional and UI bugs.

Testing & Deployment

- Team collaboratively tested:
 - Navigation flow
 - Filter and search results
 - API failure fallback handling
- Final build was verified for responsiveness and correctness across devices and screen sizes.
- Deployment readiness was tested through local runs and documented in the deliverables.

Installation Guide

This section provides a detailed step-by-step guide for setting up and running the TravelBuddy application on a local machine.

Prerequisites

Ensure the following software is installed on your system:

- Python 3.8 or later
- pip (Python package manager)
- Git (optional but recommended)
- Virtual environment module (venv)
- Visual Studio Code or another code editor
- Internet access for API integration

Step-by-Step Instructions

1. Download or Clone the Project Repository

Option A – Clone using Git:

```
git clone https://github.com/yourusername/All-in-one-travel-app-main.git
cd All-in-one-travel-app-main/capstone
```

Option B – Manually download the ZIP file:

- Extract the zip file
- Navigate to the capstone/ folder

2. Create and Activate a Virtual Environment (Recommended)

```
python -m venv venv
source venv/bin/activate    # For macOS/Linux
venv\Scripts\activate      # For Windows
```

3. Install Required Dependencies

```
pip install -r requirements.txt
```

If the requirements.txt file is not present, install dependencies manually:

```
pip install flask requests python-dotenv
```

4. Configure Environment Variables

Create a file named `.env` inside the capstone directory and add the following API keys:

```
GOOGLE_API_KEY=your_google_api_key  
WEATHER_API_KEY=your_openweather_api_key  
TICKETMASTER_API_KEY=your_ticketmaster_api_key
```

Replace each value with your actual API key. These keys are required for attractions, weather, and event modules to function properly.

5. Run the Application

If successful, Flask will start a local server on:

`http://127.0.0.1:5000/`

Open this link in your web browser to use TravelBuddy locally.

User Manual

This section provides an overview of how users can interact with each feature in the TravelBuddy application.

Login and Registration

- Users can register an account by entering their name, email, and password.
- Once registered, users may log in to access session-specific features.
- Logged-in users will see an updated navigation bar.

Homepage

- The homepage displays featured destinations including travel tips and weather overviews.
- Navigation cards are provided for quick access to Flights, Hotels, Events, and Attractions.

Flights Module

- Located at /flights
- Users enter a departure city, destination, and departure date
Upon submission, the app redirects to Expedia with the query pre-filled for real-time results

Hotels Module

- Located at /hotels
- Users enter a city code (e.g., LAX, NYC, MIA)
- Hotel listings are returned with names, ratings, images, and booking links

Events and Things to Do

- Located at /things-to-do
- Users enter a city and date range
- Filter options include event category (Music, Sports, Arts, Family)
- Results include event name, date, venue, description, and ticket purchase links

Attractions

- Located at /map
- Allows users to filter attractions by category (e.g., parks, museums, zoos)
- Displays results with names, images, ratings, and addresses
- Information is pulled from the Google Places API

Weather Module (In Development)

- May or may not provide current weather conditions based on selected location
- Data will include temperature, forecast, and weather icons
- Powered by OpenWeather API

Troubleshooting

Problem: Flask app not starting

Solution: Ensure you are in the correct directory and have activated the virtual environment

Problem: Hotel or event data not displaying

Solution: Confirm your API keys are correct and active; some APIs have daily rate limits

Problem: City not found

Solution: Use city codes or try a nearby major city; also check worldcities.csv as a reference

Problem: CSS not loading

Solution: Clear your browser cache or ensure that the static folder is linked properly

Future Enhancements

Short-Term Goals:

- Add a database (e.g., MongoDB or PostgreSQL) to persist user accounts and save trips.
- Implement a user dashboard for viewing past trips and recommendations.
- Integrate weather forecast overlay on Google Maps view.

Long-Term Goals:

- AI-generated itinerary based on trip duration, weather, and interests.
- Chatbot for real-time travel guidance using natural language input.
- Smart push notifications for price drops and event availability.
- Group trip planning with collaborative itinerary editing.

Challenges and Lessons Learned

Technical Challenges:

- API documentation inconsistencies made it difficult to structure unified responses.
- Google Places API had rate limits, forcing us to use dummy data during testing.
- Form handling required strict validation to prevent bad requests.

