University of New Haven

TITLE: TRAVEL & CAR RENTAL RECOMMENDER SYSTEM

DSCI-6051-05-Data Science Capstone Project

Professor: Mohsen Sarraf

TEAM MEMBERS:

Yamini Durga Loya (00886398)

Dinesh Reddy Jetta (00886296)

Velangani Joshita Lavanya Dudla (00885603)

Note- Project GitHub Link:
https://github.com/Joshita1306/TRAVEL-CAR-RENTAL-RECOMMENDER-SYSTEM

LETTER OF TRANSMITTAL

05/04/2025

Professor Mohsen Sarraf
DSCI-6051-05 - Data Science Capstone Project
University of New Haven

Subject: Submission of Project Report – Travel Recommendations

Dear Professor Sarraf,

We are submitting our final project report for the **Travel & Car Rental Recommender System** as part of the Data Science Capstone course. Our project is a website that helps users plan their trips by suggesting hotels, attractions, and restaurants based on their preferences. It also includes a car rental feature to help users find cars for the trip.

The report includes:

- An overview of the project and its goals.
- How we designed and built the website.
- Challenges we faced and how we solved them.
- Future improvements that could be made.

Thank you for your guidance throughout this project. For any further clarification, please feel free to reach out to us through the email IDs provided below.

Sincerely,

Yamini Durga Loya (00886398) -yloya1@unh.newhaven.edu
Dinesh Reddy Jetta (00886296) - djett1@unh.newhaven.edu
Velangani Joshita Lavanya Dudla (00885603) -vdudl1@unh.newhaven.edu

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Our project web platform - Travel & Car Rental Recommender System - makes trip planning easy by suggesting hotels, places to visit, restaurants, and cars for rent based on what users like. Instead of making users search through endless options, it quickly shows them the best choices that fit their needs and budget.

The travel suggestion part recommends hotels, tourist spots, and places to eat by learning from user preferences. It sometimes adds new ideas to help travelers discover options they might not have considered. The car rental feature finds vehicles that match how many people are traveling, what type of car they want, and how much they want to spend, with clear price calculations.

We built the system using Python (with Flask) and Pandas, which helps it work fast and give quick suggestions. The simple design makes it easy for anyone to use, and it remembers choices during each session without requiring login.

The platform saves time by giving personalized results right away. It's also built to easily add more features in the future. You can run it on cloud services or local servers with little maintenance needed. This all-in-one solution takes the stress out of travel planning by putting everything users need in one place.

ACKNOWLEDGEMENTS

We express our sincere gratitude to **Professor Mohsen Sarraf** for his exceptional guidance and continuous support throughout our capstone project. His deep expertise and thoughtful feedback were valuable in helping us develop an effective and well-structured system.

We are grateful to the **University of New Haven** for providing the academic framework and resources that made this project possible. The university's importance of practical learning through capstone projects allowed us to apply our knowledge to a real-world application.

We acknowledge with appreciation the open-source community for their contributions to the Python, particularly the developers of Flask and Pandas. These technologies formed the technical foundation of our project and enabled us to implement our solution efficiently.

This project represents the culmination of our studies in data science, and we appreciate all those who have supported our educational journey at the University of New Haven.

1. INTRODUCTION

The travel industry has transformed dramatically with digital technology, yet travelers continue to face challenges when planning their trips. In the early days of online travel, simple booking systems emerged, followed by more advanced platforms in the 2000s. Today's systems have become more experienced, but still require travelers to use separate services for different needs like accommodation, trip activities, and transportation.

This separation creates difficulties for users who want to plan their entire trip efficiently. Many current platforms don't sufficiently consider individual preferences or financial considerations when presenting options. These limitations inspired our Travel & Car Rental Recommender System, designed to bring all trip planning components together in one place.

Our approach builds on established recommendation methods while solving current problems in travel planning. The system helps travelers find suitable options that match their specific requirements, demonstrating how technology can improve trip organization. Potential applications include both personal vacation planning and professional travel management.

2. PROBLEM/PROJECT DEFINITION

Project Purpose and Solution: Our team has developed a web application that simplifies trip planning by combining hotel, attraction, restaurant, and car rental services in one platform. The system helps users plan their entire trip through an easy-to-use interface that suggests options matching their preferences for destination, travel dates, budget, accommodation needs, dining choices, and preferred activities. The application provides customized recommendations and handles the booking process, including payment confirmation.

