School of Electronic Engineering
and Computer Science

Final Year Report

Programme of study:
Computer Science with Management
with Industrial Experience (IE)

Project Title:
CabQuote

Supervisor:
Dr. Fabrizio Smeraldi

Student Name:
Devanshu Goyal
190084699

Final Year
Undergraduate Project 2022/23

Date: 09/05/2023

Abstract

CabQuote is an innovative web application that enables users to compare taxi fares from various cab companies in real-time. This project's primary objective is to facilitate the process of locating the most cost-effective mode of transportation for users, considering distance, price, and provider reputation. This application hopes to simplify the decision-making process for those in need of transport by aggregating cab prices and enabling straightforward comparisons.

The CabQuote platform is constructed with a modern web development stack, which includes Vue.js for the front end, Express.js for the back end, and SQL for database administration. The system's architecture consists of three tiers: the presentation layer, the application logic layer, and the data storage layer. The user interface is designed to be user-friendly, intuitive, and aesthetically pleasing in order to provide users with a seamless experience.

During the development process, extensive research was conducted on the APIs of various taxi companies. Due to the modification of the project's scope, however, not all APIs were incorporated into the final system. Instead, the CabQuote platform prioritises a select group of providers and locations. The system's data storage component is implemented with SQL and a custom-built database designed to store information about taxi providers, pricing, and locations.

CabQuote was implemented by establishing a development environment, developing the frontend and backend, and integrating the frontend and backend components. Multiple code samples and screenshots were used to illustrate the development process and provide visual representations of the system's essential components. The testing phase included the development of a test plan, test cases, and the execution of tests. Throughout the testing procedure, numerous issues were discovered and resolved, ensuring the functionality and dependability of the application.

The CabQuote initiative culminated with an assessment of the platform's usability, user interface, performance, scalability, and security. While the application accomplished its initial objectives, there is room for improvement in the future. Possible enhancements include expanding the number of taxi companies, integrating real-time traffic data, implementing a user review system, and investigating machine learning algorithms for improved cab recommendations.

In conclusion, the CabQuote platform demonstrates the capacity of technology to simplify routine duties and make information more accessible to users. It provides valuable insights into the practical implementation of programming skills and theoretical knowledge acquired throughout the academic programme. Despite certain limitations, CabQuote exemplifies the impact that technology can have on enhancing urban mobility and transportation options.

Acknowledgement

I would like to express my deepest gratitude to my supervisor, Dr. Fabrizio Smeraldi, for his invaluable guidance, support, and encouragement throughout the duration of this project. His expertise, knowledge, and insightful feedback have been instrumental in shaping the outcome of this work, and I am truly grateful for the opportunity to learn and grow under his mentorship.

I also extend my heartfelt appreciation to my family for their unwavering love, support, and understanding during my academic journey. Their belief in my abilities and their constant encouragement have provided me with the strength and motivation to persevere through the challenges I have faced.

My sincere thanks go to my friends and fellow students for their camaraderie, assistance, and constructive discussions. Sharing ideas, collaborating, and learning from each other have significantly enriched my academic experience.

Lastly, I would like to acknowledge the faculty and staff of the computer science department for their dedication to providing an excellent educational environment. Their commitment to fostering an atmosphere of learning, curiosity, and growth has played a crucial role in my development as a student and a professional.

To all those who have contributed, directly or indirectly, to my academic success and personal growth, I offer my heartfelt thanks and appreciation.

Table of Contents

Table of Figures

Chapter 1: **Introduction**

1.1 Background

Time and money are now irreplaceable resources in our existence.

Numerous services have been revolutionised over the past two decades by technological advancements and the widespread adoption of the internet, making them more efficient and convenient. In the modern era of technology, time is considered the most valuable resource. From digital highways and bullet trains to self-driving cars, the transport industry has witnessed massive development and innovation, all of which seek to ensure safer journeys, faster delivery, and enhanced experiences for everyone.

In this fast-paced world, the emphasis has shifted to boosting output and productivity. When it comes to travel, a growing percentage of commuters rely on public transport or ride-hailing services for work or pleasure.
The days of hailing black cabs on the street or booking taxis via phone call are long gone. Historically, the taxi industry was dominated by full-time chauffeurs. With the advent of digital innovations, however, taxi drivers can now pursue multiple careers.

Mobile apps and new technologies have significantly changed the taxi industry, transforming it into one that is more customer-centric and technology-driven. These innovations are intended to enable taxi drivers to diversify their careers while providing customers with higher-quality services. The introduction of ride-hailing applications has enhanced the taxi experience by utilising real-time traffic data and live GPS systems to compute fair prices based on the shortest distance and current traffic conditions.

1.1.1 Overview of the cab booking Industry

The advent of ride-hailing platforms such as Uber, Lyft, and Grab has led to significant growth within the taxi booking industry in recent years. Grand View Research projects that the global ride-hailing market will reach $126.5 billion by 2028, expanding at a CAGR of 19.8 percent from 2021 to 2028 (Grand View Research, 2021).

In the United Kingdom, a handful of significant players dominate the ride-hailing market. According to Kunst (2023), 67.7% of adults in the United Kingdom used the Uber platform, with Bolt coming in second at 8.8% and Ola coming in third at 7.5%. Other lesser competitors, such as FreeNow, Gett, and ViaVan, held the remaining market share (Kunst, 2023). Despite the expansion of the industry, there are still some obstacles to overcome.

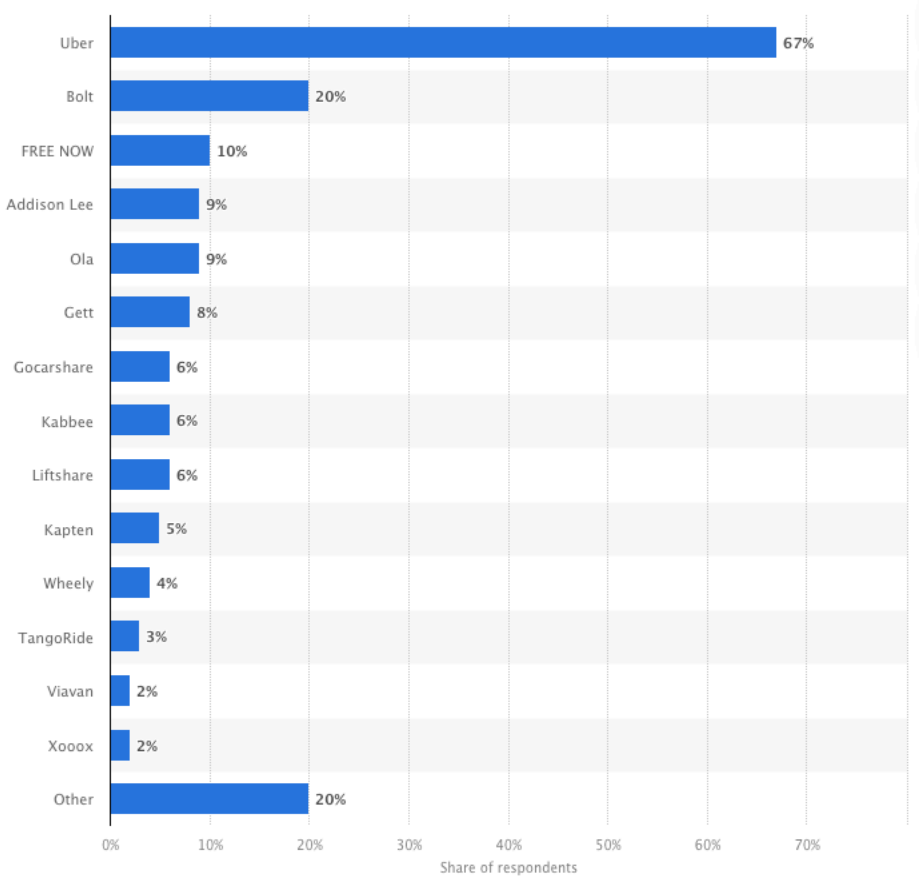

Figure 1 Ride sharing online taxi usage by brand in the UK in 2022 (Kunst, 2023)

Problem Statement

Today's online taxi service is diverse. Since the innovation and implementation of Uber in 2009, the taxi industry has become an attractive sector for many. Over the past decade, multiple competitors have launched their online cab services with varying user interfaces, providing users with new features and offers in comparison to Uber, their largest competitor. In this industry, there are numerous competitors to choose from, resulting in a large customer base comparing their delivery of service requests. Customers typically download numerous apps at first, based on the marketing strategies of competitors, including new membership offers such as discount coupons, free first few rides, spin the wheel, etc. Now, when a customer needs a taxi service, they compare multiple taxi applications based on pricing, waiting time, dependability, and passenger capacity. Customers may miss out on the best pricing offered by other potential cab service apps if they check each app individually to compare prices and availability among the few major market competitors. With few monopoly firms on the market, they have taken advantage of the lack of consumer knowledge by offering increased prices and longer wait times based on demand patterns. With many other taxi-hailing systems idle, this has become one of the more difficult problems to tackle from a customer's perspective.

1.2 Aim

The project primarily focuses on unifying the current cab booking application market in place. A focus groups of random participants were asked their level of satisfaction and ease when booking a cab during peak hours. More than half of the participants voted their level of satisfaction as an average of 4/10. The comments majorly described the hassle of checking multiple applications to get the best deal, based on two vectors – time and cost. The current proposed design of the application is a web application comparing average prices of major cab providers based on previous user data. Based on user preferences such as time/cost, the app will compare ride fare and estimated time from all the cab companies and predict the most suitable results.

The proposed development of the app includes ideas, market research, defining features and functionalities, choosing a development path, and finally testing the prototype.

1.3 Objectives

To achieve the aims of the CabQuote project, the following objectives must be met:

1. Develop a desktop-based application that enables users to compare average cab prices across major providers effectively.
2. Implement the primary functionality of fetching and displaying average cab prices based on user input, offering users a convenient platform for comparing cab fares.
3. Assess and address potential challenges in integrating real-time APIs from major cab providers for future improvements in the application.

1.4 Research Questions

The following research questions will guide the project and help evaluate its success:

1.) How does the desktop-based application streamline the process of comparing cab prices for users, making it more convenient and efficient?
2.) What are the primary challenges faced by users when comparing cab prices with existing solutions?
3.) What are the potential limitations in integrating real-time APIs from major cab providers, and how can they be addressed for future improvements in the application?

Chapter 2: **Literature Review**

Ride-hailing apps have become the most popular method to book cabs and compare best prices. However, in an attractive industry with high competition, monopoly of few major firms it has become difficult from customer-oriented point of view to avail the best deals out there. This chapter provides a detailed analysis of suggested technology to be used for the proposed solution. It will he also providing us analysis about existing solutions in the market and how the proposed solution differs from rest of them.