Team Coordination:

- Time zone differences and conflicting schedules led to slower communication during Sprint 3.
- Version control taught us the importance of pull requests and consistent commits.

Lessons Learned:

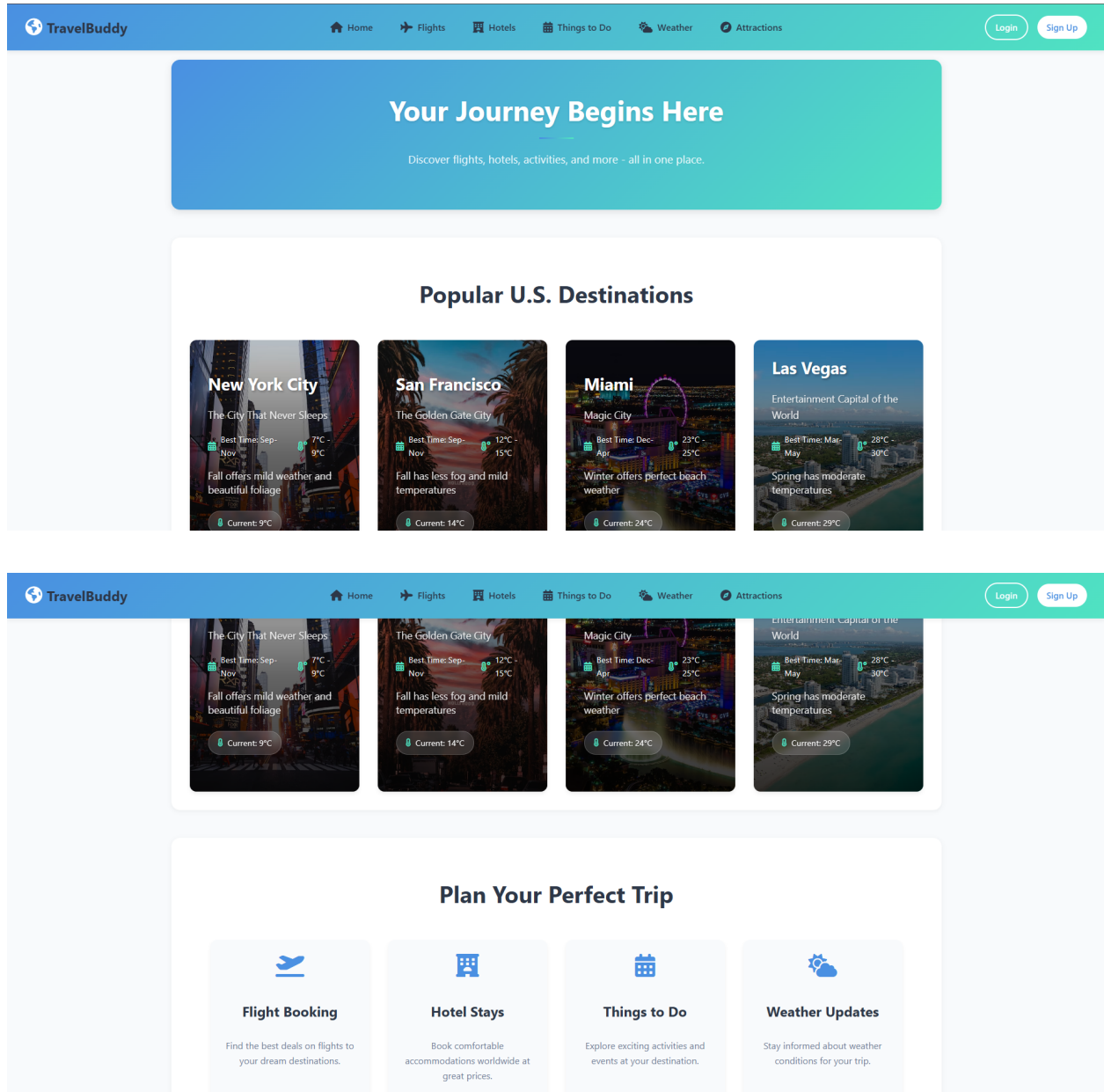
- Documenting every API's expected input/output early would've saved time.
- Keeping designs wireframed before coding improved implementation speed.
- Frontend-backend collaboration should include scheduled syncs and endpoint reviews.

