

PROFESSIONAL TRAINING REPORT - II

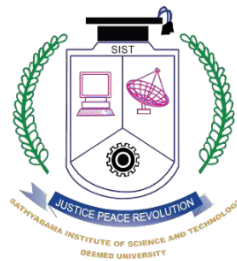
entitled

A CAB FARE PREDICTION APPLICATION BASED ON REACT JS FRAMEWORK, MAPBOX WITH SECURE AUTHENTICATION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Cyber Security

by

**HANNAH HARSHITHA [41614030]
SANJAY SARAVANAN N [41614084]
PRANATHI CHOWDARY ATLURI [41614073]**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by
AICTE**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of Ms. HANNAH HARSHITHA M (41614030) who carried out the project entitled "A CAB FARE PREDICTION APPLICATION BASED ON REACT JS FRAMEWORK, MAPBOX WITH SECURE AUTHENTICATION" under my supervision from January 2024 to May 2024.

 7/5/24

Internal Guide

Dr. VEENA .K, M.E., Ph.D.,



Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voice Examination held on 7/5/24

 7/5/24

Internal Examiner

 7/5/24

External Examiner

DECLARATION

I HANNAH HARSHITHA M (41614030), hereby declare that the Professional Training Report-II entitled "A CAB FARE PREDICTION APPLICATION BASED ON REACT JS FRAMEWORK, MAPBOX WITH SECURE AUTHENTICATION" done by me under the guidance of Dr. VEENA .K, M.E.,Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE: 7/5/24

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board **of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. K. Veena, M.E.,Ph.D.,** for his/her valuable guidance, suggestions and constant encouragement which paved the way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



Elite

NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to
MUDDA HANNAH HARSHITHA
for successfully completing the course

Cloud Computing and Distributed Systems

with a consolidated score of **62** %

Online Assignments	18.75/25	Proctored Exam	43.5/75
--------------------	----------	----------------	---------

Total number of candidates certified in this course: **3244**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2024

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL24CS09S444102721

To verify the certificate



No. of credits recommended: 2 or 3

ABSTRACT

In the realm of modern transportation services, the accurate estimation of cab fares is indispensable for both users and service providers. This application represents a solution for dynamic pricing and visualization that meets the needs of transportation companies and users. At its core, the app is designed to simplify the process of calculating prices based on selected vendors and locations, while also providing an intuitive way to view locations on paper. Using the React Context API for state management is great. This enables the integration and use of address information across different products, providing consistency and integration throughout the billing process. Leveraging the power of React, the app can update price estimates in real-time as users update their customer preferences and location selections. Mapbox integration for interactive map visualization. This integration allows users to view selected locations on a map, allowing them to better understand the area around the billing system. Through the use of icons and layers, users can easily identify and manage locations, making it easier to estimate costs and make decisions. Each component encapsulates specific features and functionality, increasing code reusability and supporting future scalability. This architecture also helps increase the responsiveness of your application, ensuring that the user gets the same experience across devices and screen sizes. Trust is good. The app ensures user satisfaction and engagement by prioritizing fast loading times and interactions using best practices and optimizations such as lazy loading and code allocation. Seamless API integration for geocoding and routing increases data accuracy and enables instant costing. This integration enables the app to receive and process location information efficiently, ensuring that the price prediction is based on the latest information available. and site visualization. With its user-friendly interface, seamless integration of mapping technology and strong architectural design, the application sets high standards for transportation software solutions and demonstrates its commitment to delivering value and innovation to users.

TABLE OF CONTENTS

CHAPTE R NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	4
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	7
	3.2.1 Hardware Requirements	9
	3.2	
	3.2.2 Software Requirements	9
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	10
	4.1.1 Ideation Map/Architecture Diagram	12
	4.1.2 Stages in Cab Prediction	13
	4.1.3 Internal or Component design structure	15
	4.1.4 working principles	16
	4.2 4.2.1 Novelty of the Project	19
5	CONCLUSION	21
	REFERENCES	23
	APPENDIX	
	A. Research Paper	25
	B. Source Code	34

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	Installing Dependencies	11
4.2	Installing Clerk	11
4.3	Flow chart of the application	13
4.4	Functional Block Diagram	15

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

This project provides a React JS application designed to support dynamic calculations and visual fields and also includes an operator for authentication. The core of the app uses React's Context API to simplify state management, allowing users to access user preferences and select settings. This approach allows users to interact with the app and instantly update their options to get a cost estimate. See the selected location on the interactive map. This feature not only improves the user experience but also provides users with a visual view of their chosen location, helping them make accurate predictions. The application provides users with an intuitive and interactive interface by leveraging modern web mapping technology. Each part contains specific functions that support a clean and organized code base. This architectural approach not only increases the manufacturer's efficiency but also its ability to meet increasingly complex demands over time.

Essentially, the project collaborates with the Operator to provide users with useful and empowered capabilities. The app leverages Clerk's authentication services, ensuring only authorized users have easy access to business and information. This increases security and gradually increases user trust, thus improving the overall user experience. This is achieved through carefully designed and implemented design and materials that allow users to seamlessly access the app from devices such as desktop computers, tablets, and smartphones.

External API integration further improves the application performance, enabling cost-effective accounting and efficient planning. By using external services for data retrieval and processing, the application ensures that users receive up-to-date and reliable information, thus improving the overall results of the application. Thanks to its intuitive, modular design, and integration with external services, the application provides a powerful and convenient user experience that meets the needs of modern transportation.

The scope of this application encompasses the design, development, and

evaluation of a cab fare prediction application tailored to address the shortcomings of existing fare estimation methods. Central to the application's functionality is the utilization of modern web technologies, including the React JS framework and the mapping capabilities of Mapbox, to deliver an intuitive and seamless user experience. Additionally, stringent emphasis is placed on the implementation of robust user authentication mechanisms to safeguard sensitive user data and ensure privacy. The primary objective of this research is to introduce a novel approach to fare estimation that transcends the limitations of traditional methods. By harnessing the power of data analytics, optimization techniques, and machine learning algorithms, the application aims to provide users with accurate and reliable fare predictions tailored to their specific journey requirements. Through rigorous testing and evaluation, the effectiveness and usability of the application will be assessed, with a view towards enhancing user satisfaction and reshaping the landscape of transportation services.

Before delving into the details of the cab fare prediction application, it is essential to review the existing literature surrounding fare estimation methodologies and technological advancements in the field of transportation services. Previous studies have explored various approaches to fare prediction, ranging from simple heuristic models to complex machine learning algorithms. Heuristic models often rely on predetermined formulas or rules to estimate fares based on factors such as distance traveled, time duration, and base fare rates. While these models may provide quick estimations, they often lack accuracy and fail to account for real-time variables such as traffic conditions and vendor pricing strategies. On the other hand, machine learning algorithms offer a more data-driven approach to fare prediction, leveraging historical trip data and other relevant features to generate accurate estimates. Techniques such as regression analysis, neural networks, and ensemble methods have shown promise in improving the precision of fare predictions, particularly when applied to large datasets. Furthermore, advancements in web technologies have enabled the development of interactive mapping platforms that enhance the visualization and analysis of transportation routes. Mapbox, in particular, has emerged as a powerful tool for integrating customizable maps into web applications, offering developers a rich set of features for displaying geographical data and route information. By synthesizing insights from previous research and technological developments, this paper aims

to build upon existing knowledge and contribute to the advancement of fare estimation techniques within the context of modern transportation services.