2.1 Cab Service Industry and Market

In recent decades, the taxicab industry has undergone significant expansion and transformation. Traditional transportation services have transitioned into contemporary ride-hailing platforms, which have disrupted the market and created new business opportunities.
Diverse factors, such as shifting consumer preferences, technological advancements, and the advent of innovative business structures, have contributed to this transition.

Initially, cab companies relied on traditional methods, such as contacting call centres or hailing vehicles on the street. The emergence of smartphones and GPS technology led to the development of ride-hailing applications such as Uber and Lyft, which have revolutionised the industry by making transportation services more convenient and accessible (Moll & Fizazi, 2018).

The global ride-hailing market is anticipated to expand at a CAGR of 20.4% between 2021 and 2028, reaching $209.6 billion by the year 2028 (Grand View Research, 2021).

The rise of ride-hailing services has been influenced by urbanisation, increasing traffic congestion, and the rising demand for convenient and cost-effective transportation options. Consumers prefer app-based services due to their user friendliness, pricing transparency, and variety of vehicle options (Gerpott & Paukert, 2019). Significant financial implications have resulted in the expansion of the taxicab industry, including the emergence of employment opportunities and a boost to local economies.

2.2 Consumer Behaviour and Price Comparisons

To adjust their offers to the requirements and preferences of those they are targeting, firms must have a solid understanding of consumer behaviour. Price, convenience, and availability are a few factors that affect customer decisions.

Given that people frequently look for the most economical transportation option, pricing is a crucial factor in decision-making. According to studies, decreasing prices have a beneficial impact on how often consumers use ride-hailing services (Wang et al., 2020).

Convenience and accessibility also have an impact on consumer decisions because they seek out services that are quick and simple to obtain and provide dependable transit options. Consumers can now effortlessly compare rates, wait times, and fleet selections among various providers thanks to the growth of ride-hailing services.

To meet this need, price comparison sites like Google Maps and Citymapper have developed new features, giving customers the ability to choose cab services with greater expertise (Danziger, 2020).

2.3 Existing and similar solutions

2.3.1 Google maps price comparison

Google Maps features integration with several ride-hailing services like Uber, Lyft, and many more. In 2018, Google Maps launched an initial ridesharing solution for in-app navigation to help firms around the globe integrate Google Maps into their apps for reliable, real-time routing. The ride-hailing industry has improved significantly in past years, with requirements for more ride-hailing services to collaborate and feature globally. (Danziger, 2020)

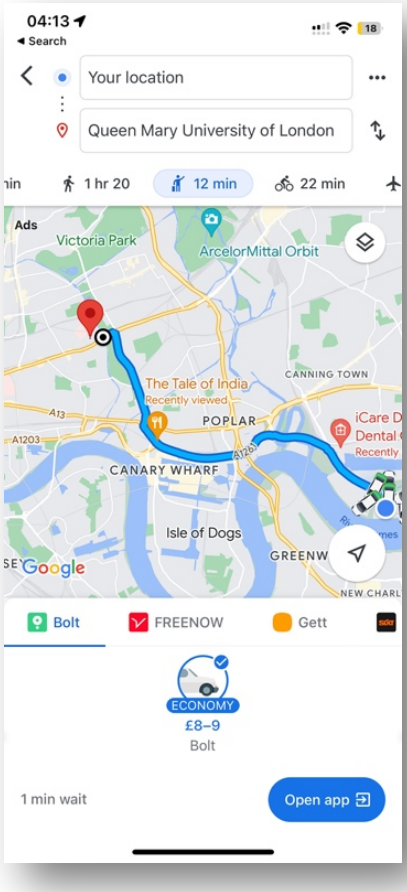

Figure 2: Google maps cab price comparison feature

Google Maps has done a great job integrating this feature into their application, allowing users to choose from different ride-hailing providers based on price and wait time. Now the issue from customer feedback and reviews with this solution is that its data is just an estimate and not real-time data. Upon clicking on Open App, it takes you to the ride-hailing provider's app, and the actual price and waiting time differ from what's estimated by the app. This is a major issue since the user must check different apps again for a real-time estimated price and waiting time. Another issue faced is that many users do

not have all the ride-hailing apps installed on their devices. This solution takes them to the app, where the user must install the app to use the services. In most cases, the user is on a mobile data network, and they face difficulty installing and registering for the new ride-hailing service on a mobile data network.

2.3.2 City mapper

City Mapper is a public transit app and mapping service that displays transport options with live timings and integrates data for all urban modes of transport, including public transport, ride-hailing services, walking, cycling, and driving. (WikiPedia, n.d.) City Mapper follows a very similar trend in comparing costs between ride-hailing services, just as Google Maps does, but with more options available to the public. The figure below accurately describes the ride-hailing comparison feature in the application.

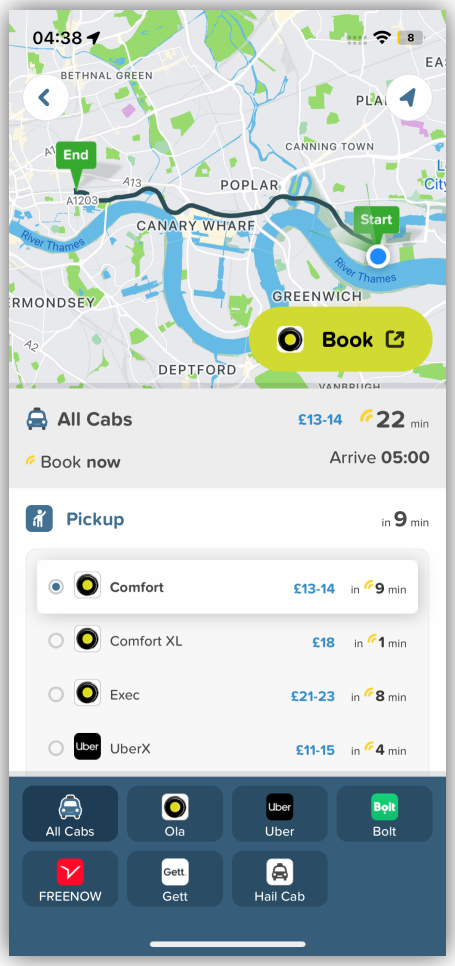

Figure 3: CityMapper cab price comparison feature

2.4 Comparison of existing solutions

Application	Features Identified	Limitations
Google Maps	• Ability to integrate cab comparison within the GPS system • Easy to use and user-friendly interface • Compatible with multiple devices • Powerful API resulting in low crash probability	• Wait time and price prediction not accurate with the real time data • Requirement of sign up and installation of 3rd party apps on the device • Only major ride hailing partners available • No in app feature to book directly
Citymapper	• Doesn't just rely on open data from city authorities, uses data from multiple sources and combine them • More accurate estimation of wait time and price prediction • Modern framework with seamless approach to application UI	• Not available globally, limited to some regions • Data source not reliable since it compiles multiple database sources • Only major ride hailing partners available • No in app feature to book directly
CabQuote(Proposed Project)	• Accurate estimation of average price based on input time data from users. • Unified system with multiple ride-hailing apps to compare from • application UI/UX, featuring multiple tab approach	• Maybe just limited to web application due to contingency and server issues • Not available globally • Difficult to support all user devices. • Geo-graphical difficulties

2.5 Technologies for Web Development

Web development technologies play a crucial role in the development of contemporary applications, such as taxicab fare comparison websites.
The technologies in question can be broadly classified as front-end and back-end tools.

Front-end technologies consist of the design and development of the user interface (UI) and user experience (UX) of a web application. HTML, CSS, and JavaScript are the fundamental technologies for developing websites and interactive elements. Modern JavaScript libraries and frameworks, such as React, Angular, and Vue, facilitate the development of scalable and maintainable web applications (Liu, 2018).

Backend technologies manage server-side processing, database administration, and application logic. Back-end development generally uses programming languages including Python, Ruby, PHP, and JavaScript (Node.js). While databases like MySQL, PostgreSQL, and MongoDB store and manage data, web servers like Apache, Nginx, and Express.js serve web pages to users (Varshney, 2019).

The web application for the proposed CabQuote project will be developed using a combination of front-end and back-end technologies that heavily make use of the tools mentioned earlier. These technologies should be chosen based on their usability, scalability, and compatibility with the project's specifications.

The proposed use case diagram below illustrates the various actions and processes of the system, from user requests to scrapping the application, some of which have been implemented and others are potentially in the production development of the application.