Appendix

A1. Home Page – Featured Destinations

Description:

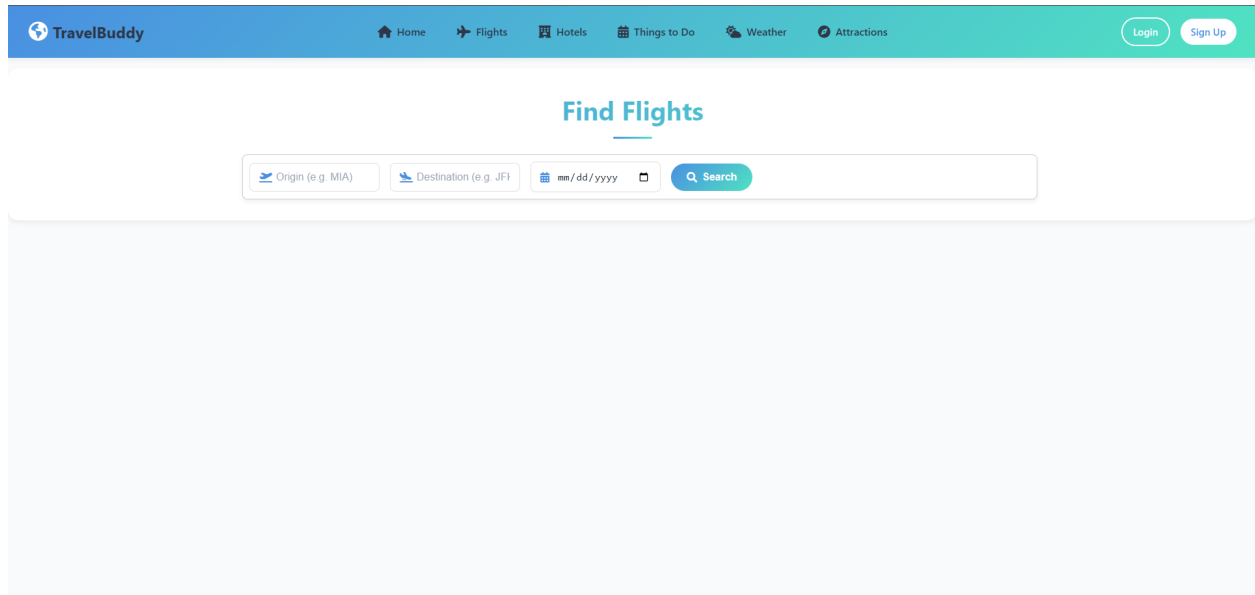
The homepage showcases a curated list of featured U.S. destinations, displaying city nicknames, ideal travel times, weather ranges, and helpful summaries. Quick navigation cards guide users to Flights, Hotels, Events, and Weather modules.



