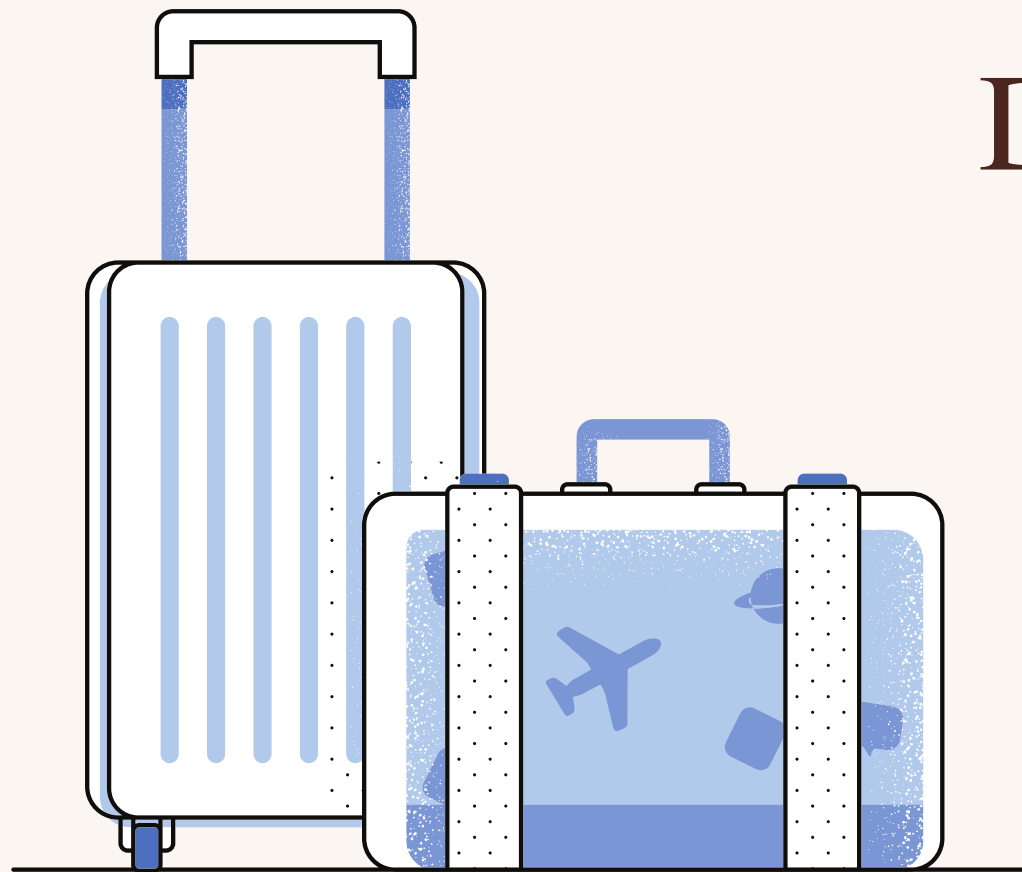


# ECO-FRIENDLY TRAVEL PLANNER

Discovering unexplored corners



# Introduction

The Eco-Friendly Travel Planner is a smart route optimization system developed using Java, Spring Boot, and JavaScript. It computes the most efficient travel routes based on distance and time by implementing Dijkstra's algorithm over a graph modeled with an adjacency list. The planner provides a seamless user experience through a robust backend–frontend integration, enabling users to select optimal paths for sustainable and efficient travel.



# OBJECTIVE

- To help users find the most efficient travel routes based on distance and time, reducing travel effort and confusion.
- To promote eco-friendly travel by minimizing fuel consumption and carbon emissions through optimized routing.
- To apply graph algorithms (Dijkstra's) practically in real-world route planning systems.
- To create a fully functional system with seamless integration of frontend and backend technologies.
- To provide a platform that can be expanded in the future to include scenic routes, toll avoidance, real-time traffic updates, and more.

## Why Choose Eco-Friendly Travel?



### Reduce Carbon Footprint

Our routes are optimized to minimize emissions and environmental impact while still getting you to your destination.



### Support Local Communities

We highlight stops and accommodations that prioritize sustainability and give back to local communities.



### Travel with Purpose

Feel good about your travel choices knowing you're making a positive difference for the planet.

