4/7/2023

UniGo

Capstone Project Report

Yash Patel
Krupal Patel

DUE DATE IS THE DO DATE

Table of Contents

Introduction:

Transportation is an essential aspect of everyday life, and university students and staff are no exception. Efficient, safe, and affordable transportation is a necessity for university students and staff, but traditional modes of transportation often fall short in meeting their needs. Taxi services are expensive, while public transportation is often unreliable and time-consuming, and sometimes provide limited accessibility based on location and timing often leading to multiple connections. Private transport is an alternative that is not available to a lot of students in the university, and adds to the environmental concerns in the form of an increased carbon footprint. All these factors lead to an unmet demand for a more efficient and affordable transportation option.

UniGo is a ride-share application that addresses these concerns and is specifically designed for the university community. UniGo aims to provide a reliable, eco-friendly, and cost-effective alternative to traditional transportation methods. The application offers a platform to its users where they can search and book rides quickly and easily and commute with their fellow students and faculty of the University commuting in the same direction, while maintaining user data security and privacy.

The development of UniGo has been a complex project, requiring a comprehensive understanding of software engineering principles and the implementation of best practices. This report provides an overview of UniGo's technical details, budget, code testing, and a final overview in the form of results.

System Design and Architecture :

UniGo is a ride-sharing web application designed to provide a convenient and reliable transportation solution for the University of Regina community. The application's architecture follows a client-server model, where the client is the web-based user interface, and the server is a Node.js backend that handles data storage and retrieval. The development of UniGo was performed by utilization of the Agile methodology, which provided a flexible and iterative approach to the project while ensuring timely completion of the deliverables and milestones.

- Front-End Architecture:

The front-end interface of UniGo is built using React, a popular JavaScript library for building user interfaces. React has a component-based architecture, which allows for modular, reusable code and enables efficient rendering of user interfaces. Additionally, Bootstrap CSS and HTML are used to provide a responsive and visually appealing design.

The components used for the front-end architecture of UniGo include a map component, a ride component, and a user component. The map component integrates the Google Maps API, enabling real-time location, and route optimization. UniGo incorporates a fare-estimation algorithm as a key feature, which is responsible for predicting the cost of the trip for the rider and driver based on parameters such as distance, time, and the route used. The ride component allows users to request or schedule a ride, while the user component provides access to user profile information, and ride history.

- Back-End Architecture:

The back-end of UniGo is built using Node.js, a popular server-side JavaScript platform that enables efficient handling of server-side logic and API integration. The back-end uses a MongoDB database to store user and ride information. MongoDB is a popular NoSQL database that provides a flexible and scalable data storage solution for web applications.

The back-end architecture of UniGo includes various modules, such as the user module, the ride module, and the admin module. The user module handles user authentication and profile management, while the ride module manages ride requests, schedules, and history. The admin module provides access to administrative tasks, such as user account management and ride monitoring.

- Future Work:

MVP that are classified as future upgrades for the project. These features include :-

- Payment System and Wallet creation : A wallet page can be introduced to UniGo that will provide users with an easy and secure method to make transactions within the web-app. This will allow users to transfer and withdraw funds between their bank account and wallet at their convenience. This wallet can then be used for transactions required for the commutes.
- Chat System : Inbuilt chat service that will enable riders and drivers to easily connect within the web-app to provide efficient coordination and updates for their scheduled rides to enhance their experience of UniGo.

- Rating System : A system that can be introduced in the web-app that will allow users to express their satisfaction/dissatisfaction on their ride experience, with the aim to promote accountability and encourage good behavior.
- Token System : To promote frequent utilization of the application, a token system will be introduced within the application that provides users with discounts based on their usage of UniGo and their ratings.

- Project Management:

Project management was performed through the user of Github's Kanban boards to keep track of progress. Kanban's easy-to-use and intuitive visual interface is helpful for monitoring the tasks, and managing the workflow of the project. Regularly scheduled scrum meetings were performed with mentors Dr. Kin, and Dr. Timothy, to showcase the progress of the web-app and gather mentor feedback and guidance on the project. The scrum meetings were valuable in providing insights and guidance on technical aspects, design choices, and maintaining the correct flow of project development to adhere to the standards set forth by the university.