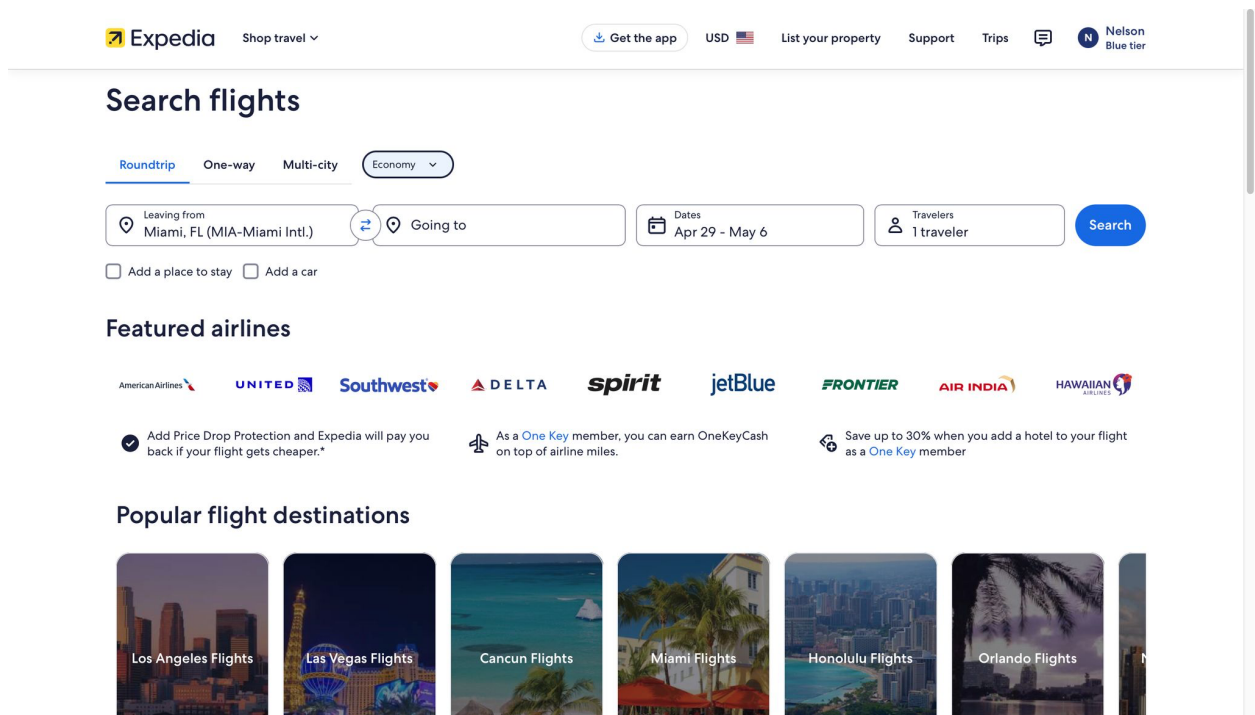
A2. Flights Search – Expedia Integration

Description:

The flight search form allows users to input their departure and destination cities, and travel date. Submitting redirects the user to Expedia with parameters pre-filled for a seamless booking experience.



The image shows a web form titled "Find Flights" from TravelBuddy. The form is located below a navigation bar with links for Home, Flights, Hotels, Things to Do, Weather, and Attractions. The form itself has a light blue header with the title "Find Flights". Below the header, there are three input fields: "Origin (e.g. MIA)", "Destination (e.g. JFI)", and "Dates (mm/dd/yyyy)". To the right of these fields is a "Search" button. The form is set against a light blue background.