# TECH STACK



- Backend: Java, Spring Boot
- Frontend: JavaScript, HTML, CSS
- Tools: Eclipse IDE, VS Code
- Algorithm: Dijkstra's Algorithm
- Data Structure: Adjacency List (Graph)
- Communication: REST APIs
- Package Manager: Node.js (npm)

The screenshot displays the EcoTravel website's search interface. At the top, there is a navigation bar with links for Home, Destinations, About Us, Blog, and Contact, along with Log In and Sign Up buttons. The main search area is centered and contains the following fields:

- From:** A dropdown menu with a magnifying glass icon and the text "Select City".
- To:** A dropdown menu with a magnifying glass icon and the text "Select City".
- Departure Date:** A date picker with a calendar icon and the text "mm/dd/yyyy".
- Travelers:** A dropdown menu with a group of people icon and the text "1 Traveler".

Below these fields, there is a section titled "Eco-Friendly Options" with six checkboxes:

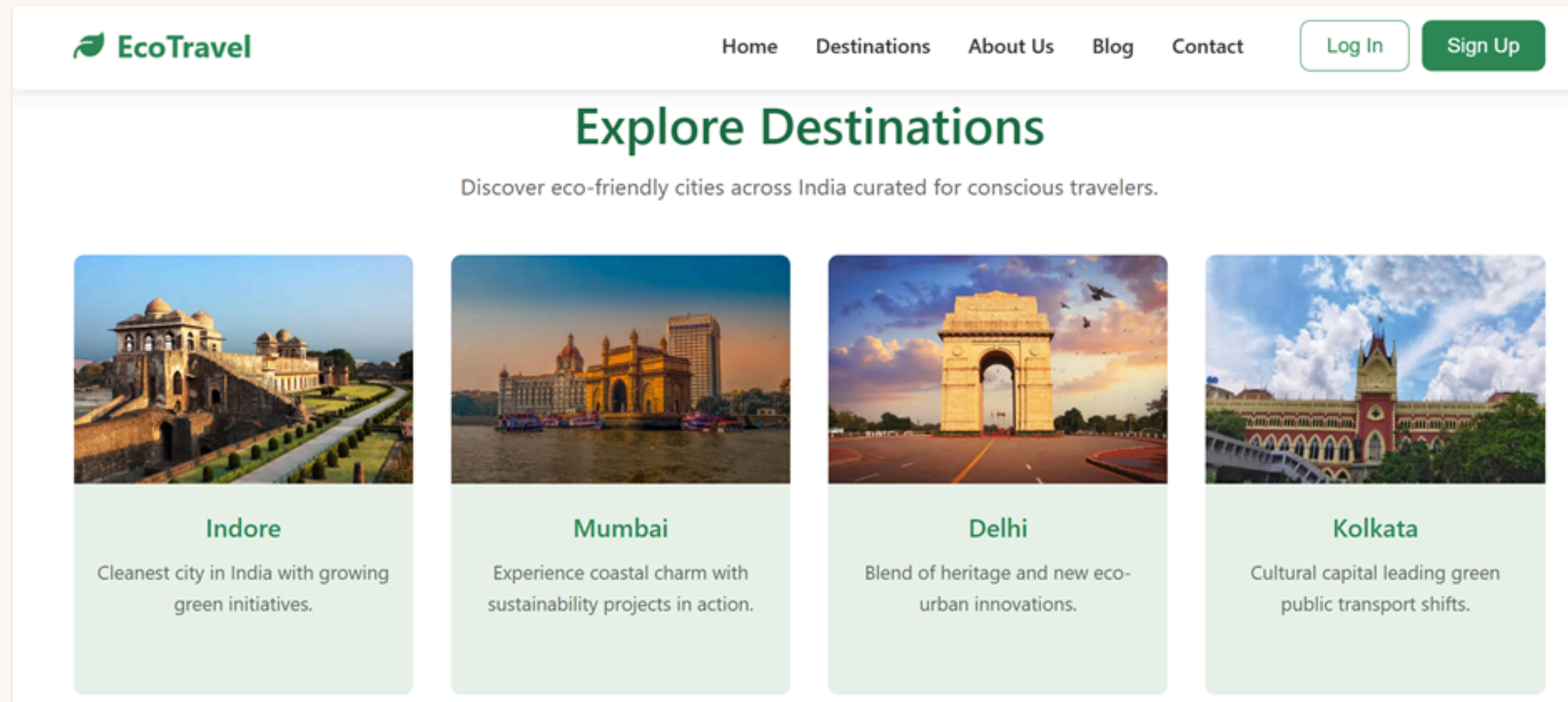
- ☐ eco Route
- ☐ carbon Offset
- ☐ public Transport
- ☐ eco Hotels
- ☐ bike Friendly
- ☐ local Experiences

At the bottom of the search area, there is a green button labeled "Search Eco-Friendly Routes". The entire interface is framed by a light gray border with decorative green vertical bars on the left and right sides.