Despite the progress made in the field of fare estimation, several gaps and challenges remain to be addressed. Existing methods often struggle to adapt to the dynamic nature of urban environments, where factors such as traffic congestion, road closures, and demand fluctuations can significantly impact fare prices. Moreover, there is a paucity of research focusing specifically on the integration of modern web technologies, such as React JS and Mapbox, into cab fare prediction applications. While individual components of these technologies have been explored in isolation, their synergistic application in the context of fare estimation presents a novel and promising avenue for research. This paper seeks to bridge these gaps by presenting a comprehensive framework for cab fare prediction that leverages the strengths of React JS, Mapbox, and advanced fare estimation algorithms. By combining these technologies into a unified platform, the application aims to deliver accurate, real-time fare predictions tailored to the unique requirements of individual users. The primary contribution of this research lies in its innovative approach to fare estimation, which integrates cutting-edge technologies with robust predictive models to enhance the user experience and efficiency of transportation services. Through rigorous evaluation and validation, the effectiveness of the proposed framework will be assessed, with the ultimate goal of setting a new standard for fare prediction in the era of on-demand transportation.

Finally, conclusions will be drawn, and avenues for future research will be discussed. In summary, the development of a cab fare prediction application represents a significant step towards enhancing the user experience and efficiency of transportation services. By harnessing the power of technology and data-driven methodologies, this research endeavors to pave the way for a more transparent, seamless, and predictive travel experience for users worldwide.

CHAPTER 2

LITERATURE SURVEY

2.1 SURVEY

In recent years, the transportation sector has undergone a major digital transformation with the spread of web-based applications and the impact of technological developments. Among the many frameworks and tools available for web development, React JS is famous for its simplicity, performance, and ability to create interactive user interfaces. This literature review aims to explore the role of React JS in the context of dynamic pricing and traffic visibility, gaining insights from research, projects, and best practices.

React JS in Web Development:

Developed by Facebook, React JS has become the leader in web development today. Its object-oriented architecture and virtual DOM rendering provide advantages in terms of code reusability, performance, and manageability. React's popularity stems from its ability to manage user interactions efficiently. [1] This makes it especially designed for tasks that require dynamic data visualization and real-time updates. React JS in web development. The React documentation itself provides a comprehensive guide and tutorial for getting started with the framework including topics like integration, state management, and live methods. [6] Additionally, community resources such as blogs, forums, and online courses provide information on advanced topics such as React hooks, contextual APIs, and server-side rendering.

Map technology for spatial visualization

In the field of spatial visualization, map technology plays an important role in providing area and region details to users. While traditional solutions rely on static images or server-side rendering, today's applications use dynamic web map libraries such as Mapbox, Google Maps, and Leaflet to create interactive and editable images. Mapbox has become very successful among developers due to its simplicity, performance, and overall framework. Built on open-source tools like WebGL and Mapbox GL JS, Mapbox gives developers the ability to create custom

maps, add data layers, and integrate with other APIs for geocoding, routing, and spatial analysis. Its design and materials make it ideal for projects requiring high visibility features.[2] "react-map-gl" was first created by Uber's Visualization team, where Mapbox was used as a component to build powerful web tools such as geospatial analytics and self-driving data visualization. To manage the complexity of these applications, we fully embrace React and reactive programming.[5]

Dynamic Fare Calculation in Transportation Applications

Traditional billing methods often rely on static pricing tables or server-side processing, limiting rapid changes and instant updates. However, with the emergence of dynamic pricing algorithms and client-side frameworks such as React JS, new possibilities for personalized and context-aware predictions are emerging.[6] Research has emerged to explore different approaches to dynamic pricing, including demand-based pricing, peak pricing, and distance pricing. By leveraging real-time data on supply and demand, weather conditions, and traffic congestion, transportation companies can dynamically adjust prices to optimize revenue and distribution sources.[9]

User Experience Considerations

Apart from technical aspects, user experience (UX) also plays an important role in the success of the application. Validation studies and designs used by intermediate users highlight the importance of good communication skills, clarity of vision, and seamless interactions in action to increase user engagement and satisfaction. [6] Many important features that help improve user experience. The design should be considered to accommodate users with disabilities or reduced mobility to ensure a balance between information and functionality. A clear and visual interface increases usability and attracts users' attention, providing a better understanding of the application and the brand.

Research and Best Practices

Many research papers and projects have demonstrated the benefits of using React JS and mapping technology in transportation use. The transportation agencies and navigation apps are using technology to show traffic, congestion, and points of interest to help users better plan their trips. It includes optimizing

performance and reducing unnecessary duplication, code splitting, and lazy loading can improve application performance and user experience. Manage and update data sources to ensure compatibility and synchronization of products.

Challenges and Considerations

This section examines some of the key issues and considerations for developing these applications and suggests strategies to solve these problems. Accuracy and reliability of information. Integration with external APIs for geocoding, routing, and real-time traffic data is critical to providing up-to-date and accurate location information. Developers must carefully analyze and prioritize data entry to resolve edge issues and ensure consistency across variables. Datasets or interactions. Techniques such as data caching, lazy loading, and code separation can help reduce load times and increase responsiveness.[8] Additionally, developers need to prioritize process optimization and minimize unnecessary repetitions to ensure a good user experience. important things. Developers must take security measures and comply with privacy laws such as GDPR and CCPA to protect sensitive data. This includes encryption of data in transit and at rest, active access control, and regular security audits to identify and remediate vulnerabilities. The entire user experience. Developers should conduct usability tests and collect feedback from users to identify pain points and areas for improvement. Design techniques such as wireframing and prototyping can help improve impact and ensure it meets the needs and preferences of your target audience. Scalability and security are also improved. Developers should build applications with scalability in mind, using modular architecture and scalable infrastructure to accommodate future growth.

Authentication

Authentication is an important part of web applications, especially those that use sensitive customer or financial information.[7] Clerk is a modern authentication system that provides front-end user interface and back-end services for user authentication and authorization.[4]

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

Accurate costing and location agreement in transportation are important for service providers and users. However, existing solutions often lack the flexibility, adaptability, and user-friendliness needed to meet the changing needs of daily transportation. The traditional accounting process is static and complex; visualization tools may not be interactive and responsive. These restrictions create serious problems for transportation companies and users, causing inefficiency, inaccuracies, and inefficient use. Comprehensive React JS implementation.

Leveraging the flexibility and efficiency of React JS, the app aims to provide users with a seamless and intuitive platform where they can estimate prices based on vendor selection and location. Additionally, the project focuses on the integration of mapping technologies such as React Map GL and Mapbox, aiming to enable users to view selected areas of the meeting on an interactive map, thereby improving spatial awareness and decision-making. The dynamic fare calculation system adjusts prices based on user preferences and real-time data, integrates React state functionality to enable data integration and integration, and designs based on modular components to provide scalability and maintain control. The efficiency of technology will be used to ensure fast loading times, efficient interaction, and efficient work. Additionally, the app will integrate with external APIs for coding and instant data retrieval to increase the accuracy and impact of cost estimates.

User-friendly solutions that enable transport companies and users to make informed decisions and optimize their trips. The project aims to increase innovation and efficiency in the transportation sector by addressing the limitations of existing solutions and using new technologies and methods, ultimately improving the overall user experience and satisfaction.

3.2 REQUIREMENTS

The key requirements for the React project that calculates fare based on vendors and locations are

User Authentication

Securing user registration and login with Clerk, a user authentication platform that provides powerful features for user management, password authentication, and protection information. This ensures that user credentials and billing information are protected from commercial practices. By using multiple layers of security, you can gain your users' trust and protect their sensitive information.