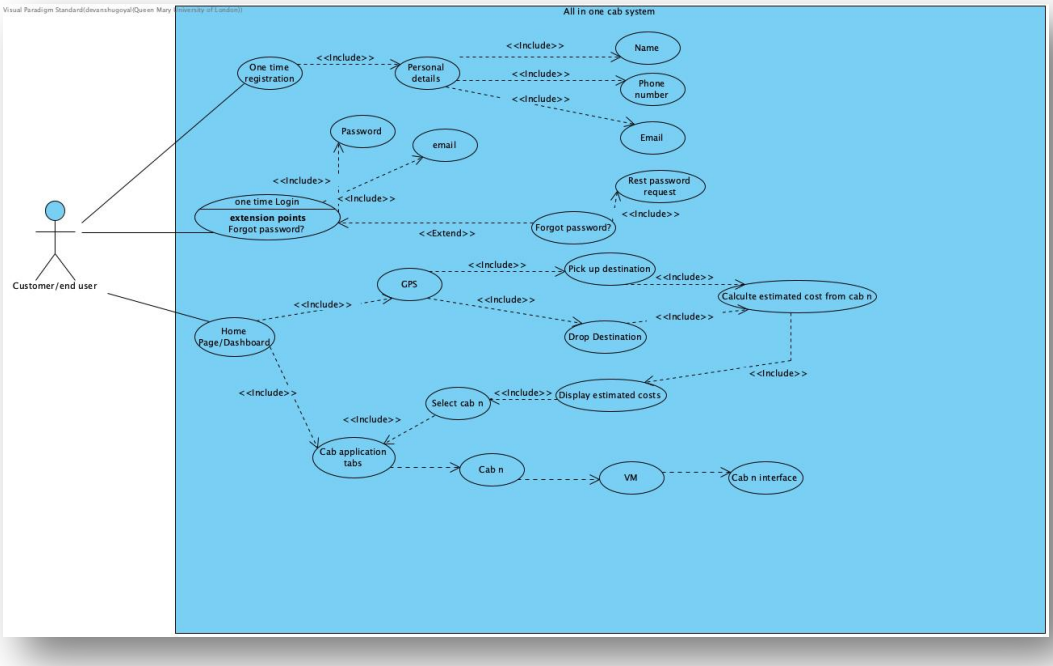

Figure 4: Proposed Use case diagram for CabQuote

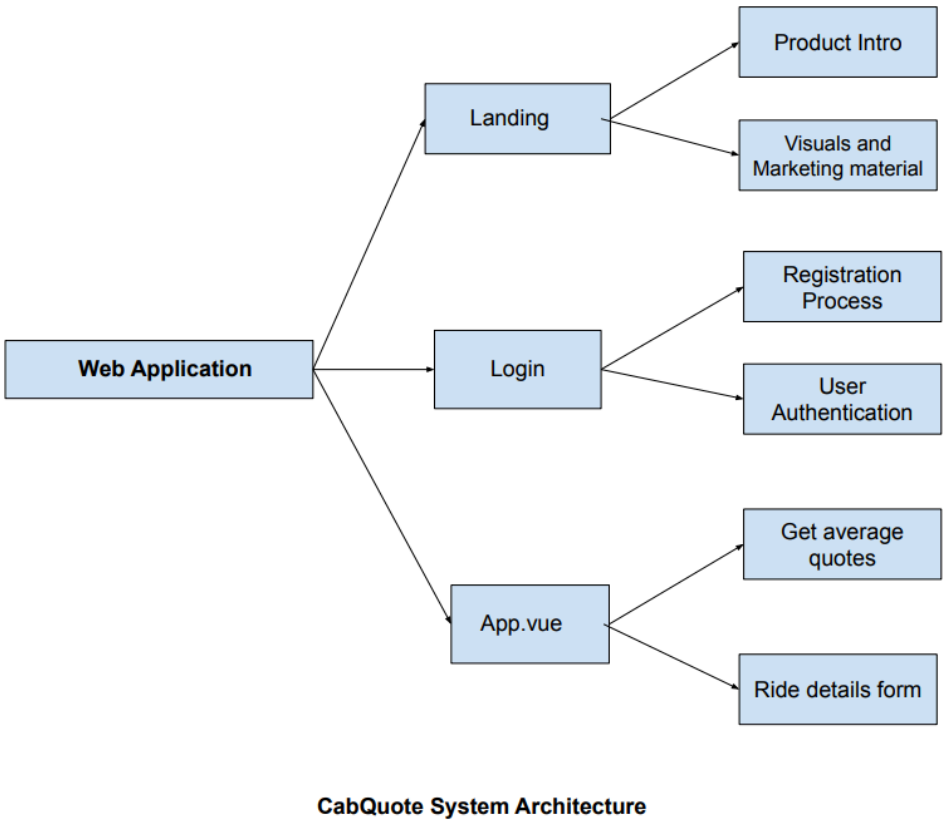

Figure 5:CabQuote System Architecture

2.6 Analysis and design

A brief analysis of core functional, non-functional and security requirements for the project. Prototype and sample designs for the project have been illustrated in this section as well.

2.6.1 Functional Requirements

	Requirement	Completed
1.	User must be able to do one time registration sign up upon using the application for the first time	Yes

2	User should be able to login into the application using their credentials. The system should make sure the login process is smooth	Yes
3	Once logged in the application must request data and location permission requests for personalised experience	Partially Yes
4	The application should have a landing page/dashboard with all essential features accessible	Yes
5	Dashboard should include a GPS API with pick up and drop off destination search	Attempted
6	Dashboard should include a subpage based on destination that compares base rates of multiple service providers	Yes
7	Application must contain multiple tab format with different service providers in a user-friendly UI	Partially Yes
9	The application should remember credentials for cab tabs to ensure efficient and hassle-free usage	Yes
10	There should be minimal time lag between the application and the cab provider interface to ensure user friendly experience	Yes

2.6.2 Non-Functional Requirements

	Requirement	**Completed**
1.	The application login page should feature a forgot password feature in case user is locked out of their account	Attempted
2	Application should be able to provide ETA for the ride duration	No
3	Application should have good security and automated database implementation	Yes
4	The application should be accessible on major devices such as Phone, tablet, and computer	Partially Yes
5	The application should be able to predict frequent user travel pattern and auto suggest cab fare’s	No

6	Application should be responsive during peak hours when there is high demand	Partially Yes
7	The application should comply with all regulatory measures of the government and 3rd party cab provider policies	N/A

2.6.3 Security requirements

	Requirement	**Completed**
1.	The application login page should feature a forgot password feature in case user is locked out of their account	Partially Yes
2	The application should have a double authentication login every 30 days	No
4	Application should clearly ask user permissions on how their data is being used	Yes

2.7 Summary

The literature review provides an overview of the cab industry and its evolution because of the emergence of ride-hailing platforms. Consumer behaviour, including price sensitivity and a predilection for convenience, has prompted the development of price comparison tools like Google Maps and Citymapper.

These limitations present an opportunity for the proposed CabQuote initiative to provide a more precise and unified taxi fare comparison platform. A combination of front-end and back-end tools will be utilised to guarantee the functionality and usability of this project, for which web development technologies play a crucial role. As the taxi service industry continues to evolve, it is essential to keep up with consumers' changing requirements and provide innovative solutions that accommodate their preferences.

Chapter 3: **Methodologies**

This chapter discusses the methodologies applied during the CabQuote application's development. It consists of requirement analysis, technology selection, risk assessment, and project planning with a timetable to comprehend the entire process.

3.1 Requirement Analysis

The requirement analysis phase of software development is crucial because it aids in determining and documenting the needs of the stakeholders, setting project goals, and defining the scope of the system. Beginning with information collection from possible consumers, surveys, and examination of already-available products on the market, the requirement analysis for CabQuote was conducted.

Based on user expectations, technological limitations, and university requirements, a preliminary list of functional, non-functional, and security needs was created. Following conversations with the supervisor, these requirements were strengthened to make sure the project matched its goals and was doable within the allotted time.

To make sure that the finished product satisfied the requirements of the project specifications, the criteria were consistently evaluated and revised during the project's development. The project's scope was changed to concentrate on the creation of a uniform average cab pricing checking system rather than a unified cab booking system thanks to the usage of an iterative methodology. This adaptability was essential for keeping the project on track with its goals while also living up to stakeholder's expectations.

3.2 Technology Selection

A variety of tools and technologies were chosen to build the web application for CabQuote. The lightweight, adaptable, and simple integration with other libraries of Vue.js led to its selection as the JavaScript framework.

The project was easily scaffolded, dependencies were managed, and the development environment was set up using the Vue CLI, which was used to develop the application.

For making API queries and managing answers, we used the well-known HTTP client library Axios. Bootstrap Vue, which offers pre-built components and a grid system that makes it simple to construct a consistent and expert-looking interface, was used for styling and responsive design. Git was used to build up the project in a version-controlled environment during development, allowing for efficient cooperation and keeping track of changes made to the codebase. The local development environment was additionally set up with the help of Node.js and npm, enabling the usage of numerous tools and packages that sped up the development process.

3.3 Risk Assessments

Risk assessments play a vital role in ensuring the success of any software project. Identifying and addressing potential risks early on helps prevent major issues that could derail the project or lead to delays. The following table presents the general risk assessments conducted for this project over six months:

Risk	Impact	Likelihood	Mitigation Strategies
Scope change	High	High	Frequent communication with the supervisor, setting clear objectives and expectations, being flexible, and ready to adapt to changes
Unfamiliar technologies	Medium	High	Investing time in learning new technologies, seeking assistance from peers, and leveraging online resources
Dependency on external APIs	High	Medium	Researching and choosing reliable APIs, implementing error handling and fallback mechanisms in the application
Tight deadlines	High	Medium	Effective time management, prioritizing tasks, breaking down work into smaller tasks, and continuously tracking progress
Lack of user feedback	Medium	Medium	Conducting surveys, gathering feedback from potential users, and incorporating their suggestions into the development process

One of the major risks encountered during the project was the drastic scope change as discussed by the supervisor. This risk was mitigated by maintaining open communication, setting clear objectives, and being flexible and adaptable to ensure the project remained on track.

3.4 Project Planning and Timeline

Proper project planning and timeline management were essential in ensuring the successful completion of the CabQuote project. The project technical implementation was planned over a three-month period, from February to April. The following table presents the project timeline, outlining the milestones achieved during each month:

Month	Milestone
February	1. Project initiation and requirement gathering
	2. Technology selection and setup of development environment
	3. Research and analysis of existing solutions
	4. Drafting initial functional, non-functional, and security requirements
March	1. Development of core application components (landing, login, and app.vue pages)
	2. Integration of external APIs for cab services and pricing information
	3. Implementation of responsive design using BootstrapVue
	4. Risk assessments and scope adjustments based on feedback from the supervisor
April	1. User testing and gathering feedback for improvements
	2. Refinement of application features and UI/UX enhancements
	3. Finalization of functional, non-functional, and security requirements
	4. Preparation of project documentation and final report

The project began with an initiation phase, which involved gathering requirements and selecting the appropriate technologies. In February, research was conducted on existing solutions, and initial requirements were drafted. During March, the core components of the application were developed, external APIs were integrated, and responsive design was implemented. Risk assessments and scope adjustments were also made during this month based on the feedback received from the supervisor.

In April, user testing was conducted to gather feedback for improvements, and the application features were refined accordingly. The finalization of requirements and preparation of project documentation and the final report also took place during this month.

This timeline provided a structured approach to the project, ensuring that each milestone was met and allowing for adjustments to be made as needed. The clear allocation of tasks and the efficient tracking of progress were critical in achieving the project's objectives within the designated time frame.

Chapter 4: **System Design**

This chapter describes the system design of CabQuote, including an architectural overview, database design, user interface design, and API design and integration. The design specifications provide a comprehensive comprehension of the system's structure and operation.

4.1 Architecture Overview

The presentation layer, also known as the user interface, is responsible for displaying data to the user and accepting input from the user. It is implemented with the ubiquitous front-end framework Vue.js. Vue.js facilitates the development of an interactive and responsive user interface, thereby facilitating a seamless user experience.

The application logic layer, also known as the business logic layer, manages user input, data retrieval, and regulations. We utilised Express.js, a Node.js web application framework, for the undertaking. Express.js facilitates the development of server-side logic and enables efficient client-server communication.

The Data Storage Layer, also known as the Database Layer, stores and retrieves system data. For the purposes of this project, we utilised the MySQL database, which has proven to be well suited to the requirements of this application. These layers exchange information and requests as necessary. The three-tier architecture's separation of concerns ensures that any changes or revisions to a specific layer have minimal effect on the other layers, thereby augmenting maintainability and scalability.

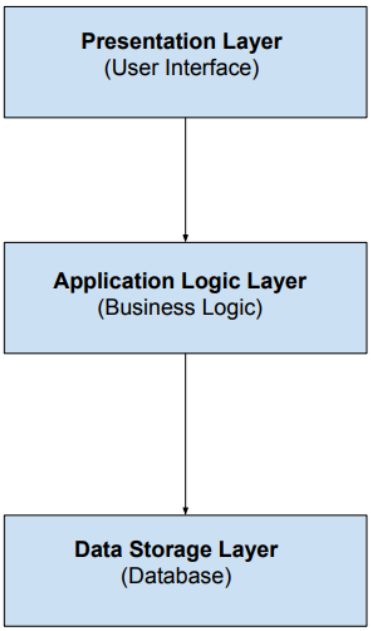

Figure 6: Three-tier Architecture System

4.2 Database Design

The database we used in the system is MySQL, a powerful, open-source object-relational database management system (RDBMS). This database system was chosen due to its robustness, scalability, and extensibility, which is beneficial for the development of the unified cab price checking application.

