Smart Route Planner: A Web-Based ETA and Location Status Tool for City Walkers

I. Introduction & Motivation

In a fast-paced and mobile world, time efficiency and planning accuracy are essential—especially for pedestrians navigating cities for tourism, errands, or daily routines. While many apps provide general navigation, few allow users to customize routes based on individual time allocation and business hours. As a student in Williamsburg, I recognized this gap and developed Smart Route Planner: a browser-based tool that allows users to search for places, mark destinations, assign visit durations, and generate a personalized walking route with estimated arrival times and predicted open/closed status.

Unlike typical routing services, this planner provides manual control over stop order and time allocation through a clean, editable interface. The project combines user-centered design with geospatial intelligence and browser-side automation.

II. Methodology & Technology Stack

The Smart Route Planner was built using a modern, modular tech stack:

- Front-end: HTML, CSS, JavaScript
- APIs: Google Maps JavaScript API, Google Places API, Google Directions API
- Data Storage: Browser-based localStorage for persistent user state
- Export: jsPDF for downloadable PDF reports

Core features include:

- Place search with Google Autocomplete
- Interactive info windows with business hours, ratings, and categories
- Categorized map markers (Food & Drink, Shopping, Services, Things to Do)
- Editable visit durations and manual reordering via drag-and-drop
- Walking route generation using Google Directions
- Arrival time calculation based on user-defined start time and date
- Real-time open/closed status prediction based on weekday_text data
- Save/load/delete named trip plans
- Export plan to PDF with a final estimated end time

III. Demonstration & User Interaction

The app interface consists of two panels:

- A Google Map (left) for visualizing marked locations
- A route planning sidebar (right) featuring an editable table

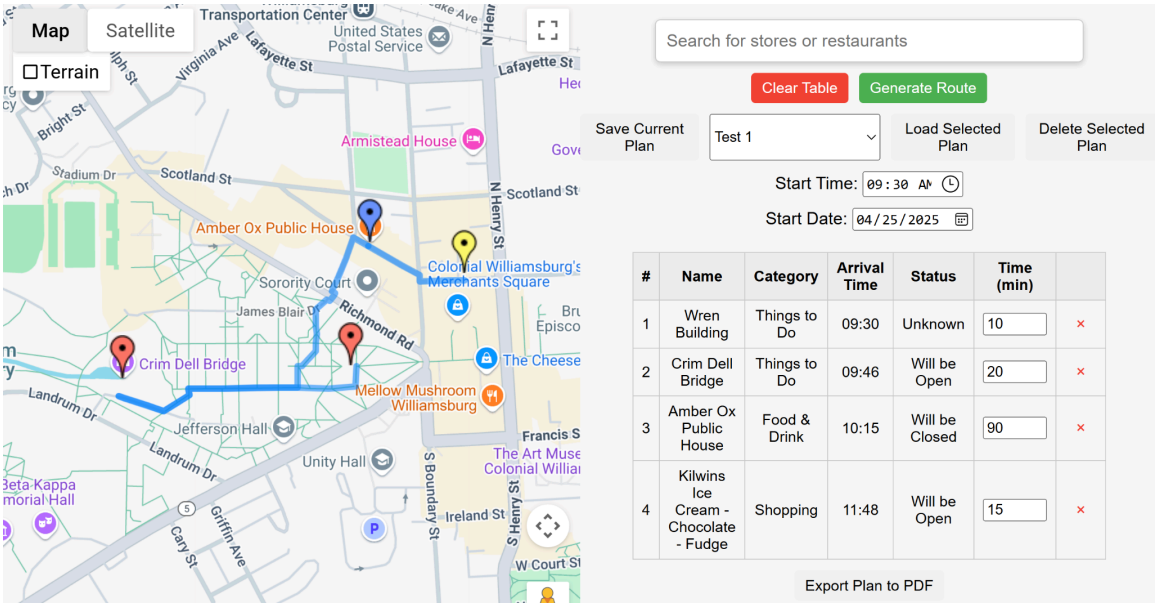
How It Works:

1. Users search for a destination using the input bars
2. An interactive info window appears with key details and a “Mark Location” button.
3. Marked locations are displayed on the map and added to the planning table.
4. Users specify a start date and time, reorder stops, and define how long they will spend at each.
5. The system computes walking times between stops and calculates:
    - Estimated Arrival Time
    - Predicted Open/Closed Status at each stop

Each row in the table displays:

- Stop order
- Name
- Category
- Arrival Time
- Status (Will be Open / Will be Closed / Unknown)
- Editable visit duration
- A delete button

Once multiple stops are added, clicking “Generate Route” calculates walking directions, updates arrival times, and refreshes the open/closed predictions in real time. Users can save their plan for later or export it as a PDF, which includes all trip details and the final estimated trip end time.