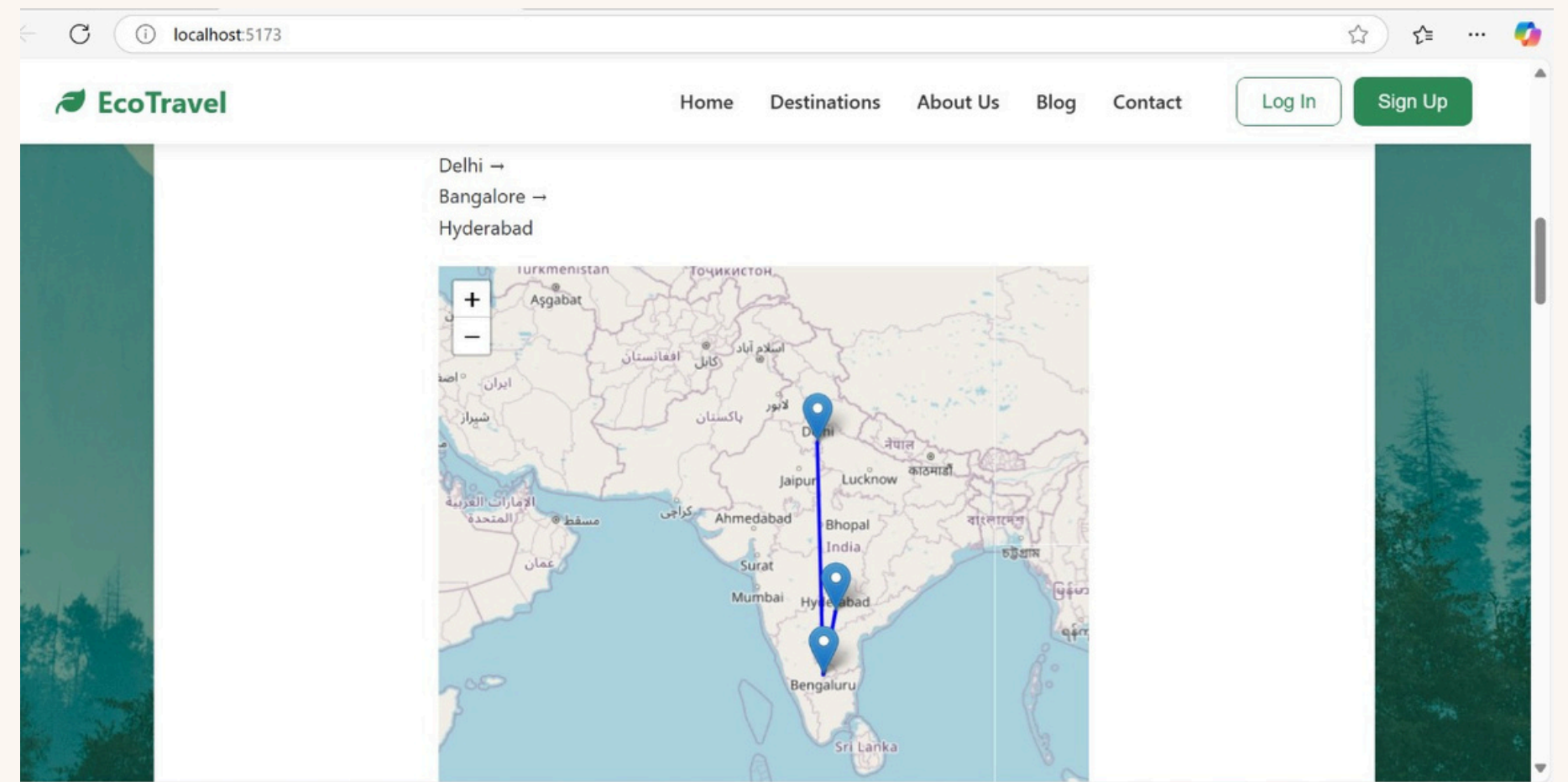
# FEATURES

- **User-Friendly Interface:** Simple and interactive frontend for easy travel planning.
- **Route Optimization:** Calculates the best route based on shortest distance or shortest time.
- **Graph-Based Pathfinding:** Uses Dijkstra's algorithm on a graph modeled with an adjacency list.
- **Customizable Travel Preferences:** Users can select their priority — distance or time optimization.
- **Eco-Friendly Focus:** Reduces unnecessary travel, indirectly supporting environmental sustainability.
- **Cross-Platform Development:** Developed using popular tools like Eclipse and VS Code.