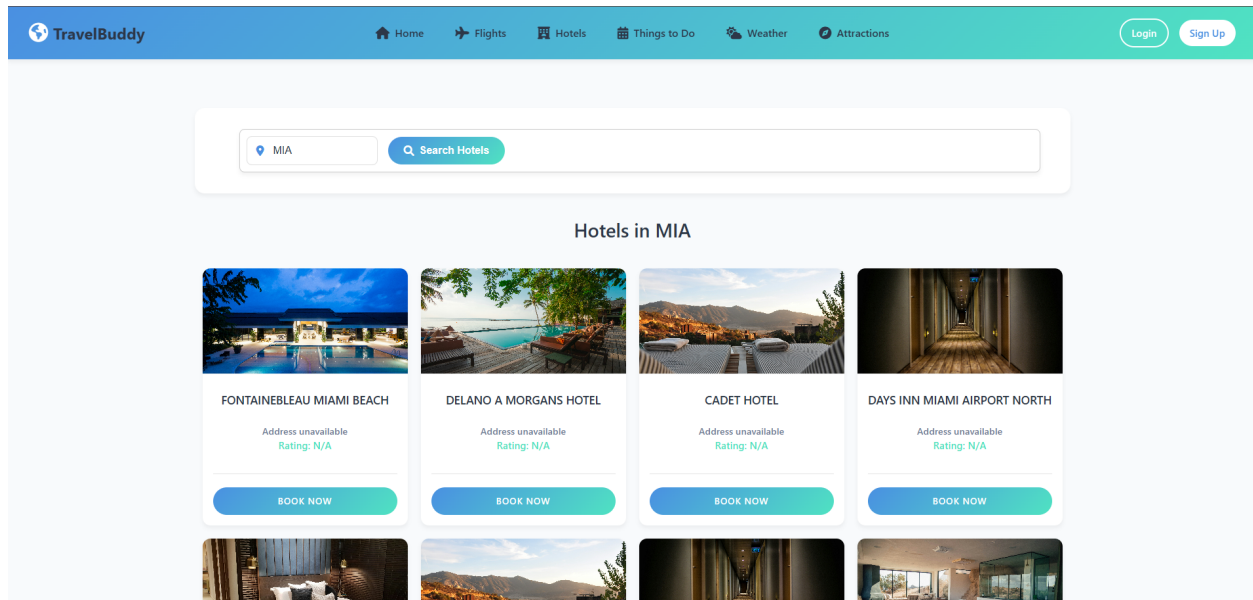


The image shows the Expedia "Search flights" form. The form is located below the Expedia navigation bar, which includes links for "Get the app", "USD", "List your property", "Support", "Trips", and "Nelson Blue tier". The form has a light blue header with the title "Search flights". Below the header, there are four tabs: "Roundtrip", "One-way", "Multi-city", and "Economy". The "Roundtrip" tab is selected. Below the tabs, there are four input fields: "Leaving from" (with a location pin icon), "Going to" (with a location pin icon), "Dates" (with a calendar icon), and "Travelers" (with a person icon). The "Leaving from" field is pre-filled with "Miami, FL (MIA-Miami Intl.)". The "Going to" field is empty. The "Dates" field is pre-filled with "Apr 29 - May 6". The "Travelers" field is pre-filled with "1 traveler". To the right of these fields is a "Search" button. Below the form, there are three sections: "Featured airlines" (with logos for American Airlines, United, Southwest, Delta, Spirit, JetBlue, Frontier, Air India, and Hawaiian Airlines), "Add Price Drop Protection" (with a checkmark icon), "As a One Key member" (with a key icon), and "Save up to 30% when you add a hotel" (with a tag icon). At the bottom, there is a section titled "Popular flight destinations" with a row of six destination cards: "Los Angeles Flights", "Las Vegas Flights", "Cancun Flights", "Miami Flights", "Honolulu Flights", and "Orlando Flights".

A3. Hotel Search Interface

Description:

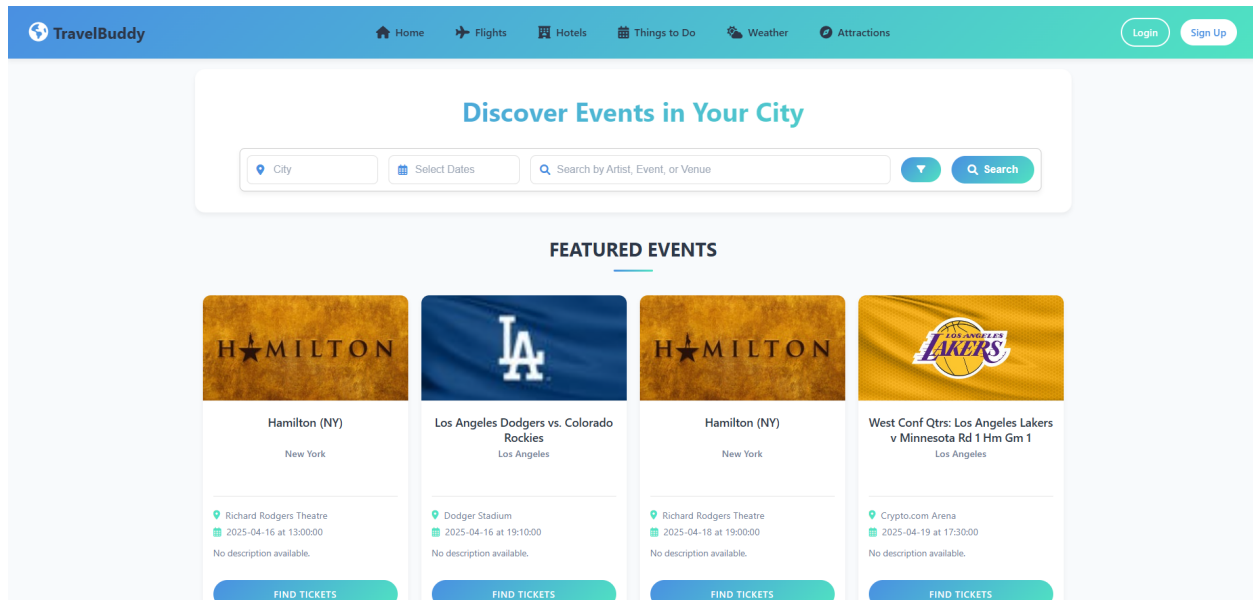
Hotel results are displayed as responsive cards with the hotel name, image, rating, and “Book Now” link. Users input a city code to retrieve location-specific accommodations.