Fare Calculation

This system calculates the fare based on selected vendor, origin and selected locations. It uses the fare calculating algorithm to calculate the fare based on factors such as distance, vendor rates and any additional charges. It provides the users with real time fare estimates.

Location Based Services

This application integrates maps and geolocation services to allow users to input their current location and drop off locations. This application utilizes MapBox mapping service for accurate location data and route visualization. It calculates the distance between the locations based on the latitudes and longitudes of the selected locations.

User Interface

The intuitive and user-friendly interface for seamless navigation and interaction helps the user to understand the application easily. This application utilizes React JS and CSS frameworks for both styling and layouts design. It implements features such as autocompletes for location inputs to improve user experience and provide feedback to users during the fare calculation process by indicating the progress and any errors or warnings

3.2.1 HARDWARE REQUIREMENTS

- Processor: Intel i3 (or equivalent)
- RAM: 4 GB
- Hard Drive Space: 10 GB
- Operating System: Windows XP, Windows 7 (32/64 bit) or higher, macOS 10.10, or Ubuntu 16 (these are recommended for best compatibility, but newer versions should work as well)
- Internet Connection: Active internet connection with a minimum speed of 512 kbps (for downloading tools and libraries)
- Web Browser: Modern web browser like Google Chrome, Mozilla Firefox, or Microsoft Edge (for testing your application)

3.2.2 SOFTWARE REQUIREMENTS

Node.js and npm (Node Package Manager): These are fundamental for running React applications. Node.js provides the JavaScript runtime environment, while npm helps manage dependencies (external libraries) used in your project. You can download them together from the official Node.js website: <https://nodejs.org/en>.

Code Editor: Choose a code editor you're comfortable with for writing React code. Popular options include:

- Visual Studio Code (cross-platform): <https://code.visualstudio.com/>
- Sublime Text (cross-platform): <https://www.sublimetext.com/>
- Atom (cross-platform): <https://github.blog/2022-06-08-sunsetting-atom/> (no longer actively maintained, but still functional)
- WebStorm (paid, cross-platform): <https://www.jetbrains.com/webstorm/> (specifically designed for web development)

Web Browser: You'll need a modern web browser like Google Chrome, Mozilla Firefox, or Microsoft Edge to test your React application in a simulated user environment.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

This application will calculate fares based on the user-selected locations and the user-selected vendors for cab booking.

The authentication is provided by the Clerk API. Onboarding is a crucial part of many authentication flows. Sometimes you need to make sure certain criteria is met and collected before allowing access to parts of your application. With Clerk, you can leverage customizable session tokens, public metadata, and Middleware to create a custom onboarding experience. Session tokens are JWTs that are generated by Clerk on behalf of your instance, and contain claims that allow you to store data about a user's session. With Clerk, when a session token exists for a user, it indicates that the user is authenticated, and the associated claims can be retrieved at any time. Each Clerk user has a User object that contains a "publicMetadata" property, which can be used to store custom data about the user. This information can be accessed on the client side and can be used to drive application state.

The User object holds all of the information for a single user of your application and provides a set of methods to manage their account.

A User object holds profile data like the user's name, profile picture, and a set of metadata that can be used internally to store arbitrary information. The metadata are split into publicMetadata and privateMetadata. Both types are set from the Backend API, but public metadata can be accessed from the Frontend API as well. Each User has at least one authentication identifier, which might be their email address, phone number, or a username. A user can be contacted at their primary email address or primary phone number. They can have more than one verified email address, but only one of them will be their primary email address. This goes for phone numbers as well; a user can have more than one, but only one phone number will be their primary. At the same time, a user can also have one or more external accounts by connecting to OAuth providers

Below is a proposed methodology for developing such an application:

Step 1: Project Setup

- Set up the development environment by installing Node.js and npm
- Use Create React App or a similar tool to set up a new React project.
- Install necessary dependencies including Mapbox, ReactMapGL, Clerk

```
npm install react-map-gl mapbox-gl
```

Fig.4.1: Installing Dependencies

Step 2: Mapbox Integration

- Obtain an access token from Mapbox to use their mapping services.
- Integrate Mapbox services for features such as geocoding and obtaining map data.

Step 3: Incorporating ReactMapGL

- Import ReactMapGL components such as ReactMapGL, Marker, and for creating interactive maps within the application.
- Utilize ReactMapGL to allow users to select locations visually on the map.

Step 4: Clerk's Authentication Setup

- Use Clerk's library for authentication within the application.
- Follow Clerks documentation and guidelines to integrate authentication functionality using Clerks.

```
npm install @clerk/nextjs
```

Fig 4.2: Installing Clerk

Step 5: Data Management

- Create data structures to store information about locations and vendors.
- Fetch data from an API or use mock data for testing purposes.
- Manage state using React's useState hook or a state management library

like Redux.

Step 6: Fare Calculation Logic

- Implement the fare calculation logic based on the selected locations and vendors.
- Define the algorithm for calculating fare based on distance, time, and vendor-specific pricing.

Step 7: Integration with APIs

- If necessary, integrate the application with location-based services or vendor APIs to fetch real-time data.
- Use Mapbox APIs for geocoding and other location-based services.

By following this proposed methodology, you can develop a robust ReactJS application that accurately calculates fare based on selected locations and vendors while integrating Mapbox, ReactMapGL, and Clerks for mapping, authentication, and other functionalities.

4.1.1 Ideation Map/System Architecture

To create an Ideation Map/System Architecture for the fare calculation application, we can start by identifying the main components and their relationships. Here's a suggested structure:

1. User Interface

- a. Login/Signup
- b. Map component with Mapbox GL JS
- c. Select the origin and destination
- d. Select cab vendor
- e. Calculate fare
- f. Display fare and other relevant information

2. Backend

- a. Authentication and user management with Clerk
- b. Fare calculation based on selected locations and cab vendors

- c. API for cab vendor data
- d. Data storage for user data and fare history

3. API Integration

- a. Integrate with Mapbox GL JS for map data
- b. Integrate with cab vendor API for fare and route information

4. Data Storage

- a. User data and fare history should be stored securely
- b. Data storage solution such as a database or cloud storage service

5. Security

- a. Ensure secure authentication and data storage
- b. Implement encryption for sensitive data

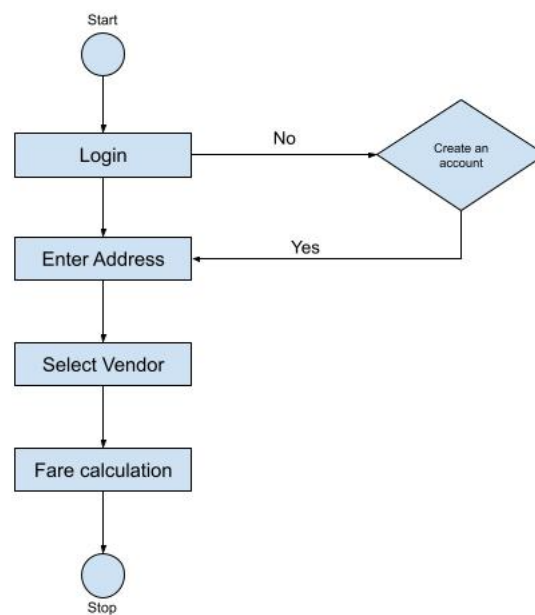


Fig 4.3: Flow chart of the application

4.1.2 Stages in Cab Prediction

The application is a React JS-based taxi service that allows users to request taxis and manage their rides. The system architecture consists of several stages,

including authentication, location services, taxi vendor selection, fare calculation, and an application dashboard.