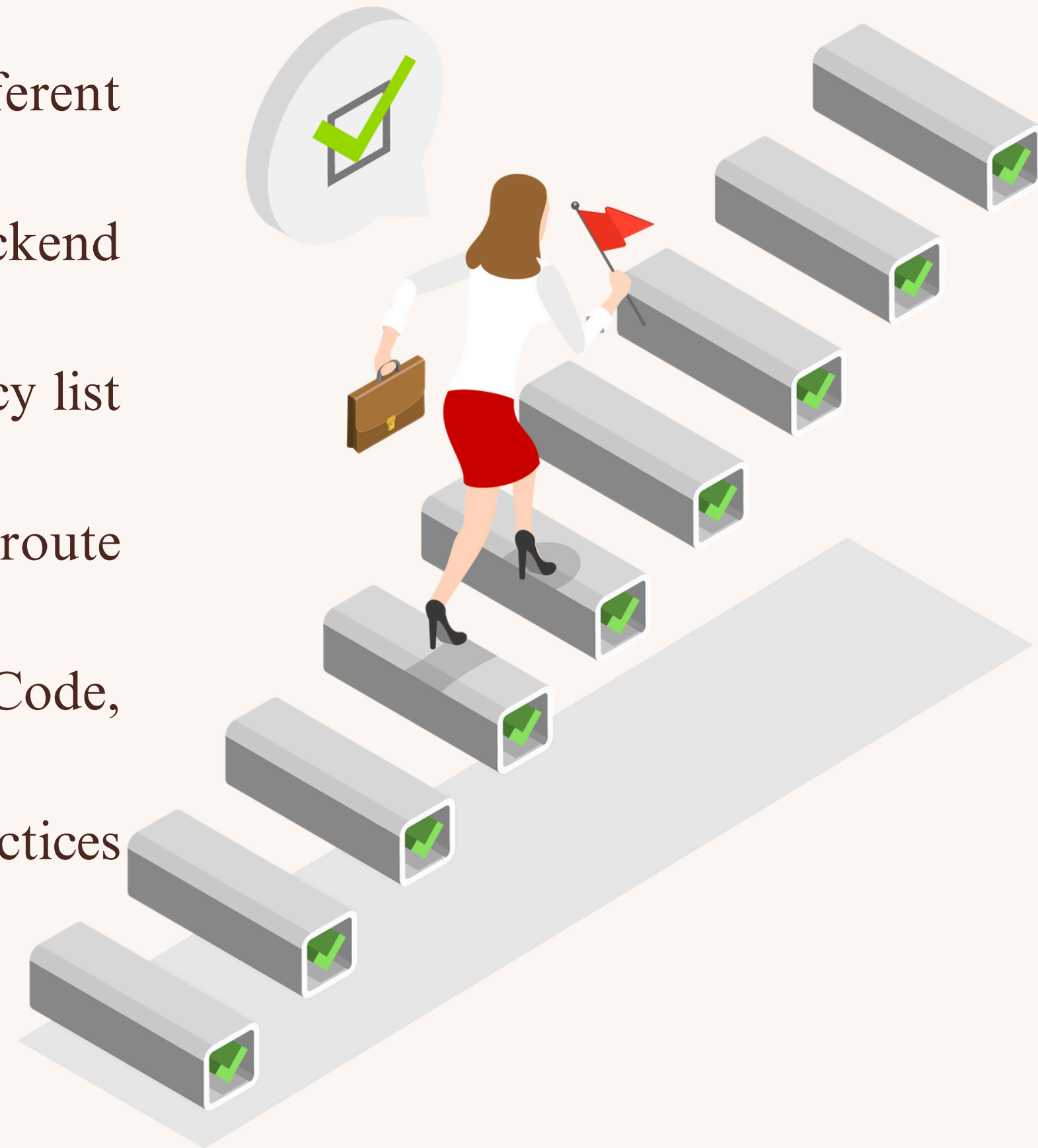
# RESULTS

- The system successfully calculates optimal travel routes based on shortest distance or shortest time using Dijkstra's algorithm.
- The backend (Java, Spring Boot) and frontend (JavaScript, HTML, CSS) are fully integrated through REST APIs, ensuring smooth data communication.
- User inputs (starting point and destination) are correctly processed and efficient routes are displayed to the user.
- The adjacency list graph structure efficiently models locations and connections, enabling fast computation even for multiple routes.
- The website delivers accurate and quick results, improving user travel experience and promoting eco-friendly travel by minimizing unnecessary journey length.