A4. Things To Do – Events Filter

Description:

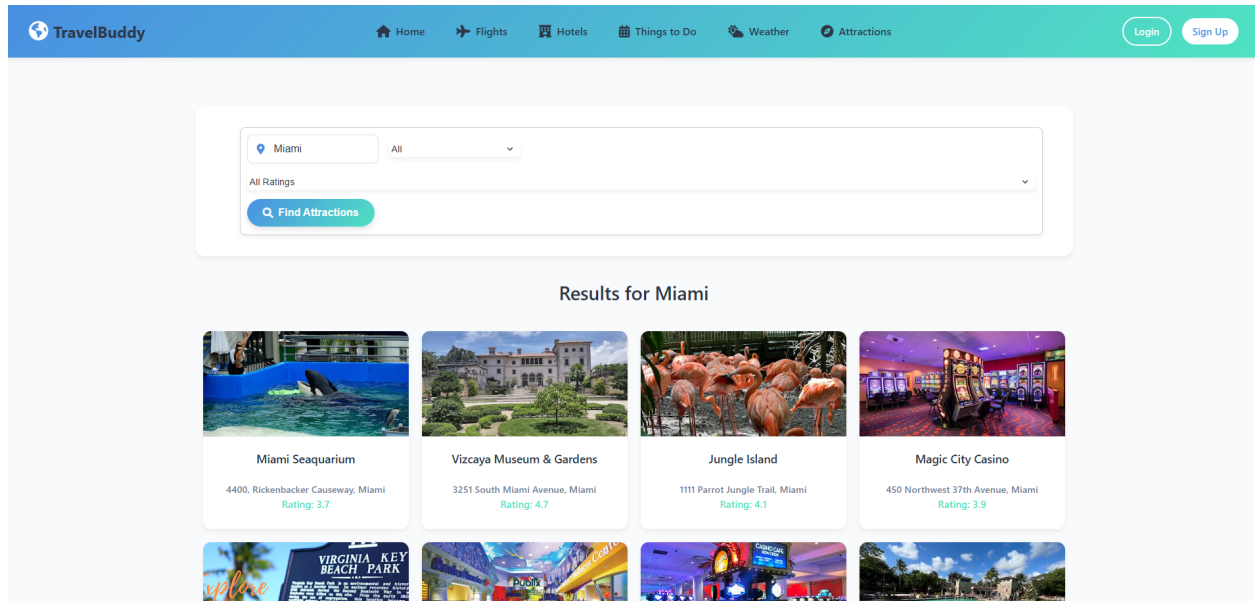
The “Things To Do” page enables users to find upcoming events by filtering based on location, date range, and event category. Each card includes ticket links and detailed event info.



A5. Attractions Finder

Description:

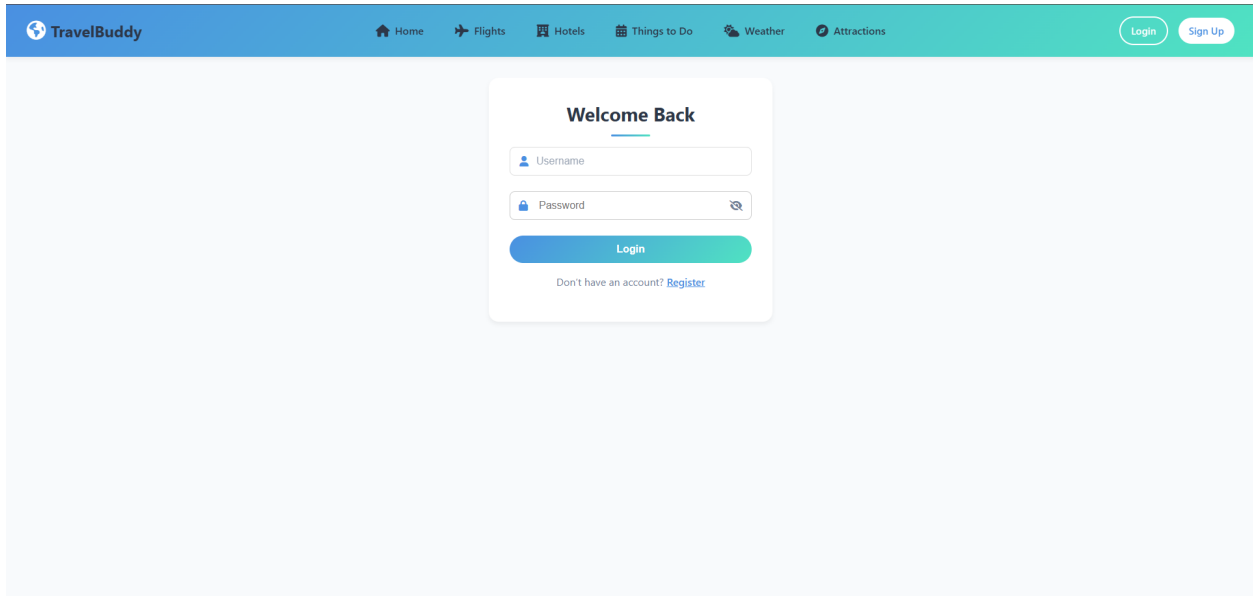
Users can explore local attractions such as parks, museums, and galleries by selecting filters. Each attraction card displays a photo, name, rating, and location, fetched dynamically via the Google Places API.



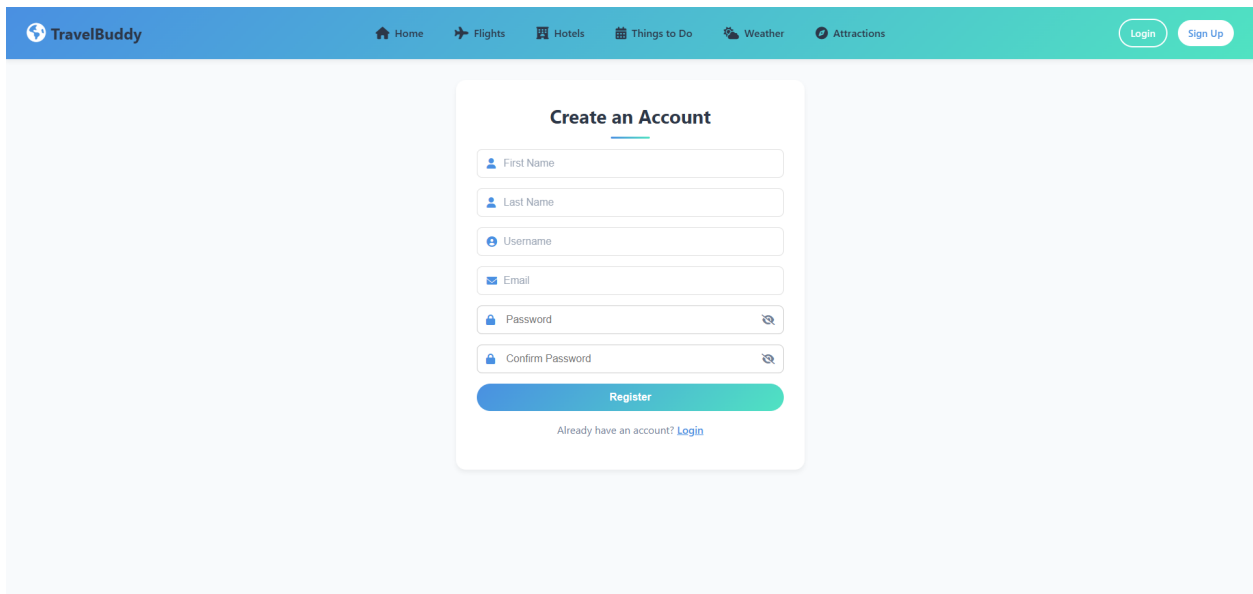
A7. Login / Register Pages

Description:

The login and register pages use form validation, session handling, and responsive design. Once logged in, the navbar updates to reflect user status.



