Applied Database Technologies

Spring 2024

Final Project Part 4

Section - 1

1. Full URL to the deployed Web App made publicly available:
   https://adtprojectg10-o8ug2mxc5bktasftwdstzz.streamlit.app/

2. full URL to the GitHub where your team stored web app coding:
   https://github.iu.edu/sgosawi/ADT_Project_G10

Section - 2

Purpose:

CityLink Rideshare Hub is an innovative platform designed to make intercity travel more accessible, enjoyable, and sustainable by connecting drivers and passengers with similar travel routes. The project aims to create a user-friendly platform where individuals can easily find and participate in ridesharing opportunities, improving connectivity and reducing transportation costs.

Tech Stack for building the project:

For the CityLink Rideshare Hub project, our development process utilized a combination of modern web technologies and database management systems to create an efficient and user-friendly ridesharing platform. Here's how we built our project using these tools:

Front-End Development with Streamlit:
We chose Streamlit for its simplicity and effectiveness in building interactive web apps quickly. Streamlit allowed us to develop the front-end interface where users can input travel details, find matches, and view rideshare opportunities.

Back-End Development with Flask:
Flask, a lightweight and flexible Python web framework, was used to handle the back-end services of our platform. It manages the API requests and responses, integrates with our PostgreSQL database, and handles the business logic.

Database Management with PostgreSQL:

PostgreSQL, a powerful open-source object-relational database system, was used to store and manage all data related to rides, users, transactions, and vehicle details. We utilized its robust features like complex queries, data integrity, and concurrency support to ensure that our database could handle the high demands of an intercity rideshare service.

API Development with FastAPI:
FastAPI was employed to build high-performance APIs for our platform, facilitating the connection between the front-end and back-end services. FastAPI’s speed, ease of use, and built-in support for asynchronous programming made it an excellent choice for our real-time rideshare matching system. It also provided automatic interactive API documentation, which simplified the testing and collaboration process.

By integrating these technologies, we built the CityLink Rideshare Hub as a comprehensive solution for intercity travel. This platform not only supports efficient data handling and user interactions but also fosters a community-oriented approach to ridesharing that emphasizes cost-effectiveness and environmental sustainability.

Data:

The CityLink National Rideshare Hub dataset is a fictional collection designed to reflect the day-to-day operations of a rideshare company. It includes important details like Ride IDs and User IDs to track each ride and user. It also lists the starting and ending cities for trips, the timing of rides, and the number of seats available and filled. This helps us understand where and when people travel and how many seats they use.

We also include the type of car used and the cost per seat. This information helps us figure out what kinds of cars people prefer and how much they're willing to pay. We used real-world car rental datasets from Kaggle to make our data realistic. This way, we can better simulate what a real rideshare service would look like and how it might operate, helping us improve our service and meet the needs of our users effectively.

Functionalities:

Rider Dashboard Functionalities:

Add New Ride: Riders can create a new listing for a ride by inputting details like departure and arrival cities, dates, and times.

Vehicle Information: They can enter specific vehicle details such as type and number, which helps passengers to know the ride they will be traveling in.

Route Customization: The dashboard allows riders to specify exact pickup and drop-off locations.

Seat Management: Riders can manage the number of seats they offer, adjusting availability as needed.

Pricing and Currency: There is flexibility in setting the price for each seat and selecting the preferred currency for the transaction.

Special Amenities: Riders can list amenities provided in the ride, making the trip more appealing to passengers.

Ride Detail Management: Once a ride is posted, riders can update or delete ride details, maintaining control over their ride offerings.

User Profile Management: Riders have the ability to update their profile settings.

Secure Authentication: The platform features a secure login and logout functionality.

Passenger Dashboard Functionalities:

Search for Rides: Passengers can search for rides using filters for departure and arrival locations and dates.

Seats Reservation: Passengers select the number of seats they want to book.

Booking and Ride Details: They can view detailed descriptions of rides, including timing and vehicle information, before booking.

Booking Management: Passengers have the option to manage their bookings, including the ability to cancel or modify their reservations.

View Booking History: There's a feature for passengers to view their booking history, including current and past rides.

Interact with Ride Listings: Passengers can interact with the list of available rides, getting necessary information to make informed decisions.

General Platform Functionalities:

User Profile Management: Both riders and passengers can edit their profiles and set preferences for their roles within the platform.

Interactive Dashboards: Both the rider and passenger dashboards are designed for ease of use, with clear navigation and intuitive layout.

Secure User Authentication: Users can securely log in to and log out of their accounts, protecting their personal data.

Responsive Design: The dashboards are designed to be responsive, providing a seamless experience across various devices and screen sizes.

Section - 3

Issues faced during the build process:

We faced technical challenges with API stability and session management in the CityLink National Rideshare Hub. The API occasionally failed to respond, so we implemented a circuit breaker pattern to maintain service continuity during disruptions. For session management, we upgraded to token-based authentication using JSON Web Tokens (JWT), allowing users to switch accounts without requiring a full session reset. This enhancement streamlined user experience, ensuring smooth and secure transitions between logins.

Teamwork:

Name	Tasks	Average Time Spent
Manikanta Kodandapani Naidu	1. BackendAPI development 2. Database design 3. Webpage design, development and integration 4. Documentation	10 hours 3 hours 4 hours 2 hours
Siddharth Gosawi	1. BackendAPI development 2. Database design 3. Webpage design, development and integration 4. Documentation	10 hours 3 hours 4 hours 2 hours
Monisha Patro	1. Webpagedesign, development and integration	10 hours 3 hours

	2. Database design 3. BackendAPI development 4. Documentation	4 hours 2 hours