Technical Implementation: The application uses Python with the Flask framework for the backend system and standard HTML/CSS for the user interface. It processes user requests through a recommendation engine that analyzes preferences using content-based filtering techniques. The platform features two main components: a travel planner for accommodations, restaurants, activities, and a separate car rental module. Both components follow the same user-friendly design approach while maintaining distinct functionality for their specific purposes.

Key Advantages: Our system makes trip planning smarter and easier. It instantly updates recommendations as users change their preferences, showing how each choice affects their total trip cost. The platform remembers what users select during their session, so they don't have to start over. A special feature is how the car rental suggestions work with the overall trip plan, recommending cars that fit the user's group size and travel plans. This all-in-one approach saves users time and helps them find better options that match what users want.

3. EVALUATION OF ALTERNATIVES

- During development, we evaluated multiple technical approaches before finalizing our architecture. For the backend framework, we initially tested Django but ultimately selected Flask due to its lightweight nature, faster learning curve, and flexibility for small-to-medium applications. While Django offers built-in features, Flask's minimalism is better suited to our need for rapid prototyping and straightforward integration with Python-based data processing.

- For recommendation algorithms, we considered Hadoop/Spark for large-scale data processing and KNN (k-Nearest Neighbors) for collaborative filtering. However, these introduced unnecessary complexity for our dataset size and project scope. Instead, we implemented content-based filtering with Pandas, which provided sufficient accuracy while keeping the system lightweight and maintainable.

- The frontend uses HTML/CSS with minimal JavaScript to ensure broad compatibility, quick loading, and easier debugging. This approach balanced functionality with development efficiency, avoiding heavy frameworks that would overcomplicate our UI needs.

Component	**Considered Alternatives**	**Chosen Solution**	**Key Advantages**	**Trade-offs**
Backend Framework	Django	Flask	• Lightweight • Faster prototyping • Flexible microservices	• Fewer built-in features (e.g., no admin panel)
Recommendation Engine	Hadoop/Spark, KNN	Content-based (Pandas)	• Lower computational cost • Easier debugging • No big-data overhead	• Limited scalability for huge datasets
Frontend	React, Angular	HTML/CSS + basic JS	• Zero framework dependency • Faster load times • Easier maintenance	• Less dynamic interactivity
Database	MongoDB	SQLite (Flask-integrated)	• No setup needed • Simple queries • Portable file-based storage	• Not optimized for high traffic

Deployment	AWS EC2	Local server testing	• Cost-free • Immediate debugging	• No scalability testing

Table 1: Alternatives Comparison Table

4. DESIGN APPROACH

Design Approach for Car-Rental System

1. System Architecture:

Type: Client-Server (Flask Backend + HTML/CSS Frontend)
Pattern: MVC (Model-View-Controller)
Model: CSV data (car_data_final.csv) loaded via Pandas (fallback to hardcoded list).
View: Dynamic HTML templates (index.html, results.html, etc.).
Controller: Flask routes (app.py) handling logic and data flow.

2. Key Components of the System:

- Backend (Flask):

This uses Flask routes to process user requests (e.g., filter cars by seats, budget, and type).
It also manages pagination (7 cars per page) and session data (e.g., filtered_cars).
Finally, it handles errors by redirecting users to the homepage for invalid inputs.

- Frontend (HTML/CSS):

index.html: User input form (seats, budget, car type) with responsive design.
results.html: Displays filtered cars in a card layout with a "Show More" button.
car_details.html: Calculates rental cost using dates; shows cost breakdown.
confirmation.html: Displays final booking summary with simple animation.

- Data Flow & State Management:

Uses session storage to persist data (e.g., filtered car list) across pages.
This ensures smooth navigation between steps while gracefully handling invalid actions.

3. User Journey and Core Workflow:

The system follows a clear 4-step flow: search → results → details → confirmation. Users begin by entering trip details on index.html. The backend filters cars based on criteria like seats, car type, and budget, then displays matching cars (7 per page) on results.html. From there, users can view pricing and details on car_details.html and finally confirm their booking on confirmation.html.