The primary table in the database, named "prices," is used to store data related to cab prices, distances, and pickup and dropoff locations. This table has the following columns: id, cab_name, distance, price, pickup_location, and dropoff_location. The id column is the primary key, uniquely identifying each record in the table. The cab_name column stores the name of the cab service provider, while the distance column represents the distance between the pickup and dropoff locations, calculated by the google maps API. The price column stores the cab fare for the corresponding distance, and the pickup_location and dropoff_location columns store the location data. An example of the data stored in the “prices” table:

```
id | cab_name | distance | price | pickup_location | dropoff_location
---+----------+----------+-------+-----------------+-----------------
1  | Cab A    |     1.00 |  1.00 | Location A1     | Location B1
2  | Cab B    |     1.00 |  1.50 | Location A1     | Location B1
3  | Cab C    |     1.00 |  2.00 | Location A1     | Location B1
```

Figure 7: prices SQL table sample data set

We also set up a backend folder to manage the connection and queries to the MySQL database. The file "***database.js***" contains the configuration for connecting to the database and the functions for querying the data. The file "***dummy-data-generator.js***" is used to create and populate the database with sample data for development and testing purposes.

By using MySQL as the database system, we ensured that the application's data storage and retrieval are efficient and scalable. This database design allows us to handle many cab service providers, locations, and price information, enabling the unified cab price checking system to be adaptable and accommodating to various user needs.

4.3 User Interface Design

Creating an engaging user interface (UI) is essential for any application's success, as it directly impacts the user experience. In our application, we aimed to design a straightforward, uncluttered, and user-friendly UI, facilitating a smooth experience for users comparing cab prices. The UI is comprised of three primary pages: the landing page, login page, and App.vue page (main interface). Each page was meticulously designed to guarantee user-friendliness and a satisfying experience.

Landing page: The landing page, as shown in *Figure 8*, features a clean, welcoming design. Serving as the initial point of interaction for users, it offers a concise overview of the application and encourages users to sign in or sign up to access the service. We carefully selected the colour scheme and typography to produce a welcoming ambiance and ensure easy readability.

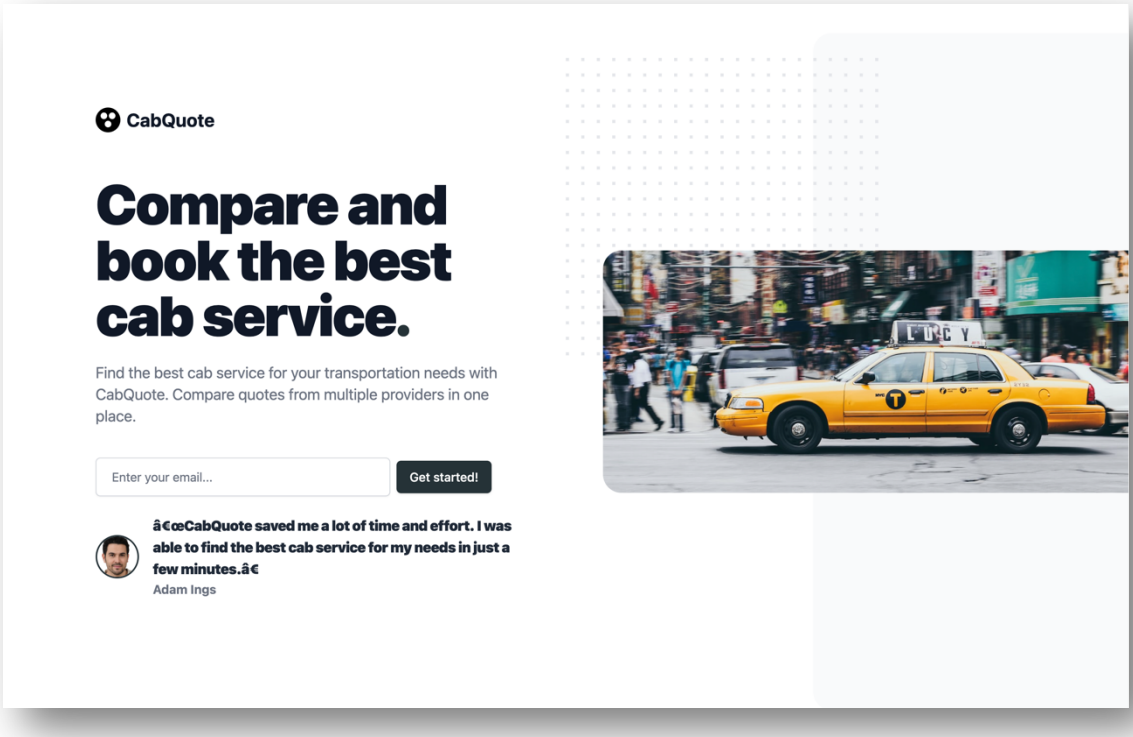

Figure 8: Landing Page Visual

Login page: The login page, depicted in *Figure 9*, enables users to input their credentials to access the application. The minimalist design ensures seamless user experience, with input fields for email and password, as well as sign up option for new users to create an account. Furthermore, the login page in future will potentially contain a conspicuous call-to-action button guiding users through the process.

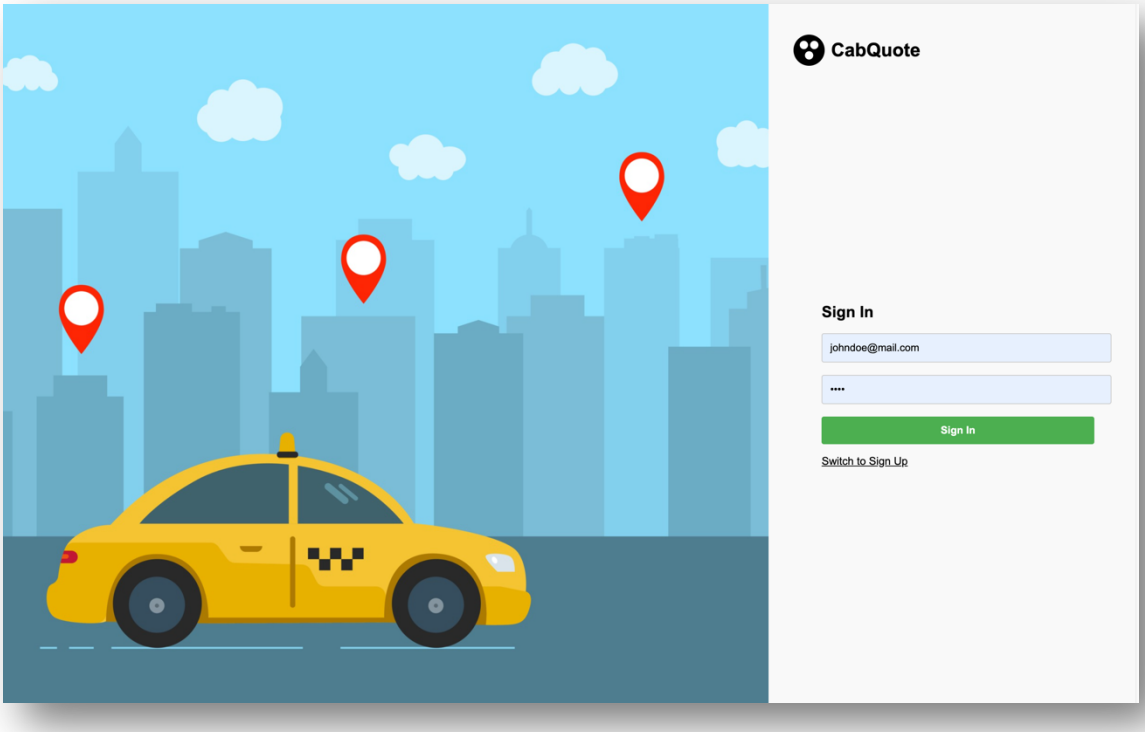

Figure 9: Login Page

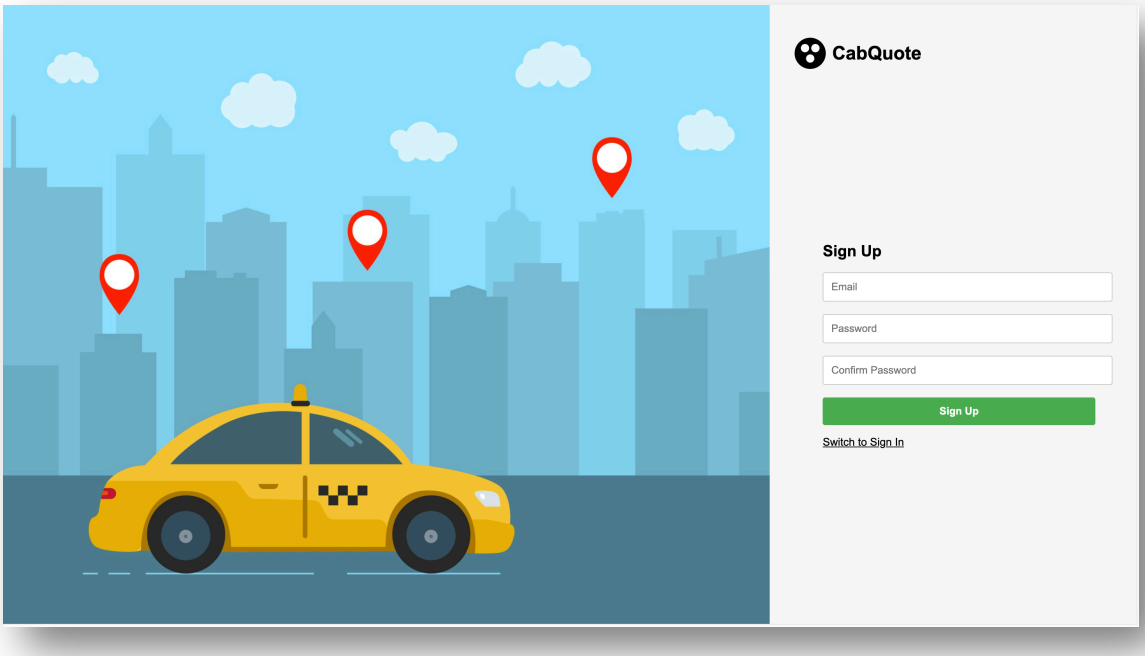

Figure 10: Login Page(Sign up tab)

App.vue page: Screenshot 3 presents the main interface of the application, accessible upon successful login. A navigation bar at the top facilitates access to various application sections. The central portion of the page contains input fields for pickup and dropoff locations and a button to initiate the cab prices comparison. Once the comparison is completed, a table displays the results, allowing users to compare prices and make informed choices. The design is user-centric, and the layout promotes easy navigation and engagement. Due to last minute scope change time constraints, the UI design is basic and emphasis more on the functionality of the application.