Budget and Cost Analysis:

The development and maintenance of UniGo involve several expenses. This section provides an overview of the costs associated with building and running the application.

- Cloud Services:

UniGo leverages cloud services for various purposes such as hosting, storage, and API calls.

Google cloud was utilized for this project for the purpose of API calls useful for maps and route optimization. This project utilized the free tier offered by Google cloud based on its current set of requirements.

The free tier offered by Amazon Web Services was utilized for the creation of the Database , as well as hosting this project.

- Operational Costs:

UniGo's operational costs include ongoing expenses such as server maintenance, bug fixes, and updates. These costs may increase as the app scales up and more users join the platform.

LoFi Prototypes :

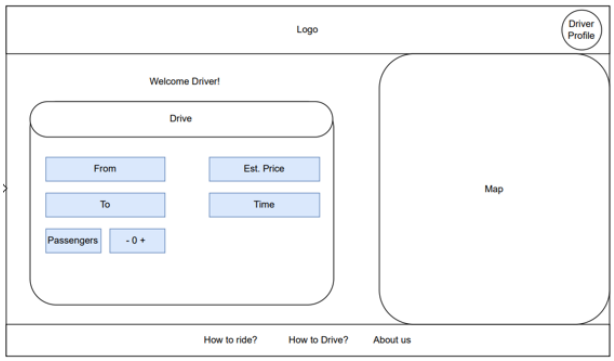

Figure 1

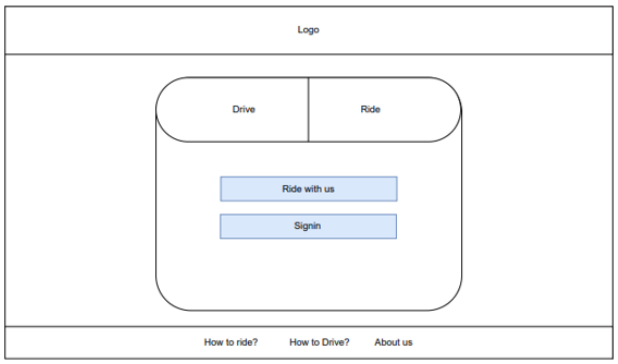

Figure 2

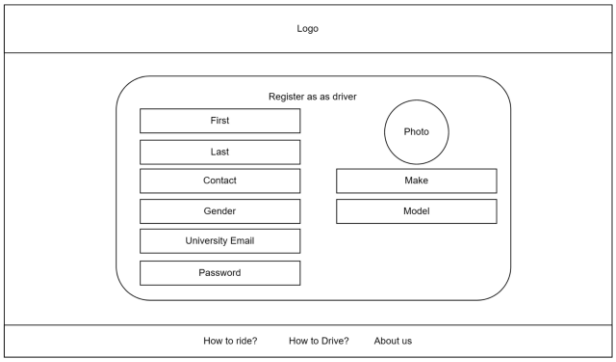

Figure 3

Testing :

The UniGo ride-sharing web app for the University of Regina community requires a comprehensive testing plan to ensure it meets functional and non-functional requirements. This section of the report summarizes the testing approach, tools, resources, and execution for the entire backend and frontend of the project.

- Test Plan

The test plan outlines the testing types, tools, resources, and timelines. It includes testing the app across various devices and platforms and verifying functional and non-functional requirements, such as integration with Google Maps API, Node.js, and MongoDB.

- Testing Framework and Setup

Jest serves as the testing framework, and supertest library is used for sending requests to app routes. The test cases, organized in "describe" blocks with "it" statements, cover both unit and integration tests for the web-app's functionalities. Tests are run in the terminal when the command ‘npm test’ is executed.