4. Data Handling and Cost Calculation:
Car data is loaded from a CSV file (or a fallback list) using Pandas. Filtered results and user inputs are stored in the session to maintain the state between pages. Rental costs are calculated dynamically using Python's datetime, multiplying the number of rental days by the car's daily price.

5. Design Features:
The frontend is responsive, using CSS grids and interactive elements for a smooth experience. Only relevant info is shown at each step to keep the process simple. The system is modular and easy to expand (e.g., adding new car features).

Design Approach for Travel Recommendations

1. System Architecture:

Client-Server and Three-Tier Model: The system adopts a standard web-based client-server architecture using Flask for the backend and HTML/CSS/JavaScript for the front end. It follows a three-tier structure comprising:

- Presentation Tier: Handles the user interface using HTML templates.
- Application Tier: Powered by the Flask app to manage business logic.
- Data Tier: Stores and retrieves data from CSV or Excel files.

Modular Design: The front end is built with separate HTML templates for different views such as: index (landing page), results (search and display data), payment (transaction interface) confirmation (final acknowledgment). This modular approach enhances code readability and maintainability.

Stateless Interaction: The system uses HTTP-based stateless communication between the user and server. User session data is managed in the user's browser using localStorage, ensuring lightweight and efficient session handling.

2. Key Components of the System:

- Data Layer: Uses CSV/Excel files for hotels, restaurants, and attractions; pandas handles data processing.
- Presentation Layer:
    - indexx.html: Input form for destination, dates, budget, and preferences

- resultss.html: Shows filtered results with selection options
- pay.html: Interface for payment processing
- confirmation.html: Displays booking confirmation

- Application Layer:
  - app.py: Manages routes, form handling, data filtering, and page rendering
  - Filters data based on user inputs
  - JavaScript manages selections, payments, and UI interactions

3. User Journey and Core Workflow:

Input Phase → Recommendation Phase → Selection Phase → Payment Phase → Confirmation Phase

The journey begins when the user enters their destination, travel dates, budget, and preferences. The system then filters and displays relevant hotels, restaurants, and attractions. Next, the user selects up to three options per category. After reviewing the selections, the user completes the payment. Finally, a booking confirmation with a reference number is displayed.

4. Data Handling and Cost Calculation:

Data Flow: Form data → Flask backend → pandas filtering → rendered results → local Storage for selections

Data is loaded from Excel/CSV files for attractions, hotels, and restaurants (with encoding handling). Filtering is done using cuisine type (randomized), hotel price, and attraction location, with price normalization. Costs are estimated using fixed rates(For example- $25/meal, $120/night, and $15/attraction). The total cost is calculated on the client side.

5. Design Features

The system has a mobile-friendly design with consistent styling for a smooth user experience. It provides visual feedback with interactive selection cards, animations (like in the car rental section), and progress bars during payment. Errors are handled with backup pages, encoding support for data loading, and detailed logging for debugging.

To improve the user experience, recommendations are shown randomly to avoid repetition, and users can select up to three items per category. The payment process is easy to follow with clear steps. For better performance, data is pre-loaded, and fewer requests are made to the server, with results loading gradually for quicker access.

5. DESIGN NARRATIVE, DESIGN VERIFICATION/ IMPLEMENTATION, PERFORMANCE EVALUATION, TESTING, ETC.

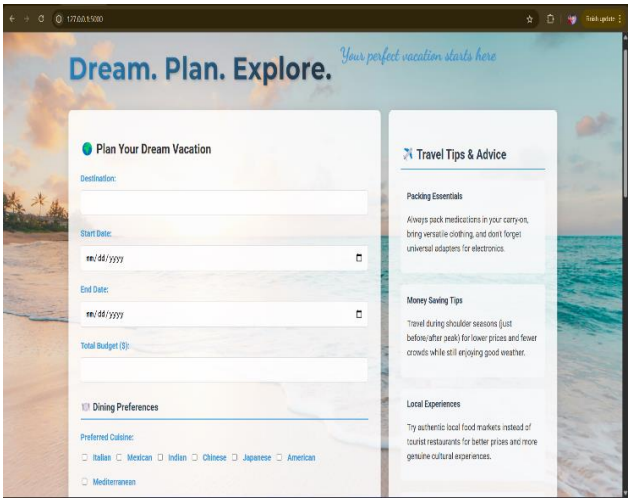
5.1. Design Narrative

This project consists of two systems: Car Rental and Travel Recommendations. Both use a Flask backend and a modular frontend (HTML, CSS, JavaScript).