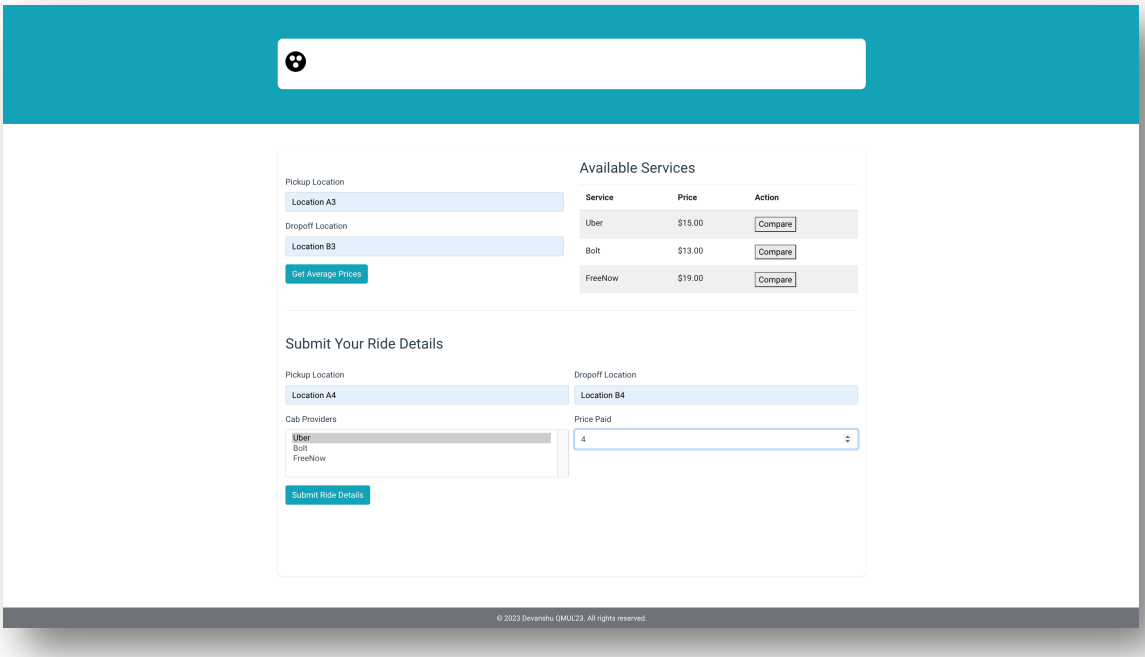

Figure 11: App.vue UI page

The application has been designed to be responsive, ensuring compatibility with a range of devices, including smartphones, tablets, and desktop computers. This flexibility caters to users' diverse requirements, making the application more inclusive.

Throughout the design phase, we prioritized best practices for UI design, emphasizing simplicity, clarity, and consistency. By crafting an intuitive interface focused on the user experience, we believe our application will effectively support users in comparing cab prices and making well-informed decisions.

4.4 API Design and Integration

At the beginning of the development of the application, extensive research was conducted on the integration of various APIs to access data and services from various sources. The objective was to facilitate seamless communication between our application and ride-hailing platforms so that users could compare prices in real time.

Several potential APIs were evaluated, including those provided by prominent ride-hailing services like Uber, Bolt, and FreeNow. This research permitted to identify the requirements, constraints, and opportunities for integrating APIs into our application.

However, as the project's scope evolved, it was ultimately decided not to integrate all these APIs. Instead, we concentrated on retrieving data from our MySQL database to provide consumers with the necessary information. Using the Express.js framework, we developed RESTful API endpoints to enable communication between the front-end and back-end of our application. Utilising the Axios library for HTTP requests allowed us to efficiently interact with external APIs and retrieve the necessary data.

Using these API endpoints, we were able to query the MySQL database and retrieve information about taxi fares, distances, and locations. For instance, when a user selects a pickup and drop-off location within the application, the front-end transmits a request to the back end containing the location information. The back end then queries the database and returns the corresponding data to the front-end, which displays it in a tabular format to facilitate comparison.

Here is an example of a taxi fare API endpoint created:

```
app.get('/api/prices', async (req, res) => {
    const { pickupLocation, dropoffLocation } = req.query;

    try {
      const prices = await getPrices(pickupLocation, dropoffLocation);
      res.status(200).json(prices);
    } catch (error) {
      res.status(500).json({ message: 'An error occurred while fetching prices.' });
    }
  });
```

Figure 12: Taxi Fare API endpoint code snippet

The integration of these APIs into our application allowed us to establish efficient communication between the front-end and back-end components. This approach ensured a responsive user experience while enabling the application to adapt to various ride-hailing service providers and data sources in the future.

Chapter 5: **Implementation**

In Chapter 5, we explore the implementation details of the CabQuote application, including the development environment setup, frontend implementation, backend implementation, frontend and backend integration, and relevant code snippets and screenshots.

5.1 Development Environmental Setup

A suitable development environment was set up to accommodate the frontend, backend, and database implementations needed to build CabQuote. The project employed the following technologies and tools:

1. **Node.js**: Node.js is a framework that supports server-side scripting and the development of scalable applications. It is based on Chrome's JavaScript runtime.
2. **Visual Studio Code**: Visual Studio Code is a quick and flexible IDE that comes with built-in support for JavaScript, Node.js, and several additional programming languages, extensions and frameworks.
3. **Express.js**: Express.js is a Node.js web application framework that makes it simple and effective to create backend APIs.
4. **Vue.js**: Vue.js is a progressive JavaScript framework that makes it possible to create single-page applications and user interfaces.
5. **SQL database** – A relational database management system that stores and organises data in an organised manner.

To make the development process easier, all dependencies were installed on the local development workstation. Git, a distributed version control programme, was used to maintain the project's codebase. Git made it possible to manage branches, track changes, and, if necessary, collaborate with other developers

5.2 Frontend Implementation

Vue.js, a versatile and effective JavaScript framework perfect for building single-page applications, was used to design CabQuote's frontend. The homepage page, login page, and App.vue page are the three main parts of the frontend as mentioned earlier.

Landing page: This element acts as the user's first point of contact, giving them an overview of the application and encouraging them to log in or register. The page was styled using Vue.js and a unique CSS stylesheet to adhere to the design requirements.

Login page: Users can access CabQuote using this feature by entering their email address and password.

The page was made with Vue.js, and form validation is used to ensure the proper input format.

App.vue page: After successfully logging in, you can view the main programme interface, which has input fields for the pickup and dropoff locations and a button to start comparing cab prices. When finished, a table shows the outcomes. With the aid of Vue.js and unique CSS styles, the App.vue component was created.

A submission form at the bottom of the page allows the user to contribute the price paid between two locations, which auto-fills in the database to calculate average quotes.

To provide a seamless and responsive user experience, the frontend also includes several utility features and auxiliary components. These routines take care of things like data retrieval, data manipulation, and form validation.

```
<template>
  <div class="login">
    <h1>Login</h1>
    <input v-model="email" type="email" placeholder="Email" />
    <input v-model="password" type="password" placeholder="Password" />
    <button @click="handleLogin">Login</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      email: "",
      password: "",
    };
  },
  methods: {
    handleLogin() {
      // Code to handle login process goes here
    },
  },
};
</script>
```

Figure 13 Login Page Code snippet

5.3 Backend Implementation

Node.js and the Express.js framework, which enable the effective construction of APIs and handling of HTTP requests, were used to implement the backend. The management of data storage and retrieval in the SQL database, as well as administration of front-end interaction, are the main responsibilities of the backend. The following essential components comprise the backend:

Server setup: Node.js and Express.js were used to configure the server to listen on a specific port for incoming requests. The ports used in this project on the local machine were localhost:8080, localhost:3000 and localhost:4000.

API endpoints: RESTful API endpoints were developed to make it easier for frontend and backend communication. These endpoints provide functions like user authentication, data fetching, and data modification while supporting the retrieval and storage of data.

Database connection: The SQL database was connected using a database.js file, which also provided query functions for data processing. The use of SQL queries for operations like user registration, user authentication, and data retrieval is made possible by this file.

Data generator: To create sample data for testing and development reasons, a dummy-data-generator.js file was made.

```
  const express = require('express');
const router = express.Router();
const { check, validationResult } = require('express-validator');
const authController = require('../controllers/authController');

router.post('/login', [
    check('email').isEmail().withMessage('Invalid email address'),
    check('password').isLength({ min: 6 }).withMessage('Password must be at least 6 characters long'),
  ], authController.login);

module.exports = router;
```

Figure 14: API endpoint(backend/server.js)

5.4 Integration of Frontend and Backend

The following techniques were used to integrate both the front and backend of CabQuote:

API requests: To carry out operations like user authentication and data retrieval, the frontend sends HTTP requests to the backend's API endpoints. The popular promise-based HTTP client axios is used to handle these requests in JavaScript applications. Axios makes processing requests and responses simpler, ensuring effective and trustworthy communication between the front end and back end.

Data transfer: The backend responds to API requests by processing them and, if necessary, contacting the database. The frontend then provides the outcome, which is frequently in JSON format. With its reactive data-binding framework, Vue.js makes sure that the front end shows the most recent data.

Authentication: User authentication is a critical component of CabQuote, ensuring that only registered users can access the platform's features. A session token is generated and provided to the frontend after a successful login in the backend using email and password verification. Access control and session management can be implemented since the frontend stores the token and uses it in subsequent calls to the backend.

```
methods: {
  async handleLogin() {
    try {
      const response = await axios.post('/api/auth/login', {
        email: this.email,
        password: this.password,
      });
      // Save session token and redirect to the main application
    } catch (error) {
      // Handle login errors
    }
  },
},
```

Figure 15 code snippet of an axios request to and API endpoint

5.5 Source Code

The source code for CabQuote was developed using a combination of programming languages and libraries. To demonstrate the structure and organization of the project, screenshots of key components have been included. In this section, we will showcase the code for the Login.html, App.vue, and Landing Page files.