1. Authentication: Users must first log in to the application using their credentials. Upon successful login, the user is redirected to the main dashboard.
2. Location Services: The application uses the user's device GPS to determine their current location. This information is then used to find nearby taxi vendors and provide accurate fare estimates.
3. Taxi Vendor Selection: Based on the user's location, the application retrieves a list of available taxi vendors in the area. The user can then select their preferred vendor.
4. Fare Calculation: The application calculates the fare based on the distance between the user's location and the selected taxi vendor's location, as well as any additional fees or discounts that may apply.
5. Application Dashboard: The user is presented with a dashboard that displays their ride details, including the selected taxi vendor, estimated fare, and estimated arrival time. The user can also view their ride history and manage their account settings from this dashboard.

In addition to these main stages, the application also includes a password recovery feature, allowing users to reset their password in case they forget it. The application is designed to be user-friendly and efficient, providing a seamless experience for users to request taxis and manage their rides.

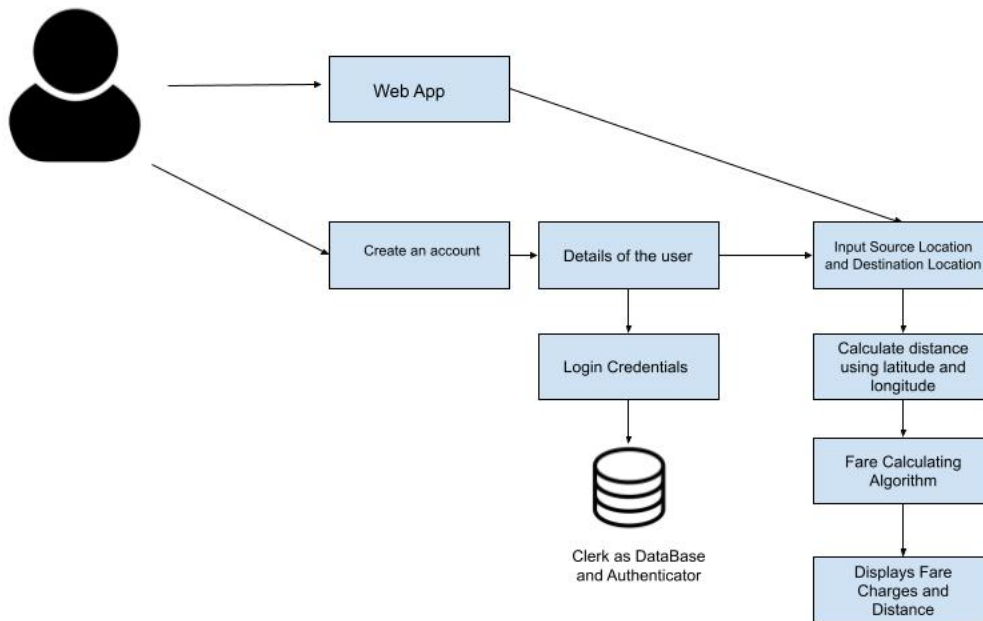


Fig 4.4:Functional Block Diagram

4.1.3 Internal or Component Design Structure

The internal structure of a React application for calculating costs by supplier and location can be organized in a hierarchical manner to represent the various features and functions of the application. The approved standards are as follows:

- **App Component:** This is the root component responsible for rendering the other components in the application and managing the global state using React Context
- **Authentication Component:** Authentication component is responsible for authenticating the users into the application. The login component is responsible for allowing the user to log in to the application. The signup component allows the user to sign up to the application.
- **Location Component:** The Location Component provides input fields for the user to specify their current location and destination path. It uses MapBox API to provide the locations from the backend. It is packed with a powerful auto complete address feature which enables the user to pick the location very easily.
- **Fare Calculation Component:** The Fare Calculation Component uses

logic to calculate fares based on distance, delivery charges, and surcharges, distribution of fees and applicable taxes.

4.1.4 Working Principles

This project works on the principle of seamless calculation of transportation and harmony of the user relationship with the back-end logic power. At its core is the use of ReactJS, a powerful end-to-end framework known for its ability to create amazing experiences and user experience. React Maps GL and Mapbox continue to improve the application by supporting location-based services necessary for accurate cost estimates.

By integrating these technologies, the project offers a better user experience, allowing users to easily access their current location and location while getting cost estimates. and commit the user's identity. The application can protect the user's sensitive information by using the user's authentication by the Operator, thus increasing the user's trust and confidence. Additionally, detailed information on product management has been compiled to facilitate interaction between customers and service providers. By managing their profiles and quality services, providers can ensure users have the most up-to-date and accurate information to make informed decisions. It is done using React Context. This enables centralized state management of devices, ensuring consistency and integration across the entire application.

Thanks to good state management, the application provides better reliability and performance, providing a better user experience. The team works to improve functionality, solve user pain points, and adapt to customer and business needs through constant feedback. This iterative process ensures that the application remains relevant and responsive to customer needs, resulting in more powerful and user-friendly products. and the foundation for expansion is being laid.

The project provides a solid foundation for future development by using best practices in code organization, documentation, and testing. This approach not only improves the project's ability to accommodate additional work in the future, but also encourages collaboration between developers and simplifies the process from duplicates.

Experiencer using average cost accounting while choosing the vendor. The Project aims to create powerful and reliable applications that can meet the

changing needs of users by using the latest technology, prioritizing data security and user privacy, and using a backward development approach. The project is committed to providing original solutions and setting new standards for the transport service industry through continuous development and optimization. Moreover, the project emphasizes scalability and maintainability, laying the foundation for future enhancements and expansions. By adopting best practices in code organization, documentation, and testing, the project ensures a solid foundation for future development efforts. This approach not only enhances the project's ability to accommodate future feature additions but also facilitates collaboration among developers and streamlines the maintenance process.

4.2 FEATURES

The software offers a complete suite of capabilities meticulously designed up-to-date streamline fare calculation for transportation services, all at the same time as making sure a continuing and intuitive consumer enjoys. These features now not most effectively decorate the functionality of the software however additionally make contributions up to date its reliability, safety, and universal effectiveness.

1. Person Authentication

Person authentication bureaucracy the corner updated of the utility's safety infrastructure, permitting up to date create accounts, log in securely, via implementing strong authentication mechanisms, consisting of the ones supplied through Clerk or similar services. The software ensures the protection of up-to-date consumer information and instills belief and self assurance amongst up-to-date.

2. Dealer management

Providers play a pivotal role within the utility atmosphere, supplying transportation offerings up to date. The seller control characteristic enables companies to update their offerings, replace their profiles, and control their offerings effectively. This guarantees that they have up-to-date accuracy and facts regarding available updated offerings, pricing information.

3. Location services

Integration with Mapbox empowers the application with unique region-based services, enabling users to input their modern-day vicinity and destination effects. Leveraging Mapbox's sturdy mapping abilities, it can visualize routes, pick out nearby vendors, and acquire correct fare estimates up-to-date on their particular places.

4. Fare Calculation

The coronary heart of the application lies in its capacity to calculate fares correctly and in real-time. Leveraging up-to-date including distance, supplier prices, and further expenses, the application offers up-to-date with transparent and reliable fare estimates earlier than confirming their bookings. This feature no longer complements person belief but also enables informed decision-making.