- Objectives

- Ensure functional requirements are met
- Verify reliability, performance, and usability
- Ensure compliance with quality standards and requirements

- Testing Strategy
    - Unit Testing: Verify individual components' functionality
    - Integration Testing: Verify component integration with each other and external systems (Google Maps API, Node.js, MongoDB)
    - System Testing: Verify the app's overall functionality

```
//Unit Test SignIn
describe(signin, () => {
  it("post" + signin, async () => {
    const response = await request(app).post(signin).send({
      email: "Ad4rwere4@gmil.com",
      password: "rahul1",
    });
    expect(response.status).toEqual(401);
  });
});
// Integration test SignIn
describe(signin, () => {
  it("post" + signin, async () => {
    const response = await request(app).post(signin).send({
      email: "Ad4rwere4@gmil.com",
      password: "rahul123",
    });
    expect(response.status).toEqual(200);
  });
});
```

Figure 4

- Test Environment
    - Various devices (smartphones, tablets, laptops)
    - Different operating systems (iOS, Android, Windows)
    - Google Maps API, Node.js, MongoDB

- Test Deliverables
    - Test Plan
    - Test Cases
    - Test Scripts
    - Test Reports
    - Defect Reports

- Backend Testing

Carious aspects are covered in this part of testing, such as user authentication, ride management, and error handling. Test cases include a combination of unit and integration tests for verifying routes, data handling, and interactions with external systems.

- Frontend Testing

This part of testing involves testing out React component rendering, user interface functionality, and interactions with the backend. Tests utilize tools like React Testing Library, Jest, and Cypress for verifying component rendering, user actions, and navigation within the app.

```
import { render, screen } from '@testing-library/react';
import App from './App';

it("renders without crashing", () => {
  render(<App />);
});
```

Figure 5

Final Preview of Web-App :

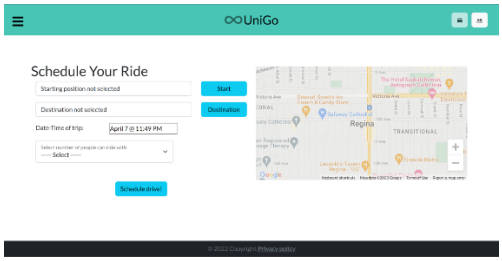

Figure 6

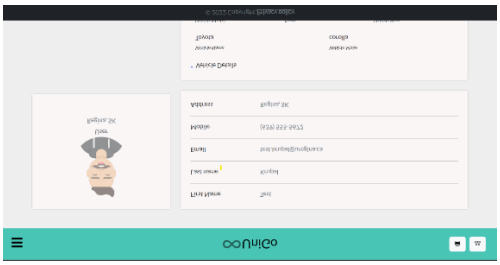

Figure 7

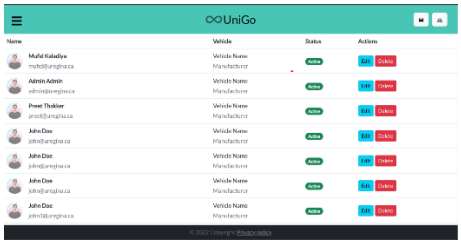

Figure 8

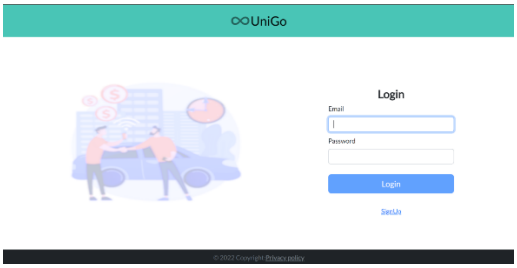

Figure 9

Conclusion :

The development of UniGo as a ride-sharing application catered to the needs of the students and faculty of the University of Regina, has been a success. The implementation of key features of the application, such as the ride scheduling, optimized routing, and algorithm-based price estimation were successfully added. The scalability of the application ensures that addition of various new features to the application in the future is easily done. The web-app displays its secure development with the use of services such as Google Cloud and Amazon Web Services, and by the use of Agile methodology for its development process.

Overall, the UniGo ride-sharing application is poised to be a valuable addition to the ride-sharing market, with its user-friendly interface and efficient features. The project team hopes that this application will be useful to students and staff of the university, and also to the wider community beyond. The experience gained from this project has also been invaluable to the team members, providing them with real-world software development experience and preparing them for future careers in the industry.