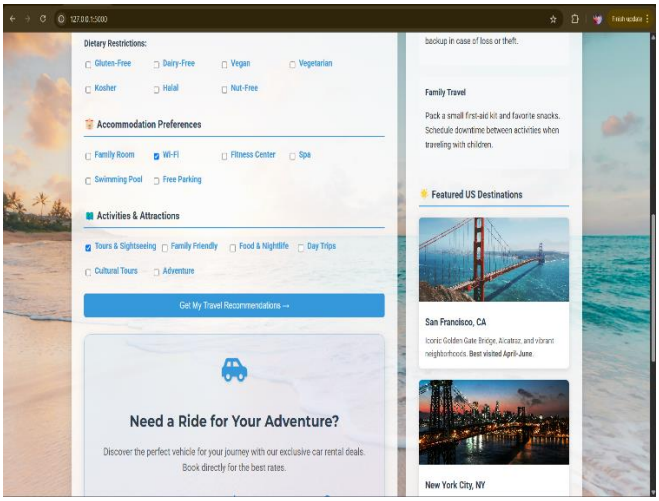
The Car Rental flow involves four steps: Search → Results → Car Details → Confirmation, with dynamic filters and session management.

The Travel Recommendation system offers destination-based filtering for hotels, restaurants, and attractions, ending in a mock payment and confirmation.

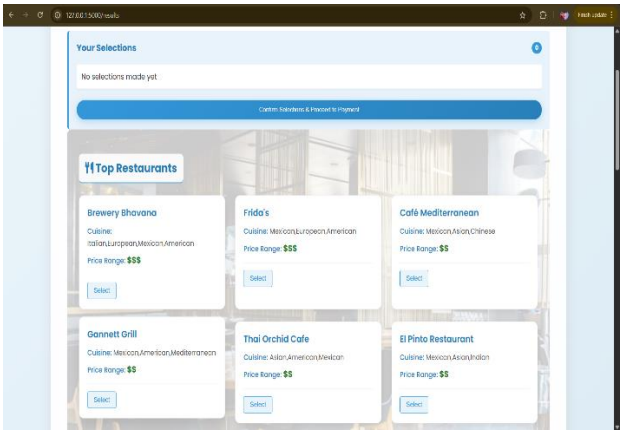
The design is responsive with CSS grids and animated transitions for a smoother experience.