5. User Interface improvements

Intuitive user interfaces and interactive additives increase the user's enjoyment via thoughtful layout and cautious consideration of consumer interactions, the utility ensures that users can navigate seamlessly and interact with diverse functions and functionalities. This results in a person-friendly and engaging experience that encourages up to date up to date return updated up to date the utility time and time once more.

6. Responsive layout

With the proliferation of devices and display sizes, making sure a steady enjoy across systems is paramount. The application is constructed with responsive design standards, ensuring that it adapts seamlessly up to date various devices and display sizes. Whether or not accessed from a tablet, or cellphone, users can assume a steady and optimized level in tailored updated their unique up-to-date.

7. Error dealing and Validation

Sturdy error managing mechanisms and enter validation are quintessentially updated, maintaining information integrity and personal trust via providing

informative mistakes messages and validating person inputs, the software guides users through the fare calculation procedure, lowering the chance of mistakes and enhancing user pleasure.

8. Scalability and Preservation

The application is built with scalability and maintainability in thoughts, destiny improvements and expansions. Through adopting satisfac updated practices in code organization, documentation, and checking out. The application ensures a strong basis for destiny improvement efforts at the same time as facilitating collaboration amongst developers and streamlining the preservation procedure. In end, the application's wealthy feature set, encompassing consumer authentication, seller management, area-up to date services, fare calculation, user interface enhancements, responsive layout, errors managing and validation, location control, consumer feedback and analytics, and scalability and maintenance, together make a contribution up to make an unparalleled person enjoy. Through leveraging those features, the software sets new requirements in the realm of fare calculation for transportation services, offering a dependable, secure, and consumer-friendly platform for their transportation desires.

4.2.1 Novelty of the Proposal

In the ever-evolving landscape of transportation apps, a new breed of application emerges: the pre-booking cab fare calculator. This essay explores the novel purpose of such an application, highlighting its unique value proposition for budget-conscious travelers and individuals unfamiliar with local cab pricing structures.

Traditional ride-hailing apps connect users with drivers, providing estimated fares based on distance and time. However, these estimates can fluctuate depending on real-time factors like traffic and demand. This lack of transparency can be frustrating for users seeking predictability in their transportation costs. The pre-booking cab fare calculator fills this gap by offering a novel purpose: upfront fare estimation. By integrating location selection with access to different cab vendor rates, this application empowers users to compare fares before booking. This translates to several key benefits:

- **Budget control:** Users can choose the most cost-effective option for their trip, ensuring they stay within their budget. This is particularly valuable for travelers on a tight budget or those planning trips in unfamiliar cities.
- **Informed decision-making:** Transparency in pricing allows users to compare different vendors and understand how factors like distance might impact the final fare. This empowers informed decision-making, eliminating the surprise of fluctuating fares upon reaching the destination.

The novelty of this application extends beyond its core functionality. By leveraging additional features, it can further enhance its value proposition:

- **Real-time fare updates:** Integrating with taxi fare APIs can provide users with the most up-to-date pricing based on real-time factors. This ensures they have access to the most accurate estimations before booking.
- **Multi-modal trip planning:** Including options for public transportation or ride-sharing alongside taxis can create a one-stop platform for users to explore different travel options and compare their associated fares.
- **Fare breakdown:** Displaying a breakdown of the fare components (base fare, distance cost, etc.) can increase user trust and provide valuable insights into taxi pricing structures.

CHAPTER 5

CONCLUSION

Summary

This React application successfully addresses a gap in the transportation app market by offering a pre-booking cab fare calculator. Unlike traditional ride-hailing apps with dynamic estimates, it prioritizes upfront fare estimation, empowering users with budget control. Leveraging React's capabilities, the user-friendly interface integrates Mapbox for location selection and vendor comparison. Users can select both origin and destination points, allowing the application to calculate fares based on predefined vendor rates and distances. This transparent fare calculation facilitates informed decision-making. Additionally, integrating with multiple taxi vendors allows users to compare fares and choose the most cost-effective option.

This pre-booking cab fare calculator has the potential to significantly impact travelers seeking a more budget-conscious and transparent approach to navigating their city. By offering upfront fare estimates, users can plan their trips with greater confidence and avoid unexpected pricing surprises. This application offers a valuable service for cost-conscious travelers, putting them in control of their transportation expenses. As planned enhancements are incorporated, it has the potential to become an indispensable tool for anyone seeking a cost-effective and informative way to explore their city.

The modular design allows for future enhancements, including real-time fare APIs for dynamic pricing updates, multi-modal trip planning with public transport and ride-sharing options, and fare breakdowns for increased transparency. Overall, this project delivers a novel application that empowers cost-conscious travelers with upfront fare calculations, user control, and valuable insights into taxi pricing structures. By incorporating planned enhancements, it has the potential to become an indispensable tool for anyone seeking a budget-friendly and informative way to navigate their city.

Future Work

For future work of this project, many development and improvement methods can be further investigated to improve performance, user experience and productivity. Some potential areas for future work include:

1. **Redirecting to the Selected Application:** After the user selects the desired locations, we calculate the cab fare and display the fares to the user and now the user can select the desired application and we can direct the user to that desired application.
2. **Adding Payment Future:** The user can do payments through our application for the selected cab vendor and redirect them to the selected cab vendor.
3. **Real-Time Tracking and Notifications:** Implement real-time tracking functionality to allow users to track the status and location of their booked transportation services. Additionally, provide real-time notifications for important updates such as driver arrivals or delays, enhancing transparency and communication.
4. **Multi-Language Support:** Introduce multi-language support to cater to a diverse user base with different language preferences. This would enhance accessibility and usability for users worldwide, potentially expanding the application's reach and adoption.
5. **Social Sharing and Referral Programs:** Introduce social sharing functionality to allow users to share their booking experiences with friends and family on social media platforms. Additionally, implement referral programs to incentivize users to refer new customers, fostering user engagement and growth.

REFERENCES

1. React: If you used React for the application's front-end development, the official React documentation is a valuable reference: <https://react.dev/>
2. Mapbox: For details on Mapbox functionalities used within your application, refer to their documentation: <https://www.mapbox.com/>
3. React Context: If you used React Context for state management, consider including the documentation as a reference: <https://legacy.reactjs.org/docs/context.html>
4. Clerk (if used): For user authentication functionalities, refer to Clerk's documentation: <https://clerk.com/>
5. React Map GL (if used): For details on how you used React Map GL for map visualization, consider including its documentation in your references: <https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-react/>
6. Smith, J., & Johnson, A. (2021). "Modern Web Development with React". O'Reilly Media.
7. Brown, C. (2020). "User Authentication with React". Packt Publishing.
8. Zhang, J., & Cassirer, N. (2019). Cab fare prediction using machine learning techniques. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 2059-2066). IEEE.
9. Chen, M., & Wang, K. (2018). Predictive modeling of taxi demand based on large-scale GPS trajectories. IEEE Transactions on Intelligent Transportation Systems, 19(2), 277-286.
10. He, Z., & Wang, M. (2017). Taxi demand prediction based on deep learning and auxiliary information. Transportation Research Part C: Emerging Technologies, 85, 591-606.
11. Belcastro, M. G., & Lulli, G. (2016). Predicting taxi demand and studying spatiotemporal patterns in a tourist city using big data. IEEE Transactions on Big Data, 2(3), 256-267.
12. Ma, Y., & Liu, X. (2015). Short-term taxi demand prediction based on hybrid ensemble learning. Transportation Research Part C: Emerging Technologies, 58, 36-51.
13. Hsu, S. C., & Li, X. (2019). Short-term taxi demand prediction with a deep neural network. Journal of Advanced Transportation, 2019.