# CHALLENGES

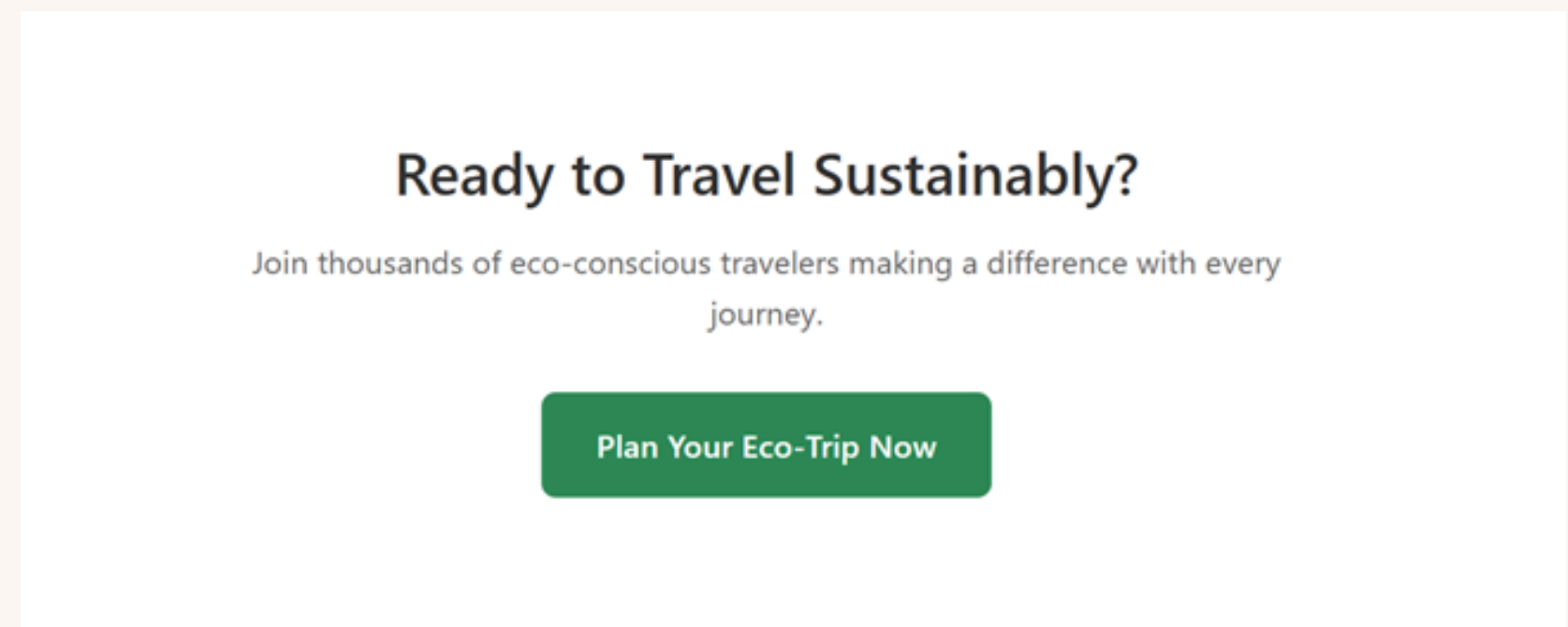
- Implementing Dijkstra's Algorithm efficiently for different user preferences (distance vs. time).
- Ensuring smooth integration between frontend and backend through REST APIs.
- Modeling real-world routes using a graph and adjacency list structure accurately.
- Handling performance optimization to ensure fast route calculations even for larger graphs.
- Setting up development environments (Eclipse, VS Code, Node.js, Vite) and managing dependencies correctly.
- Maintaining code modularity and following best practices for future scalability and easier debugging.





# CONCLUSION

- **Efficient Travel Planning:** Combines Java-Spring Boot backend and JavaScript frontend.
- **Route Optimization:** Uses Dijkstra's algorithm for distance and time-based route calculation.
- **Modular Architecture:** Structured design with separate layers (controllers, services, models, repositories) for scalability.
- **Tools Used:** Eclipse for backend, VS Code for frontend.
- **Future Enhancements:** Ability to integrate scenic routes, toll avoidance, and real-time traffic data.
- **Platform for Innovation:** Strong foundation for continuous improvements in eco-friendly travel planning.





Thank  
you very  
much!