The screenshot shows the 'Welcome Back' login page for TravelBuddy. The header is a blue bar with the TravelBuddy logo and navigation links: Home, Flights, Hotels, Things to Do, Weather, and Attractions. On the right side of the header are 'Login' and 'Sign Up' buttons. The main content area is light blue and features a white login form. The form has a title 'Welcome Back', a 'Username' field with a user icon, a 'Password' field with a lock icon and a toggle for visibility, a 'Login' button, and a link to 'Register' for users who don't have an account.



The screenshot shows the 'Create an Account' register page for TravelBuddy. The header is identical to the login page. The main content area is light blue and features a white registration form. The form has a title 'Create an Account', fields for 'First Name', 'Last Name', 'Username', 'Email', 'Password', and 'Confirm Password' (each with an icon and a toggle for visibility), a 'Register' button, and a link to 'Login' for users who already have an account.

A8. Poster and Demo Preparation

Description:

Final project deliverables include a poster summarizing the system and a demonstration video walkthrough, both prepared during Sprint 7.



School of Computing & Information Science

Spring 2025 Senior Design Project

Travelbuddy

Students: Alejandro Morales, Haadiya Khan, Elena Sotolongo, Gabriel Rodriguez, Argelio Rodriguez
Instructor/Faculty: *Dr. Masoud Sadjadi*, Florida International University

PROBLEM

In today's digital age, planning a trip often involves juggling multiple platforms one for booking flights, another for accommodations, a third for finding local attractions, and yet another for events and weather updates. This fragmented experience can be overwhelming, time-consuming, and inefficient for travelers.

Most users are forced to:

- Navigate several unrelated websites to gather basic trip information.
- Manually compare prices and availability across different platforms.
- Miss out on local events or hidden attractions due to lack of centralized data.

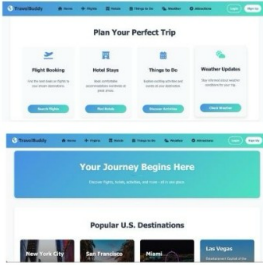
This disjointed approach increases planning fatigue and can lead to missed opportunities, especially for spontaneous or last-minute travelers. Furthermore, many travel apps focus on just one aspect of the journey leaving users to fill in the gaps themselves. Our goal is to change that.

The Travelbuddy App was designed to simplify the travel experience by consolidating essential services flights, accommodations, events, attractions, and weather into one intuitive and user-friendly platform. By streamlining the planning process, users can focus more on the excitement of travel and less on the logistics.

ACKNOWLEDGEMENT

This poster presented in this poster is based upon the work supported by Florida International University. We thank Dr. Masoud Sadjadi for his assistance and mentorship that we received throughout the senior design project.

CURRENT SYSTEM



Efficiently connects users to real-time hotel data and third-party booking links

FEATURES IMPLEMENTED

- ✈️ **Flights** – Users input origin, destination, and date; redirected to Expedia with search pre-filled.
- 🏨 **Hotels** – Hotel listings via Amadeus API with Booking.com redirection.
- 🎪 **Events** – Integration with Eventbrite and Ticketmaster APIs, filterable by city, type, and date.
- 📍 **Attractions** – Google Places integration for points of interest by category and rating.
- 🔑 **Authentication** – User registration, login, email verification via Flask-Mail.
- 🌤️ **Weather** – (In Progress) Planned integration with Tomorrow.io for live weather and forecasts.

SYSTEM DESIGN / ARCHITECTURE

- Flask app with modular structure: app.py routes UI, logic separated into helper functions.
- Environment variables and api_keys.json for secure API access.
- Templates rendered using Jinja2 (HTML/CSS).
- SQLAlchemy models handle database interaction.
- Caching to improve performance in location and event searches.



FLORIDA INTERNATIONAL UNIVERSITY