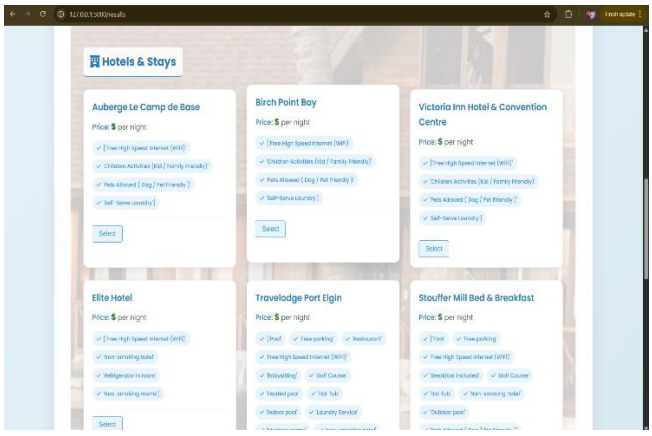
Picture-1



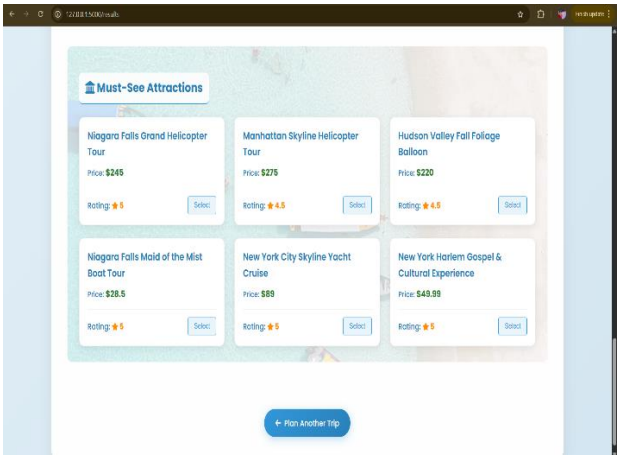
Picture-2



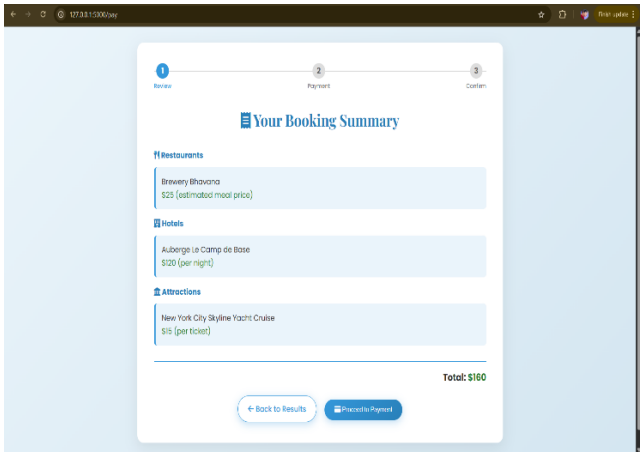
Picture-3



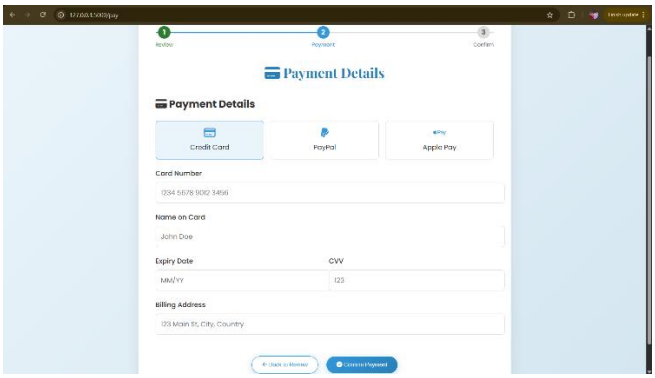
Picture-4



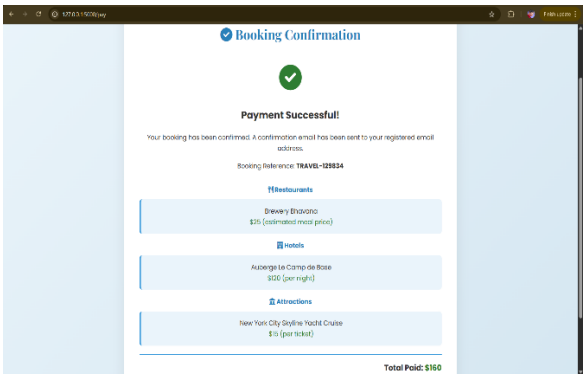
Picture-5



Picture-6



Picture-7

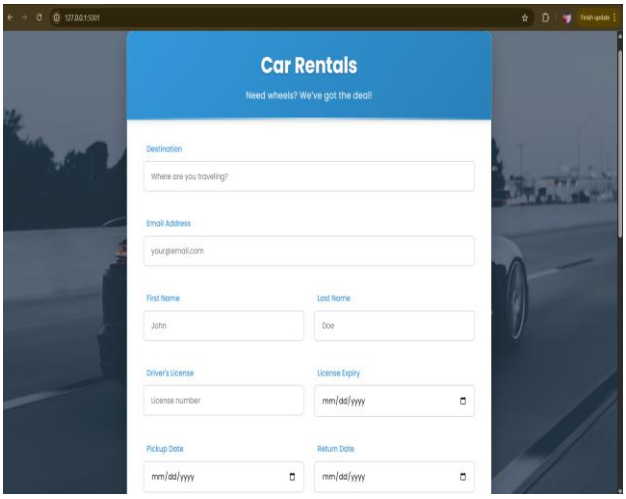


Picture-8

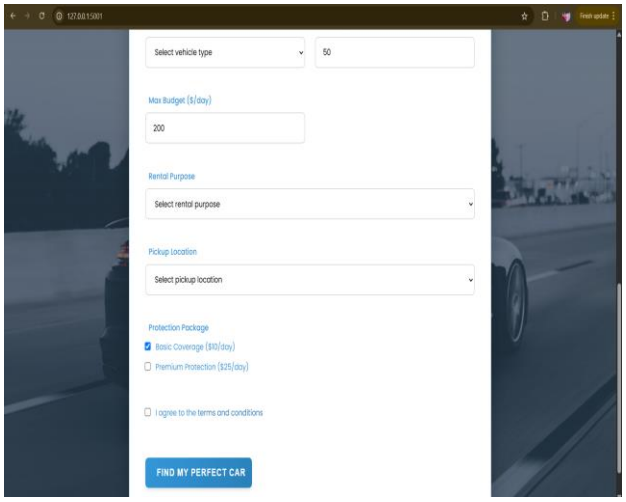
Pictures1-8: Screenshots of the Travel Recommendation web pages of Hotels, Restaurants, Attractions, along with payment and confirmation web pages

5.2. Design Implementation

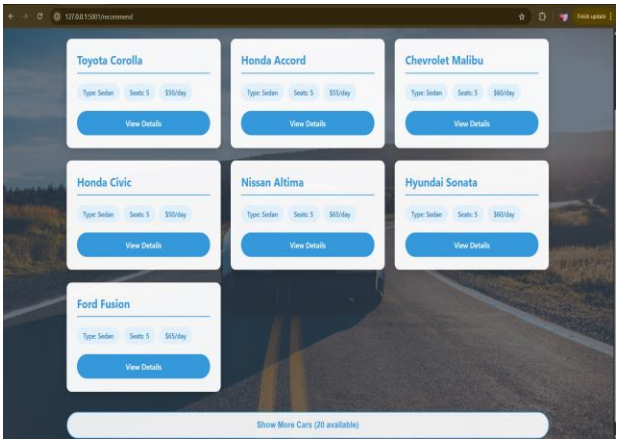