#	Name	Category	Arrival Time	Status	Time (min)	
1	Wren Building	Things to Do	09:30	Unknown	10	×
2	Crim Dell Bridge	Things to Do	09:46	Will be Open	20	×
3	Amber Ox Public House	Food & Drink	10:15	Will be Closed	90	×
4	Kilwins Ice Cream - Chocolate - Fudge	Shopping	11:48	Will be Open	15	×

IV. Reflection

This project has been a comprehensive exercise in applied development, greatly expanding both my technical capabilities and my approach to user-focused design. Notable areas of growth include:

- Real-time geographic data processing

Working with live location data through the Google Maps and Places APIs deepened my appreciation for the challenges of geospatial computing. I learned how to extract structured and unstructured data (e.g., place metadata, geometry, and human-readable hours) and convert it into useful, dynamic insights like estimated arrival time and open/closed status. The ability to manipulate geocoordinates, visualize them on a map, and compute distances and durations based on real-world infrastructure gave me a hands-on understanding of location intelligence.

- User-centered design principles

From the earliest stages, I focused on building a user experience that felt intuitive, flexible, and responsive. This meant simplifying inputs, adding color-coded categories for readability, and organizing route data into a clean, reorderable table format. Allowing users to manually assign time and reorder stops mirrors real-world behavior—something missing in most automatic route optimizers. Additionally, features like saving and restoring plans were designed to support users' changing needs throughout the day.

- Asynchronous API handling in JavaScript

Integrating multiple asynchronous services, including search autocomplete, place detail retrieval, and route generation, helped me better understand how to coordinate asynchronous flows in client-side JavaScript. I became more proficient at handling callbacks, managing race conditions, and chaining API requests while maintaining UI responsiveness. This real-time interactivity was critical to enabling smooth marker placement, dynamic table updates, and live ETA calculations.

- Building persistent, interactive client-side web apps

By implementing localStorage, I enabled users to preserve route data across sessions without needing an account or backend. This introduced me to the architecture of lightweight web applications that maintain state while remaining privacy-friendly and fast. I also gained experience separating presentation (CSS), logic (JavaScript), and structure (HTML), reinforcing best practices for scalable front-end development.

V. Future Ideas

To extend Smart Route Planner into a more powerful and comprehensive tool, the following key improvements are envisioned:

1. Multi-Day Trip Planning

Currently, the planner supports a single-day itinerary. A future upgrade will allow users to:

- Create and manage multi-day plans with separate routes for each day.
- Assign destinations to specific dates, view timelines per day, and visually switch between them via tabs or a calendar view.
- Automatically balance trip load by suggesting how to distribute stops across multiple days based on estimated durations and hours of operation.

This will make the app suitable for full travel itineraries or weekend getaways.

2. Real-Time Context Awareness

   To provide smarter, situation-aware routing, the app can be enhanced with:

   - Live closure detection, drawing from APIs like Yelp, Google Business, or city open data portals.
   - Local event alerts to avoid street festivals, road closures, or public gatherings that may affect access or timing.
   - Weather-aware adjustments, such as recommending indoor locations or notifying users about rain forecasts before departure.

   Together, these features will help users avoid unexpected delays and make informed decisions in real time.

3. Smart Recommendations

   Rather than relying solely on user input, future versions will offer intelligent suggestions:

   - Personalized recommendations based on category preferences, time of day, or previous routes.
   - Time allocation guidance, using crowd data or Google Popular Times to suggest how long to spend at each stop.
   - Optimized routing options, giving users the choice between manual ordering or algorithm-driven efficiency based on shortest travel time or maximum venue availability.

   This makes the tool more helpful not just for planning, but also for discovery and optimization.

VI. References

- Google Maps JavaScript API Documentation: https://developers.google.com/maps/documentation/javascript/overview
- Google Places API (weekday_text): https://developers.google.com/maps/documentation/places/web-service/details
- Mozilla Developer Network (MDN): JavaScript Reference

- Stack Overflow (general web guidance)