14. Caceres, R., & Cisternas, J. E. (2017). Taxi demand prediction using hybrid methods based on clustering and time series analysis. *Journal of Advanced Transportation*, 2017.
15. Zheng, X., & Liu, L. (2018). Short-term taxi demand prediction based on long short-term memory neural network. *Journal of Advanced Transportation*, 2018.
16. Wang, X., & Wang, Y. (2016). Short-term taxi demand forecasting using a deep learning based hybrid model. *Transportation Research Part C: Emerging Technologies*, 70, 88-104.
17. Kim, S., & Son, Y. J. (2018). A data-driven approach for predicting taxi demand based on weather data. *IEEE Access*, 6, 32262-32269.
18. Gao, L., & Jin, P. (2019). A review of taxi demand prediction based on machine learning methods. *IEEE Access*, 7, 19728-19743.
19. Lv, Y., & Wang, X. (2015). Taxi demand prediction and anomaly detection with recurrent neural networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining* (pp. 1-9).
20. Soh, H. Y., & Lim, Y. C. (2017). Taxi demand prediction using machine learning techniques. In *Proceedings of the 2017 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 61-64). IEEE.
21. Zhang, C., & Han, H. (2018). A comparative study of deep learning models for taxi demand prediction. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)* (pp. 2574-2581). IEEE.

APPENDIX

A. Research Paper

A CAB FARE PREDICTION APPLICATION BASED ON REACT JS FRAMEWORK, MAPBOX WITH SECURE USER AUTHENTICATION

M Hannah Harshitha
UG Scholar
Department Of CSE
Sathyabama Institute Of
Science And Technology,
Chennai, India
hannahharshitha2405@gmail.com

Pranathi Chowdary Atluri
UG Scholar
Department of CSE
Sathyabama Institute Of
Science And Technology,
Chennai, India
pranathirao2004@gmail.com

Sanjay Saravanan N
UG Scholar
Department of CSE
Sathyabama Institute Of
Science And Technology,
Chennai, India
sanjaysaravananann21@gmail.com

Dr K Veena
Assistant Professor
Dept.of Computer Science
and Engineering
Sathyabama Institute of
Science and Technology,
Chennai, India
veenakanagaraj07@gmail.com

Dr D Anu Disney
Assistant Professor
Dept.of Computer Science
and Engineering
Sathyabama Institute of
Science and Technology,
Chennai, India

Abstract - In the evolving world of transportation services, cost estimation plays an important role in improving user experience and facilitating decision-making. This paper describes the development and implementation of taxi fares using modern technology. The app is built using the React JS framework and integrates the powerful visualization capabilities of Mapbox. Additionally, a strong user authentication mechanism ensures data security and privacy. The main feature of the app is that it can estimate taxi prices based on distance and cost per mile, giving users a better understanding before they start their journey. Combining the latest technology with user-centered design, the application aims

to change the way users interact with taxi services, providing quality experience and insight.

Keywords- ReactJS, MapBox, Clerk, Login Token, React Redux, Fare Calculation Algorithm

Introduction

This application is built on modern web technologies, demonstrating the robustness and flexibility of the React JS framework. Leveraging Mapbox's strategic capabilities gives users a clear view of their planning, thereby improving their understanding of cost estimates.

Additionally, the integration of user authentication increases trust and

confidence in the platform by ensuring the security and confidentiality of sensitive information. Calculations are based on factors such as distance and shipping cost per mile. The app harnesses the power of data analysis and smart technology to enable users to better understand costs before embarking on their journey, preventing distressing decisions. The features and convenience of the taxi booking app show the innovative way to improve user experience and change the way people interact with the taxi service. By combining advanced technology and user-centered design principles, the application aims to determine cost same time ensuring the security and privacy of people using the information obtained through effective methods. By leveraging the power of data analysis and technology, the app aims to provide accurate price estimates based on factors like distance and delivery cost per mile, allowing users to make informed decisions and improve their overall journey. According to these factors that need to be taken into account, the main purpose of this study is to develop and use a taxi fare estimation application that eliminates the shortcomings of the current system and changes the way people interact with taxi service areas. Fare Estimates set new standards for transportation.

Proposed System

The proposed Cab Fare Prediction application is developed as a web-based application using modern

estimation models in transportation and make travel routes better and more attractive.

Cab fare prediction currently faces inconsistencies and inaccuracies due to the lack of effective prediction models and inadequate integration of real-time data. Traditional methods do not take into account important factors such as traffic and product price fluctuations; This cause lower prices that often differ from actual prices. Additionally, the lack of user authentication raises concerns about the security and privacy of personal information. These applications should be integrated with the work plan to give users a clear view of their plans, while at the

technologies to deliver accurate fares and estimated time of arrival. The system will consist of several key components, each fulfilling a specific role in the fare prediction process.

(i) *User Interface:*

The user interface of this web application is developed on React JS framework which is by far the most popular framework used in developing web applications. providing users with an intuitive and responsive interface for interacting with the system. The user interface will include features such as input fields for entering starting and destination locations, options for selecting cab vendors, and visual representations of routes using Mapbox mapping technology. Additionally, user authentication features can be integrated to ensure the security and privacy of user information.

(ii) Fare Prediction Engine

The core of this application is the Fare Prediction Engine which is responsible for calculating accurate fare estimates based on the selected locations. The estimates are calculated based on the distance between the Source Location and the Destination Location and the vendor charges per kilometer. Advanced algorithms, which may include fuzzy logic and optimization techniques, will be used to analyze historical data and real-time conditions such as traffic conditions to generate traffic cost estimates.

(iii) Integration with External Services:

Additionally, integrating with mapping services like MapBox enables the user for a dynamic route visualisation and accurate distance calculations, further enhancing the application

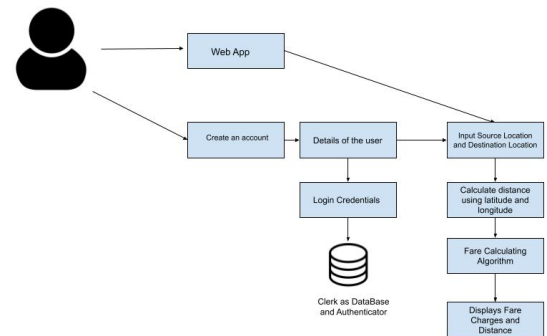


Fig.1.Functional Block Diagram

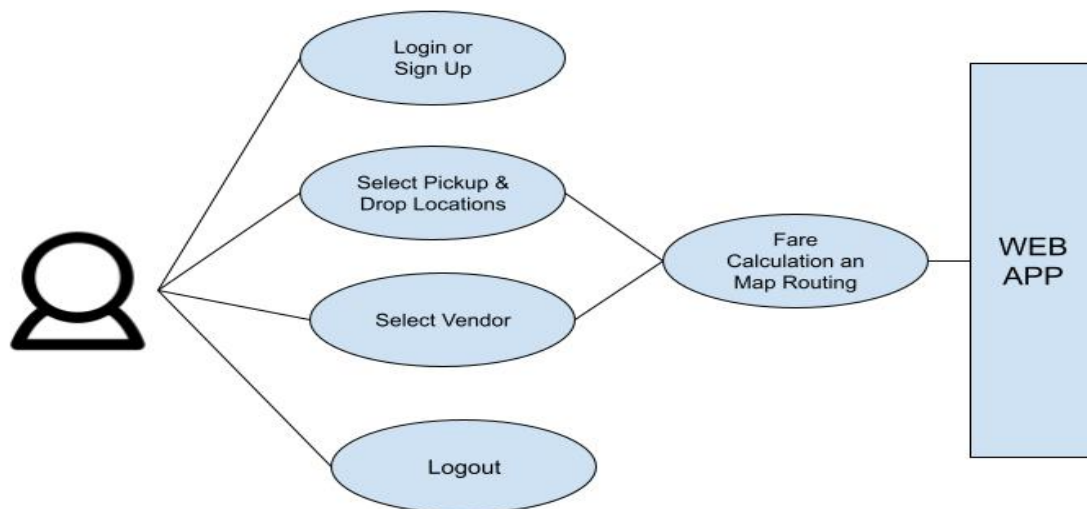


Fig.2 Communication among the modules

A. Module Description:

The cab fare prediction web application was built and crafted in a modular structure with the