5.5.1 Landing Page (CabQuote _ Compare and Book Cabs.html)

The Landing Page file contains the HTML, JavaScript, and CSS required for rendering the landing page of the application. It serves as the first page users see when accessing the CabQuote platform, providing an overview of the service and prompting users to either log in or sign up. The following figure is a snippet from the Landing Page code:

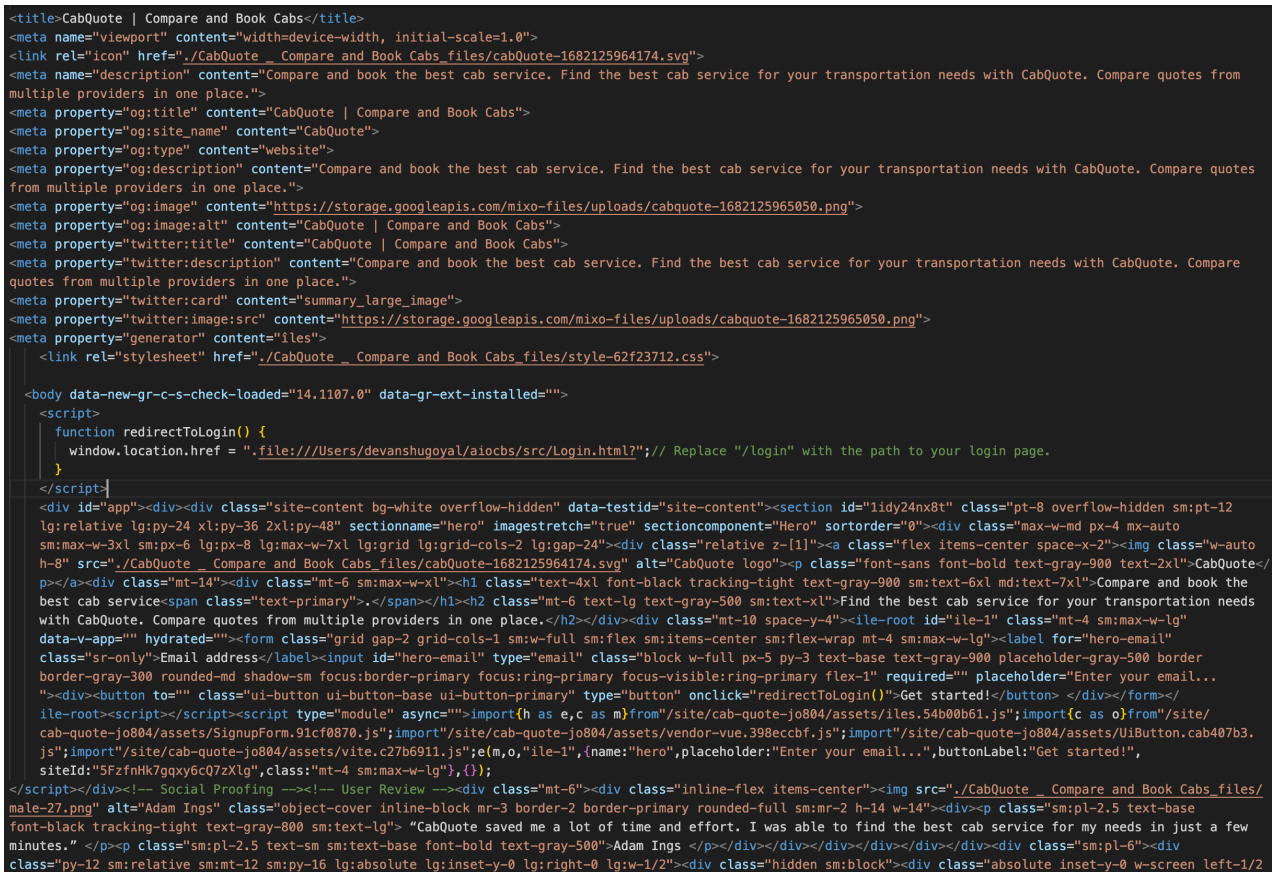

Figure 16: Landing page source code snippet

5.5.2 Login.html

The Login.html file is a part of the frontend and contains the HTML template, JavaScript code, and CSS styles for the login page. It utilizes Vue.js to handle user input and trigger the login process. The following fiugre shows the code snippet for the Login.html file:

```
<body>
    <div class="container">
        <div class="image-section"></div>
        <div class="form-section">
            <div class="header">
                <img src="./assets/CQlogo.svg" alt="CabQuote Logo">
                <h1>CabQuote</h1>
            </div>
            <div class="form">
                <form class="login-form" action="http://localhost:4000/login" method="post">
                    <h2 id="form-header">Sign In</h2>
                    <input type="email" name="email" placeholder="Email">
                    <input type="password" name="password" placeholder="Password">
                    <button>Sign In</button>
                </form>
                <form class="signup-form" action="http://localhost:4000/register" method="post" style="display: none;">
                    <h2>Sign Up</h2>
                    <input type="email" name="email" placeholder="Email">
                    <input type="password" name="password" placeholder="Password">
                    <input type="password" placeholder="Confirm Password">
                    <button>Sign Up</button>
                </form>
                <p class="form-toggle" id="form-toggle">Switch to Sign Up</p>
            </div>
        </div>
    </div>

    <script>
        const formToggle = document.getElementById('form-toggle');
        const formHeader = document.getElementById('form-header');
        const loginForm = document.querySelector('.login-form');
        const signupForm = document.querySelector('.signup-form');
        let isLoginForm = true;

        formToggle.addEventListener('click', () => {
            isLoginForm = !isLoginForm;
            if (isLoginForm) {
                formHeader.innerText = 'Sign In';
                formToggle.innerText = 'Switch to Sign Up';
                loginForm.style.display = 'block';
                signupForm.style.display = 'none';
            } else {
                formHeader.innerText = 'Sign Up';
                formToggle.innerText = 'Switch to Sign In';
                loginForm.style.display = 'none';
                signupForm.style.display = 'block';
            }
        });


        async function onSubmit(event) {
  event.preventDefault();
```

Figure 17: Login page source code snippet

5.5.3 App.vue

The App.vue file is the main application component, serving as the root element and container for all other components. It handles navigation and includes various components, such as the header, footer, and content sections. The screenshot below demonstrates the organization of the App.vue file:

```
<template>
  <div id="app">
    <nav class="navbar navbar-expand-lg navbar-dark bg-info">
      <div class="container">
        <a class="navbar-brand" href="#">CabQuote</a>
        <button
          class="navbar-toggler"
          type="button"
          data-toggle="collapse"
          data-target="#navbarNav"
          aria-controls="navbarNav"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav">
            <li class="nav-item active">
              <a class="nav-link" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">About</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
    <div class="container mt-5">
      <div class="row">
        <div class="col-md-6">
          <form>
            <div class="form-group">
              <label for="pickup-location">Pickup Location</label>
              <input
                type="text"
                class="form-control"
                id="pickupInput"
                placeholder="Enter pickup location"
                ref="pickupInput"
                v-model="pickupLocation"
              />
            </div>
            <div class="form-group">
              <label for="dropoff-location">Dropoff Location</label>
              <input
                type="text"
                class="form-control"
                id="dropoffInput"
                placeholder="Enter dropoff location"
                ref="dropoffInput"
                v-model="dropoffLocation"
              />
            </div>
            <button type="submit" class="btn btn-info" @click.prevent="onSubmit">Get Average Prices</button>
```

Figure 18 App.vue <template> source code snippet

```
<script>
/* global google */
export default (await import('vue')).defineComponent({
  name: 'App',
  data() {
    return {
      showAveragePrices: false,
      apiReady: false,
      pickupLocation: '',
      dropoffLocation: '',
      cabServices: [],
      priceDifferences: [],
    };
  },

  methods: {


    selectService(cabService) {
      // Add logic for handling selected cab service
      alert(`Selected service: ${cabService.name} - Average Price: ${cabService.averagePrice}`);
    },

    currency(value) {
      return '$' + parseFloat(value).toFixed(2);
    },

    initAutocomplete() {
      if (!this.apiReady) return;

      const initAutocompleteInterval = setInterval(() => {
        if (google && google.maps && google.maps.places) {
          clearInterval(initAutocompleteInterval);

          const pickupInput = this.$refs.pickupInput;
      const dropoffInput = this.$refs.dropoffInput;

      new google.maps.places.Autocomplete(pickupInput);
      new google.maps.places.Autocomplete(dropoffInput);
    }
  }, 100);
},
async getAveragePrices() {
  try {
    const pickupLocation = this.pickupLocation.replace(/\+/g, ' ');
    const dropoffLocation = this.dropoffLocation.replace(/\+/g, ' ');
    const response = await fetch(
      `http://localhost:3000/prices?pickup=${encodeURIComponent(pickupLocation)}&dropoff=${encodeURIComponent(dropoffLocation)}`
    );
    if (!response.ok) {
      throw new Error("Error fetching prices");
    }
    const prices = await response.json();
    console.log("Received response from server:", prices); // Added this line
    console.log("Prices from server:", prices);
```

Figure 19 App.vue <script> source code snippet

Chapter 6: **Testing**

The focus of Chapter 6 is the testing process for CabQuote, including the test plan, test cases, test execution and results, as well as problems encountered and their solutions. This chapter ensures that the application's requirements are met and that it functions properly.

6.1 Test Plan

A comprehensive test plan was developed for CabQuote to ensure that the platform functioned as expected, providing a seamless user experience. The plan included testing different components of the application, such as frontend, backend, and integration between the two, to identify and resolve any potential issues. Various testing methodologies were utilized, including functional testing, performance testing, and security testing. The plan also consisted of a series of test cases designed to validate the platform's functionality, performance, and security.

6.2 Test Cases

The test cases were created to cover key aspects of CabQuote, such as user authentication, navigation, cab quotation, and database interactions. Some example test cases include:

1. **Test Case: Verify user login**.

- Objective: Ensure that users can successfully log in with valid credentials.
- Test Steps:
    a) Access the login page.
    b) Enter a valid email and password.
    c) Click the login button.

- Expected Result: The user is successfully logged in and redirected to the main application.

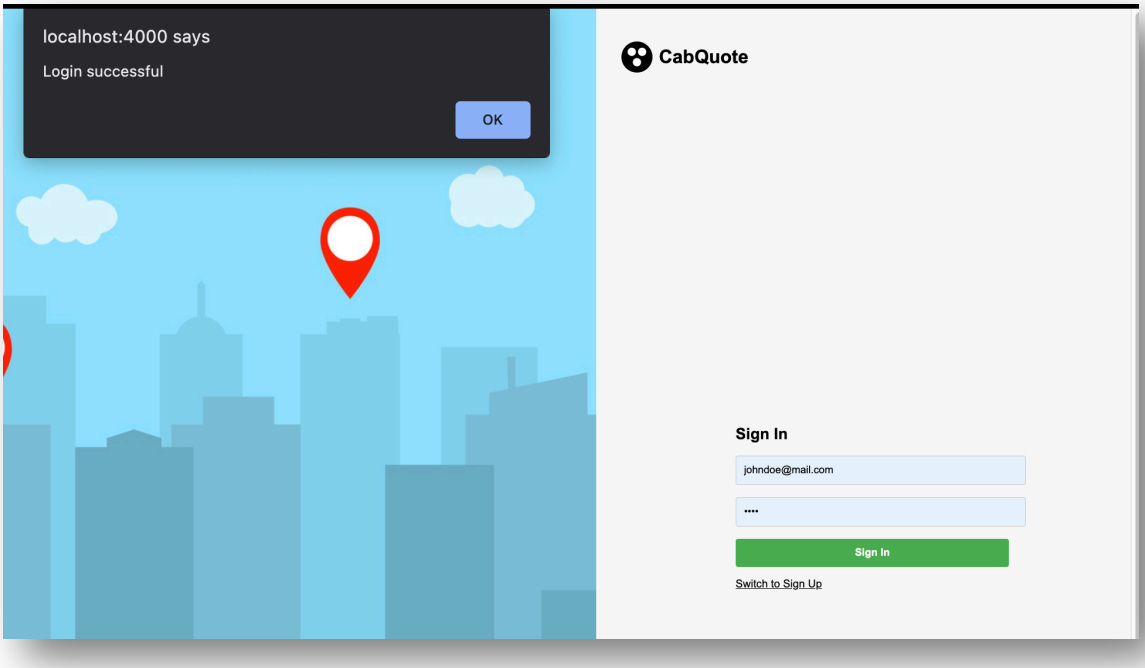

Figure 20 : Success Login Page

2. **Test Case: Verify invalid login attempts.**

- Objective: Ensure that the system does not allow access with invalid credentials.
- Test Steps:
    a) Access the login page.
    b) Enter an incorrect email or password.
    c) Click the login button.