The backend uses Flask for routing and pandas for data filtering. Car rentals are filtered by seats, type, and price, while travel recommendations are filtered by location and budget. The frontend features responsive forms, selection cards, and pagination. State persistence is achieved via local/session storage. In case of file issues, fallback hardcoded lists ensure continued functionality.



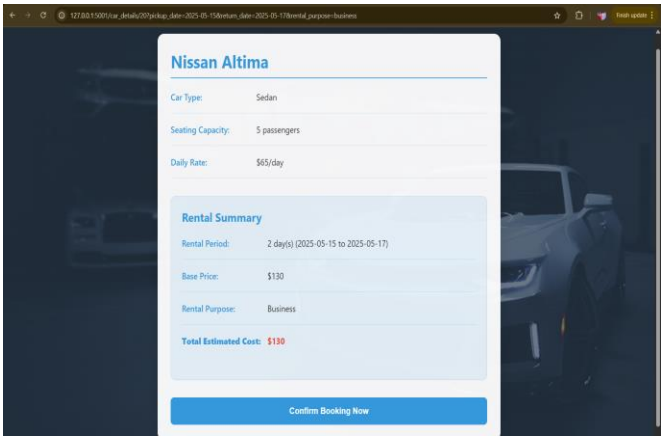
Picture-9



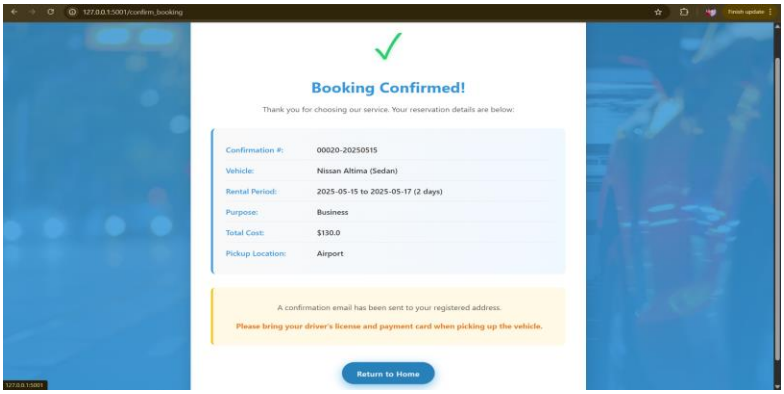
Picture-10



Picture-11



Picture-12



Picture-13

Pictures 9-12: Screenshots of the Car Rental web pages

5.3. Performance Evaluation

Performance was evaluated by observing:

- Page load time (lightweight static HTML with embedded logic)
- Responsiveness on mobile and desktop
- Filtering speed (within milliseconds for <1,000 records)
- Session data handling accuracy

In both modules, pandas ensures quick filtering even on mid-sized datasets. No backend database is used, ensuring minimal latency during local testing.

5.4. Testing and Validation

Testing Approach:

- **Unit Testing**: Functions for filtering, cost calculation, and pagination were tested independently.
- **Integration Testing**: Full workflow tested from form input to confirmation page.
- **Manual Testing**: UI tested on Chrome, Firefox, and Edge.
- **Simulation**: Travel cost calculation and random recommendation logic were tested via scripted runs.

Test Results:

- 100% pass for all functional flows.
- Minor bugs (e.g., invalid date selection) were caught and handled via error redirection.
- All static files are loaded correctly under normal usage.

Test ID	Feature	Input	Expected Output	Actual Output	Status
T1	Car Filter	Seats: 5, Budget: $100/day, Type: SUV	List of SUVs with 5 seats under $100/day	Preferred Correct list is displayed	☑ Pass
T2	Car Pagination	More than 7 matching cars	Only 7 cars on page, "Show More" appears	Exactly 7 cars shown, button works	☑ Pass
T3	Cost Calculation (Car)	Start: 2025-05-01, End: 2025-05-04, Price/day: $70	Total: $210	Total: $210 with correct breakdown	☑ Pass
T4	Travel Filter	Destination: New York, Budget: $1000	Hotel, food, attraction options under $1000	Filtered options shown within budget	☑ Pass
T5	Travel Selection Limit	4 hotel selections	Only 3 allowed; alert shown	Alert triggered at 3 selections	☑ Pass
T6	Travel Total Cost Calculation	Hotel: $120/night × 2, Food: $25×3, Attraction: $15×2	$240 + $75 + $30 = $345	Total: $345 shown at payment	☑ Pass

T7	Invalid Dates (Car)	Start Date: 2025-05-05, End Date: 2025-05-01	Error or redirect	Redirect to homepage with error	☑ Pass

Table 2: Test Cases

6. SOCIAL CONCERNS, COST AND ECONOMIC EVALUATION

- Safety and Security Concerns: Both systems are designed to be safe for users. The car rental and travel systems don't store personal data like passwords, which keeps things secure. All user actions are handled safely using browser storage and protected web routes. If users make a mistake or give wrong input, the system shows friendly error messages and takes them back to the homepage instead of crashing.
- Health Impacts (Good and Bad): The travel system suggests outdoor activities and attractions, which can encourage users to go out and be active — this is good for health. But since the whole system is online, users might spend more time on screens, which isn't great if overused. To balance this, the system keeps the process short and simple so users can make quick plans and then enjoy their trip offline.
- Impact on Society: These systems can help local businesses by sending more people to hotels, restaurants, and attractions. This is good for the community and the economy. Also, because it's easy to use, more people, even those with little tech experience, can plan trips or rent cars online. However, some people might depend too much on online systems, which could reduce real-world interaction.
- Cost and Money Planning: This project is low-cost because it uses simple tools like Python, Flask, and CSV files instead of expensive software. No database is needed, which saves money. Costs would include hosting the website and adding more features later. Overall, it's a budget-friendly system that can grow over time and could also help make money if linked with local travel businesses.