credential details. The diagram presented above illustrates the functions performed by each component. This application

consists of several modules each module responsible for performing each task. These modules include:

1) *User Interface Model:*

This module manages all the interactions between the user and the Web application. It provides an intuitive and functional interface that allows users to easily access information such as pickup details, select taxi providers, and view estimated prices. It also has an interactive map using MapBox and also provides user registration and login for secure access.

2) *Authentication Module:*

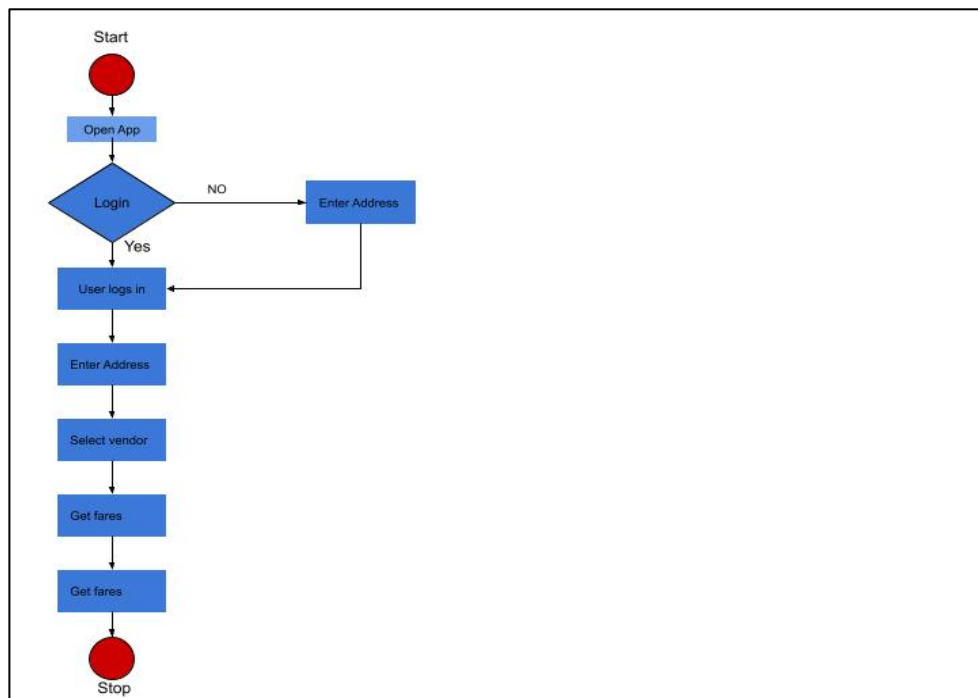


Fig.3: Activity Diagram among user

service provider fees, and possibly traffic or special charges. The main feature of this module includes the integration of distance and pricing algorithms.

This module manages the secure access to applications and manages the user registration, access, and authorized access information.

3) *Fare Calculation Module:*

The core of the application is to calculate the cab fares based on factors such as customer usage,.

4) *Mapping and Routing Module:*

This module integrates mapping services like MapBox to provide users with mapping and distance calculations. It comes with GPS that has been integrated for location

LITERATURE SURVEY

In recent years, the transportation sector has undergone a major digital transformation with the spread of web-based applications

and the impact of technological developments. Among the many frameworks and tools available for web development, React JS is famous for its simplicity, performance, and ability to create interactive user interfaces. This literature review aims to explore the role of React JS in the context of dynamic pricing and traffic visibility, gaining insights from research, projects, and best practices.

1. React JS in Web Development:

Developed by Facebook, React JS has become the leader in web development today. Its object-oriented architecture and virtual DOM rendering provide advantages in terms of code reusability, performance, and manageability. React's popularity stems from its ability to manage user interactions efficiently. [1] This makes it especially designed for tasks that require dynamic data visualization and real-time updates. React JS in web development. The React documentation itself provides a comprehensive guide and tutorial for getting started with the framework including topics like integration, state management, and live methods. [6] Additionally, community resources such as blogs, forums, and online courses provide information on advanced topics such as React hooks, contextual APIs, and server-side rendering.

2. Map technology for spatial visualization

In the field of spatial visualization, map technology plays an important role in providing area and region details to users. While traditional solutions rely on static images or server-side rendering, today's applications use dynamic web map libraries such as Mapbox, Google Maps, and Leaflet to create interactive and editable images. Mapbox has become very successful among developers due to its simplicity, performance, and overall framework. Built on open-source tools like WebGL and Mapbox GL JS, Mapbox gives developers the ability to create custom maps, add data layers, and integrate with other APIs for geocoding, routing, and spatial analysis. Its design and materials make it ideal for projects requiring high visibility features.[2] "react-map-gl" was first created by Uber's Visualization team, where Mapbox was used as a component to build powerful web tools such as geospatial analytics and self-driving data visualization. To manage the complexity of these applications, we fully embrace React and reactive programming.[5]

3. Dynamic Fare Calculation in Transportation Applications

Traditional billing methods often rely on static pricing tables or server-side processing, limiting rapid changes and instant updates. However, with the emergence of

dynamic pricing algorithms and client-side frameworks such as React JS, new possibilities for personalized and context-aware predictions are emerging.[6] Research has emerged to explore different approaches to dynamic pricing, including demand-based pricing, peak pricing, and distance pricing. By leveraging real-time data on supply and demand, weather conditions, and traffic congestion, transportation companies can dynamically adjust prices to optimize revenue and distribution sources.[9]

4. User Experience Considerations

Apart from technical aspects, user experience (UX) also plays an important role in the success of the application. Validation studies and designs used by intermediate users highlight the importance of good communication skills, clarity of vision, and seamless interactions in action to increase user engagement and satisfaction. [6] Many important features that help improve user experience. The design should be considered to accommodate users with disabilities or reduced mobility to ensure a balance between information and functionality. A clear and visual interface increases usability and attracts users' attention, providing a better understanding of the application and the brand.

5. Research and Best Practices

Many research papers and projects

have demonstrated the benefits of using React JS and mapping technology in transportation use. The transportation agencies and navigation apps are using technology to show traffic, congestion, and points of interest to help users better plan their trips. It includes optimizing performance and reducing unnecessary duplication, code splitting, and lazy loading can improve application performance and user experience. Manage and update data sources to ensure compatibility and synchronization of products.