- Expected Result: The system displays an error message indicating invalid credentials.

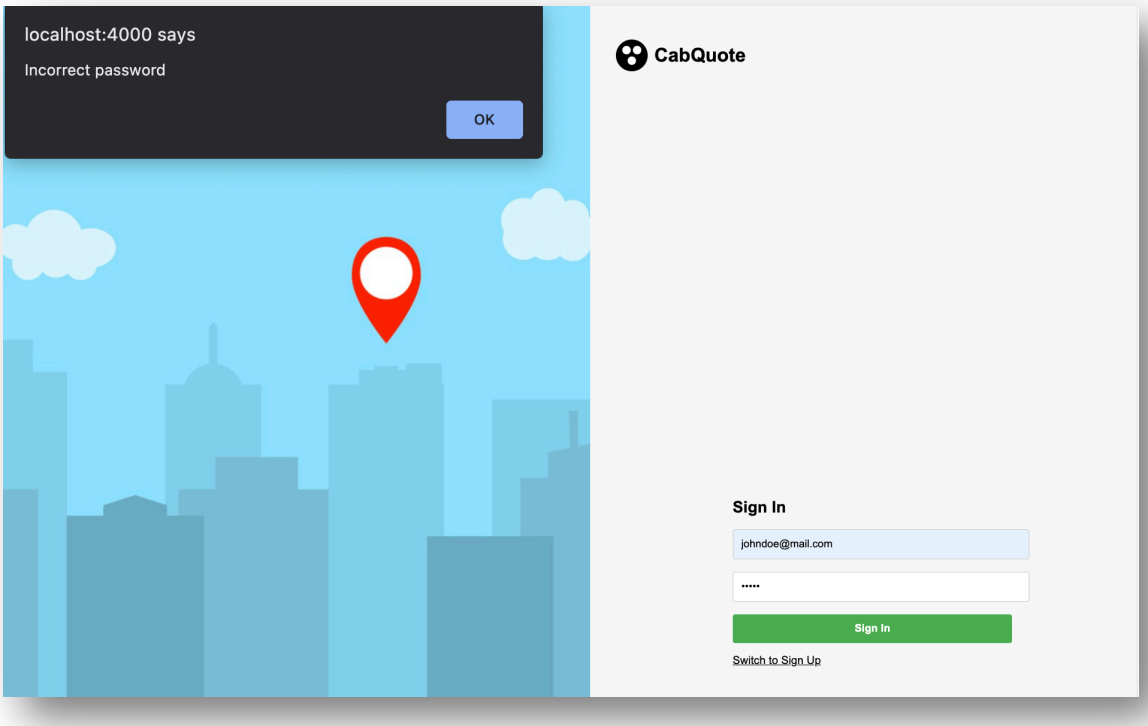

Figure 21: Unsuccessful Login Page

3. **Test Case: Verify cab quotation functionality.**

- Objective: Ensure that users can obtain accurate cab quotes based on their input.
- Test Steps:
    a) Access the main application page.
    b) Enter pickup and dropoff locations, along with the distance between them.
    c) Click the 'Get Quote' button.

- Expected Result: The system displays a list of available cabs and their respective average quotes based on the input. For testing, we have multiple entries between *Location A3 and B3* in the database, the average price of all the entries corresponds with the output data on the app.vue page.

- Uber:(14.00 + 15.00 + 16.00) / 3 = $15.00
- Bolt:(12.00 + 13.00 + 14.00) / 3 = $13.00
- FreeNow:(18.00 + 19.00 + 20.00) / 3 = $19.00

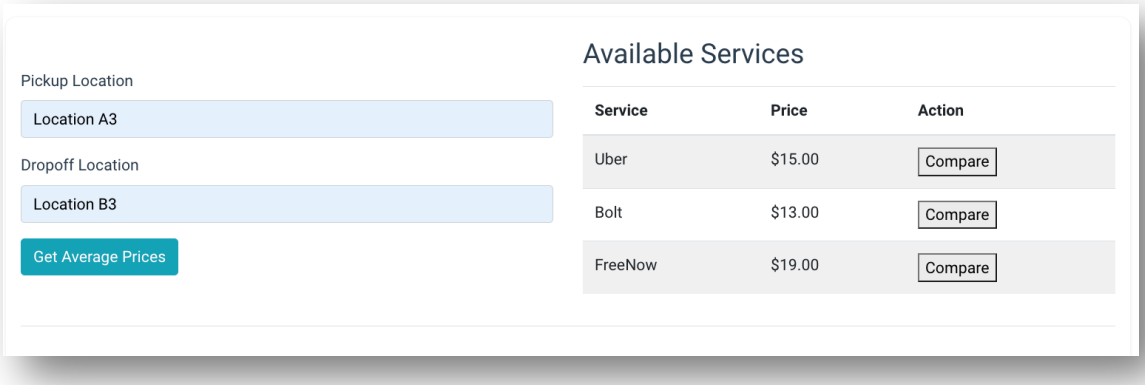

Figure 22: Cab average price Quotation

id	cab_name	distance	price	pickup_location	dropoff_location
2014	Uber	10.00	14.00	Location A3	Location B3
2015	Uber	10.50	15.00	Location A3	Location B3
2016	Uber	11.00	16.00	Location A3	Location B3
2017	Bolt	15.00	12.00	Location A3	Location B3
2018	Bolt	15.50	13.00	Location A3	Location B3
2019	Bolt	16.00	14.00	Location A3	Location B3
2020	FreeNow	20.00	18.00	Location A3	Location B3
2021	FreeNow	20.50	19.00	Location A3	Location B3
2022	FreeNow	21.00	20.00	Location A3	Location B3

Figure 23: SQL database set for Location A3 and B3

6.3 Test Execution and Results

The test cases were executed according to the test plan, and the results were documented. Overall, the CabQuote platform functioned as expected, with most test cases passing. The user authentication process was secure and reliable, while the cab quotation functionality returned accurate results. Database interactions were also tested, and the system was able to store and retrieve data correctly.

However, during the testing process, a few issues were identified that required attention. These issues are detailed in the next section, along with their resolutions.

6.4 Issues Faced and Resolution

During the testing process, several issues were encountered and subsequently resolved. Some of these issues include:

1. **Issue: Inaccurate quotes**
   Resolution: The issue was traced back to incorrect calculations within the backend logic. The calculation formula was revised, and the issue was resolved.

2. **Issue: Database connection errors**
   Resolution: Incorrect configuration settings in the database connection were to blame for the errors. The settings were updated, and the connection was re-established successfully.

3. **Issue: UI inconsistencies across different devices**
   Resolution: CSS media queries were employed to ensure that the user interface would adapt to different screen sizes and resolutions, providing a consistent experience across devices.

These issues were addressed and resolved, resulting in an improved CabQuote platform that adhered to the project requirements and functioned as intended.

Chapter 7: **Evaluation**

This chapter evaluates the CabQuote application's usability, user interface, performance and scalability, and security measures. This evaluation provides insight into the application's efficacy in achieving its goals.

7.1 Application Usability

CabQuote is meant to provide a simple and user-friendly experience, allowing users to rapidly find and compare average cab fares between major cab providers in their area. The interface is designed to promote simple interactions, with an emphasis on meeting the needs of the user. The software allows users to easily enter their desired pickup and drop-off locations, offering accurate fare comparisons tailored to their specific needs. Furthermore, the application's easy navigation ensures that users can easily explore various features and pages.

Supervisor feedback was critical to fine-tuning CabQuote's usability throughout the development process. Continuous testing and user input indicated areas for improvement, resulting in a better overall user experience. The finished product demonstrates the importance of user-centric testing in ensuring that the platform is personalised to users' requirements and preferences.

7.2 User Interface Evaluation

The visual design of CabQuote adheres to modern design principles, employing a minimalist approach that promotes a clutter-free interface, applying principles from final year UX design module. The colour scheme, typeface, and layout all contribute to a unified and visually enticing user interface. By minimising visual distractions and emphasising key information, users can focus on the primary task of searching and comparing taxi fares.

Throughout the development process, the efficacy of the user interface was evaluated through user testing. This feedback provided insight into prospective interface enhancements. Subsequently, the development team iterated on the design, refining it to produce an interface that was user-friendly and simple to comprehend.

7.3 Performance and Scalability

Performance and scalability were crucial factors in the development of CabQuote. By adhering to best practises for web application design, both frontend and backend components were optimised to ensure optimal performance. Therefore, the platform can accommodate increased user traffic and efficiently manage user requests.

Scalability is essential for the application to accommodate future growth and fluctuating user demands. The modular architecture of CabQuote enables the platform to incorporate new features and manage an increased workload without compromising the user experience.

Under varying conditions, the CabQuote application was subjected to load testing to evaluate its performance under stress. These evaluations validated the platform's capacity to manage high traffic volumes while maintaining a rapid response time.

7.4 Security Measures

Throughout development, protecting user data and ensuring the overall security of the CabQuote application were top priorities. By attempting to adhere to industry-standard security practises, the developer implemented a variety of measures to reduce potential vulnerabilities. These measures included the use of secure communication protocols, thorough input validation, and adherence to secure coding practises.

The developer took a proactive approach to application security by continuously evaluating the codebase and infrastructure for potential security vulnerabilities.

This vigilance allowed for the expeditious identification and correction of security flaws, ensuring that user data remains secure, and the application is resistant to prevalent security-threats.

7.5 Future Enhancements

As a platform designed to accommodate evolving requirements, CabQuote has ample space for future development. The modular architecture permits the addition of new features, thereby expanding the application's capabilities as user needs and market conditions evolve.