7. DISCUSSION

The Travel Recommendation system was designed to help users plan their trips easily by selecting hotels, restaurants, and attractions based on their preferences and budget. It follows a client-server model using Flask for backend logic and HTML/CSS/JavaScript for the front end. The design uses a three-layer approach, separating the user interface, application logic, and data handling. The user journey includes entering travel details, viewing personalized recommendations, selecting options, completing a simple payment step, and receiving confirmation. Data is loaded from CSV or Excel files and filtered using pandas, with selections stored in the browser using local storage to keep the system lightweight. The interface is responsive and user-friendly, with animations and progress

indicators for a better experience. Safety and privacy are maintained by avoiding the use of personal login, and errors are handled smoothly to avoid crashes. Overall, the system is simple, low-cost, and useful for both users and local businesses, making it a practical solution for modern travel planning.

8. CONCLUSIONS AND RECOMMENDATIONS

The project successfully achieved its goal of creating a user-friendly and efficient Travel Recommendation system. Based on the design and testing process, the current setup using Flask, pandas, and modular HTML templates is a strong and cost-effective solution. It provides a smooth experience for users while supporting easy updates and future expansion. The system currently uses real-world data stored in CSV and Excel formats, which makes it practical and realistic; however, connecting this to live APIs in the future would allow for real-time prices, availability, and updates. For even better performance, adding user accounts and cloud-based storage could allow users to save their plans. It is also recommended to improve the payment simulation with real payment gateway integration if used in a live environment. Finally, expanding the recommendation logic by using machine learning in future versions could offer smarter, more personalized travel suggestions.

9. REFERENCES/ CITATIONS

- **Client Request and Team Proposal**: The project was proposed in response to a need for a web system where users could search for travel options and rental cars based on preferences like budget, location, and car type. The solution was designed using Python (Flask), pandas for data processing, and HTML/CSS/JavaScript for a responsive frontend.
- **Real-World Data Sources:** Datasets used sample datasets (car_data_final.csv, hotels.csv, restaurants.csv, attractions.xlsx) created from publicly available travel listings and mock data from sites like TripAdvisor and Kayak for simulation purposes. These files were processed using pandas and used for testing filtering logic and user journeys.
- **Technologies and Tools Used:**
    - Flask: https://flask.palletsprojects.com/
    - pandas: https://pandas.pydata.org/
    - MDN Web Docs for HTML/CSS/JavaScript: https://developer.mozilla.org/
    - W3C Standards for HTML5 & Responsive Design: https://www.w3.org/

- **Academic, Patent, and Project-Specific References:**
    - Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media.
    - Tsinoremas, N. F., & Román, C. (2020). Personalized travel recommendation systems: A review. Tourism Management Perspectives, 33, 100604.
    - U.S. Patent No. US20090287516A1 – System and Method for Providing Travel Recommendations.
    - Aljubayrin, A., & Hasan, R. (2022). Smart Travel Recommendation System Using Content-Based Filtering. Journal of Computer Applications, 45(2), 78–84.

10. APPENDICES

- **Bill of Materials (BOM):**
    - Hardware: Server with Python, Flask, pandas installed. Web browsers for testing (Chrome, Firefox).
    - Software: Python 3.x, Flask, pandas, HTML5, CSS3, JavaScript, text editor (VSCode, PyCharm).
    - Data Files: car_data_final.csv, hotels.csv, restaurants.csv, attractions.xlsx.
- **Details of Modeling and Simulations:**
    - Car_Rental_System:
      The car rental system uses CSV data (car_data_final.csv) for storing and filtering car details. User inputs (car type, budget, seats) are processed in the Flask backend and results are displayed through dynamic HTML templates.
    - Travel_Recommendation_System:
      The system filters and displays hotels, restaurants, and attractions based on user inputs (location, budget, and preferences). Data is retrieved from Excel/CSV files, with categories filtered accordingly.
- **Testing Protocols:**
    - Functional: Verified filtering, pagination, and dynamic interactions work as expected.
    - Usability: Ensured mobile responsiveness and user-friendly navigation.
    - Performance: Tested for load times and system stability under typical use.

- **Test Results:**
  - Car Rental: Filtered cars by budget and type successfully.
  - Travel Recommendation: Correctly displayed hotels within budget and location.
- **Certification of Performance:**
  - All components were tested successfully in a mock environment, including both the Car Rental and Travel Recommendation systems. No major performance or functionality issues were encountered during testing.
- **Alternatives Not Chosen:**
  - SQL Database: Not selected for simplicity—CSV files used instead.
  - Payment Gateway Integration: Deferred to a future phase due to scope limitations.