A potential improvement would be the incorporation of additional cab service providers, thereby expanding the available fare comparisons. This would provide consumers with more comprehensive information, allowing them to select a taxi service with greater knowledge.

Real-time traffic data could also be incorporated, allowing for more accurate fare estimates and travel durations. By providing users with current traffic information, CabQuote can enhance its value proposition.

In addition, the CabQuote platform could be expanded to include reserving and payment options within the app. This would improve the user experience by enabling users to not only search and compare fares but also schedule and pay for their preferred taxi service within the application. This seamless integration of services would make the user experience more convenient and efficient.

In conclusion, the user interface and overall user experience could be enhanced. The developer can ensure that CabQuote remains a visually appealing and user-friendly platform by incorporating user feedback and keeping up with design trends.

CabQuote was created with an emphasis on usability, performance, security, and scalability. The platform has undergone extensive testing and evaluation to ensure that it satisfies the requirements of its users, both in its current state and as it evolves in the future. Therefore, CabQuote is a valuable resource for users who wish to search and compare taxi fares efficiently and swiftly.

Chapter 8: **Conclusion**

Chapter 8 concludes the project by summarising key learnings and accomplishments, discussing future enhancements and limitations, and offering concluding thoughts on the development of the CabQuote application and its prospective impact on users.

8.1 Key Learning and Achievements

Throughout the duration of this project, numerous important discoveries and accomplishments have been achieved. The development of the CabQuote platform has resulted in a deeper understanding of various web technologies, frameworks, and tools, including Vue.js, Express, and SQL. The project has demonstrated the significance of appropriate planning, project management, and team communication.

The CabQuote platform has achieved its goal of providing users with a method to compare average taxi fares from various providers, based on a database populated by user contributions. The system design and implementation processes were carried out methodically, considering a variety of factors including user experience, system performance, and security measures. The project has demonstrated the viability of combining multiple APIs and leveraging technology to develop a consumer-friendly and beneficial application.

8.2 Future Improvements

Although CabQuote demonstrates its fundamental functionality and achieves its goals, there is always place for improvement. Potential platform enhancements include the following:

1. Expansion of the list of cab companies to include more options for users, encompassing a broader range of locations and providing a more thorough comparison.
2. Integration of real-time traffic data to enhance the precision of taxi fare and travel time estimates.
3. Implementation of a user review and rating system to assist users in making better judgements based on the experiences of other users.
4. Improving the user interface with more intuitive navigation, filtering options, and customization options to provide a more individualised experience.
5. Exploration of machine learning and natural language processing (NLP) algorithms for improved prediction and recommendation of taxi options, considering user preferences, historical data, and travel patterns into account.

8.3 Final thoughts

The CabQuote project has been a valuable and challenging learning experience. It has provided an opportunity to apply the knowledge and skills acquired throughout the computer science degree in a real-world context. This project has served as a testament to the potential of technology to make a positive impact on people's lives by simplifying everyday tasks and providing more accessible information.

As technology continues to evolve rapidly, the possibilities for further improvements and advancements in the field of cab comparison platforms are vast. With the growing trend of smart cities and the Internet of Things, applications like CabQuote can play a crucial role in enhancing urban mobility and transportation options for individuals.

8.4 Limitations

While the CabQuote platform has accomplished its primary goals, it is important to recognise its limitations. Firstly, the project's current scope is based on fictional locations and a handful of taxi companies. Extending coverage to real-time locations and providers would necessitate substantial effort and resources, such as database updates and the integration of additional APIs.The functionality of the application based on fictitious locations has been demonstrated to be applicable in real-world situations.

Secondly, the accuracy of the average taxi prices and location is dependent on the data provided by registered users. Any discrepancies or inaccuracies in these data sources may impact the quality of the user's displayed results.

Lastly, the project was concluded within a constrained time frame, which prevented the exploration of more advanced features and optimisations. This limitation also reduced the time available for extensive testing and user feedback, which could have led to additional platform refinements and enhancements.

Appendix A: Project Milestones

Seven key development milestones of app development which are essential for a successful delivery of the project.

1. **Research:** Specify the key idea and a minimum viable product at the end of the lifecycle. Researching the current market while keeping the end user's and other stakeholders in mind.
2. **Wireframing**: Documenting and wireframing the application, this includes drawing detailed sketches of the envisioned product which helps uncover usability issues. Post sketching, wireframing the app refines the ideas and arrange all the components of the design in the right way. Any technical limitation encountered can be overcome in this initial phase.
3. **Technical feasibility assessment**: At this milestone, one individual might have a clear understanding of core functionalities and vision of the project but needs to make sure that the back-end system's will be able to support the app's functionality. To check if the application is technically feasible, one needs to get access to public data by sourcing public API's.
4. **Prototype**: Building a rapid prototype is essential for the project lifecycle. Getting the app concept into user's hands as quickly as possible is crucial to encounter any common user case scenarios, by using rough wireframes for the product.
5. **Design**: Application design will prove to be a multi-step process, but it gives a clear direction for the development phase of the product. It majorly involves envisioning, designing and minimum achievements expect from the product.
6. **Development**: At this step, one take is the programming requirements and then proceed with the iterative process of coding, testing, revisiting, and retesting once all the bugs and issues are addressed,
7. **Final deployment** – This is the final stage of the product lifecycle where the product is finally delivered to the end users. This phase involves extended product lifecycle, hyper care and extended support required for the product after implementation.

Appendix B: Risk Analysis

Description of risk	Description of Impact	Probability	Impact Score	Mitigation plan
Potential risk of poor UI/IX design feedback	Poor UI/UX design feedback can degrade customer satisfaction and usage	Low	High	Making sure with the development phase to cross check the interface designs with Supervisor
Unreliable Virtual Machine host system	Crash in the virtual machine host provider causing severe delays in processing application	Low	High	Choosing the most reliable VM with positive feedback in the market
App market rejection	Rejection by App store/ Play store due to regulatory issues	Low	High	Cross check the regulatory policies during testing phase
Loss of confidentiality	Application plan leakage by external/internal members of the project	Low	Medium	Classifying the information based on level of confidentiality and sharing them accordingly
Loss of sponsor funding/ Project termination	Project aborted due to budget/ feasibility of the sponsors (Supervisors)	Low	High	Clear communication plan with the sponsor/Supervisor
Lack of team support	Misunderstanding and coordination issues if any potential team forms	Low	Medium	Motivating the team by reminding them objectives and end goals of the project
Time estimation	Running over/under the time for the project cycle	Medium	Medium	Following strict agile methodology tracking by assessing the milestones

Bibliography

Danziger, E. (2020, october). *Announcing the Google Maps Platform On-demand Rides & Deliveries solution*. Retrieved from google cloud: https://cloud.google.com/blog/products/maps-platform/announcing-google-maps-platform-demand-rides-deliveries-solution

VMware. (2019). *What is virtualization?* Retrieved from vmware.com: https://www.vmware.com/uk/solutions/virtualization.html#:~:text=Virtualization%20relies%20on%20software%20to,of%20scale%20and%20greater%20efficiency.

WikiPedia. (n.d.). *Citymapper*. Retrieved from wikipedia: https://en.wikipedia.org/wiki/Citymapper

Grand View Research. (2021). Ride Sharing Market Size, Share & Trends Analysis Report By Service Type (E-hailing, Car Sharing, Station-based Mobility, Car Rental), By Region, And Segment Forecasts, 2021 - 2028. **https://www.grandviewresearch.com/industry-analysis/ride-sharing-market**

Kunst, A. (2023). Ride-sharing online taxi usage by brand in the United Kingdom (UK) in 2022. Statista. **https://www.statista.com/statistics/769756/most-used-ride-sharing-online-taxi-apps-by-brand-uk/**

Danziger, A. (2020). Google Maps vs. Citymapper: The Ultimate Showdown of Mapping Apps. The Points Guy. Retrieved from https://thepointsguy.com/guide/google-maps-vs-citymapper/

Gerpott, T. J., & Paukert, M. (2019). Ride-hailing service adoption: An empirical analysis of decisive factors. Transportation Research Part A: Policy and Practice, 123, 1-16.

Grand View Research. (2021). Ride-Hailing Service Market Size, Share & Trends Analysis Report By Service Type (E-hailing, Car Sharing, Station-based Mobility), By Vehicle Type, By Region, And Segment Forecasts, 2021 - 2028. Retrieved from https://www.grandviewresearch.com/industry-analysis/ride-hailing-service-market

Liu, J. (2018). A comparison of popular web development frameworks: Angular, React, and Vue. Journal of Digital & Social Media Marketing, 6(2), 189-201.

Molla, R., & Fizazi, A. (2018). Understanding consumer behavior in the sharing economy: A review of research on Airbnb, Uber, and Lyft. In The Sharing Economy: Perspectives, Opportunities, and Challenges (pp. 1-22). IGI Global.

Varshney, M. (2019). Full Stack Web Development with Angular and Spring MVC. Packt Publishing Ltd.

Wang, M., Wang, Q., Zhang, C., & Zhao, X. (2020). The effect of price promotions on consumer adoption of ride-hailing services: An empirical study of Didi Chuxing. Transportation Research Part A: Policy and Practice, 132, 467-478.

Wikipedia. (n.d.). Citymapper. Retrieved from https://en.wikipedia.org/wiki/Citymapper

JUnit, "Getting Started," **https://junit.org/junit5/docs/current/user-guide/** .

Selenium, "Introduction," **https://www.selenium.dev/documentation/en/introduction/**

TestRail, "Writing Test Cases: Tips and Tricks," **https://www.gurock.com/testrail/test-management/writing-test-cases-tips-and-tricks**

IEEE Standard 829-2008, "IEEE Standard for Software and System Test Documentation," **https://standards.ieee.org/standard/829-2008.html**.

Mozilla Developer Network, "Using Media Queries," **https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries**

Vue.js. (n.d.). Vue.js - The Progressive JavaScript Framework. Retrieved from https://vuejs.org/

Axios. (n.d.). Axios - Promise based HTTP client for the browser and node.js. Retrieved from https://axios-http.com/

BootstrapVue. (n.d.). BootstrapVue - Bootstrap v4 Components and Grid System for Vue.js. Retrieved from https://bootstrap-vue.org/

Node.js. (n.d.). Node.js - JavaScript runtime built on Chrome's V8 JavaScript engine. Retrieved from https://nodejs.org/

Express. (n.d.). Express - Fast, unopinionated, minimalist web framework for Node.js. Retrieved from https://expressjs.com/

MySQL. (n.d.). MySQL - The world's most popular open source database. Retrieved from https://www.mysql.com/

NPM. (n.d.). NPM - Package manager for JavaScript and the world's largest software registry. Retrieved from https://www.npmjs.com/

JWT. (n.d.). JSON Web Tokens - Introduction. Retrieved from https://jwt.io/introduction/

Google Maps Platform. (n.d.). Google Maps Platform - Build with real-time, comprehensive data. Retrieved from https://developers.google.com/maps