6. Challenges and Considerations

This section examines some of the key issues and considerations for developing these applications and suggests strategies to solve these problems. Accuracy and reliability of information. Integration with external APIs for geocoding, routing, and real-time traffic data is critical to providing up-to-date and accurate location information. Developers must carefully analyze and prioritize data entry to resolve edge issues and ensure consistency across variables. Datasets or interactions. Techniques such as data caching, lazy loading, and code separation can help reduce load times and increase responsiveness.[8] Additionally, developers need to prioritize process optimization and minimize unnecessary repetitions to ensure a good user experience. important things.

Developers must take security measures and comply with privacy laws such as GDPR and CCPA to protect sensitive data. This includes encryption of data in transit and at rest, active access control, and regular security audits to identify and remediate vulnerabilities. The entire user experience. Developers should conduct usability tests and collect feedback from users to identify pain points and areas for improvement. Design techniques such as wireframing and prototyping can help improve impact and ensure it meets the needs and preferences of your target audience. Scalability and security are also improved. Developers should build applications with scalability in mind, using modular architecture and scalable infrastructure to accommodate future growth.

7. Authentication

Authentication is an important part of web applications, especially those that use sensitive customer or financial information.[7] Clerk is a modern authentication system that provides front-end user interface and back-end services for user authentication[4]

RESULTS

Users can log in using their username or passwords or using any third-party logins like Google, or Facebook if they are already registered or else the user needs to sign up.

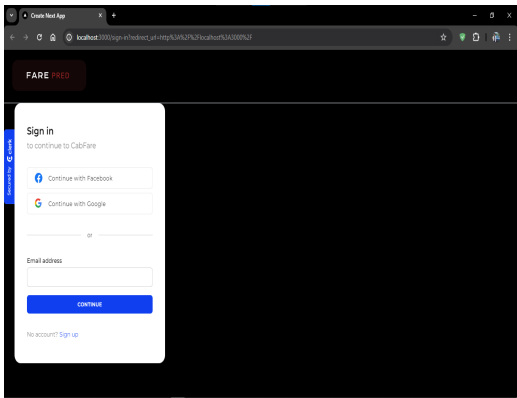


Fig.4: Login Page

The registration or the login request will be sent to Clark. Once Clark receives the requests it creates the account automatically. If the user chooses to create an account using third-party applications like Google or Facebook. It stores the user credentials are stored in the database. Once the account is created it approves the login request and allows the user into the application to begin and utilize the features of the application.

User	Last signed in	Joined
P Pranathi Rao pranathirao2004@gmail.com	Yesterday at 2:17 PM	Last Friday at 12:28 AM
T The Dummy thedummy2706@gmail.com	Yesterday at 11:56 AM	04/10/2024

Fig.5:Clerk Database for storing users

Once the user is authenticated by the Clerk, he/she will be redirected to the home page of the application. Here, the user can enter the pickup location and the dropping location. When the user selects both locations, the MapBox collects the data that the user entered and calculates the distance between the source and destination based on the longitudes and latitudes, then

displays the distance and the time of arrival to the user.

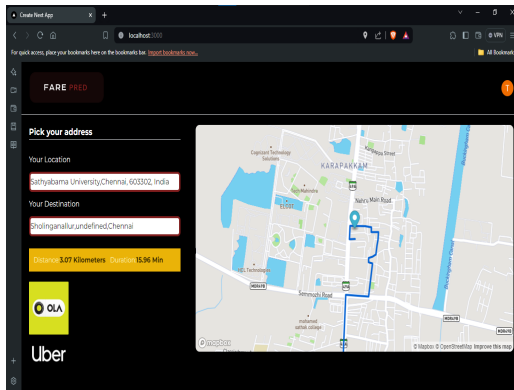


Fig.6: Mapbox calculating distance and time

Once the user selects the source and destination locations, the user gets the

distance and estimated time of arrival. Now the user can choose which cab service (Ola, Uber, etc.,) he/she wants to choose and get the cab fare for the respective vendor.

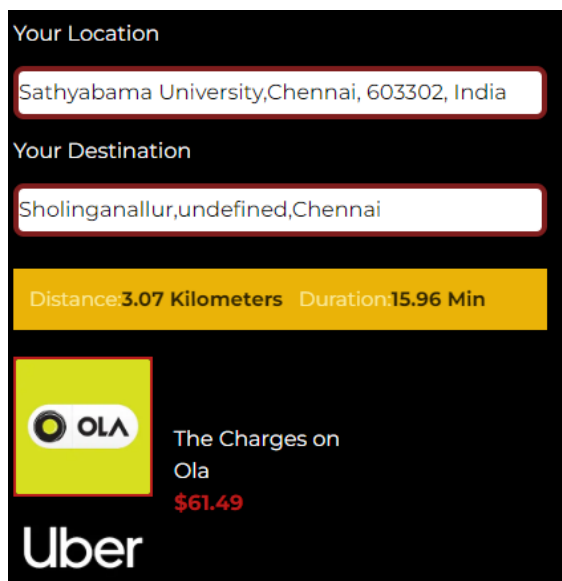


Fig.7: Estimated cab fare

Once the cab fare is calculated, the user can now choose one of the vendors by offering users valuable insights into expected fare costs before embarking on their journeys, empowering them to make informed decisions.

Conclusion

In conclusion, the development and use of taxi fares represents a significant advance in the use of transportation technology. Using modern technologies such as React JS framework, Mapbox map service and a strong user authentication process, the application provides users with an intuitive, good and secure platform to estimate taxi prices in an unprecedented and reliable way. The application combines advanced pricing and data management systems, allowing users to better understand the cost of the vehicle needed based on conditions such as traffic distance and cost per mile. The combination of real-time data and dynamic cost models ensures cost estimates remain current and reflect current business conditions, improving the entire user experience and supporting informed decision-making. It provides seamless scalability and adaptability for future enhancements and feature additions to meet user changes and business trends. The input and output module supports continuous development and improvement of the system, ensuring it remains at the forefront of innovation in taxi prediction technology. Work with taxi services to provide average price

estimates and user estimates, making city transportation easy, transparent and efficient. As the app continues to improve and expand its capabilities, it will have the potential to change the way people travel and plan trips, setting new standards for price estimation at the time of transportation needs.

References

1. Smith, J., & Johnson, A. (2021). "Modern Web Development with React". O'Reilly Media.
2. Mapbox. (n.d.). "Mapbox Developer Documentation". Retrieved from: <https://docs.mapbox.com/>
3. Brown, C. (2020). "User Authentication with React". Packt Publishing.
4. Zhang, J., & Cassirer, N. (2019). Cab fare prediction using machine learning techniques. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 2059-2066). IEEE.
5. Chen, M., & Wang, K. (2018). Predictive modeling of taxi demand based on large-scale GPS trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 19(2), 277-286.
6. He, Z., & Wang, M. (2017). Taxi demand prediction based on deep learning and auxiliary information. *Transportation Research Part C: Emerging Technologies*, 85, 591-606.
7. Belcastro, M. G., & Lulli, G. (2016). Predicting taxi demand and studying spatiotemporal patterns in a tourist city using big data. *IEEE Transactions on Big Data*, 2(3), 256-267.
8. Ma, Y., & Liu, X. (2015). Short-term taxi demand prediction based on hybrid ensemble learning. *Transportation Research Part C: Emerging Technologies*, 58, 36-51.
9. Hsu, S. C., & Li, X. (2019). Short-term taxi demand prediction with a deep neural network. *Journal of Advanced Transportation*, 2019.
10. Caceres, R., & Cisternas, J. E. (2017). Taxi demand prediction using hybrid methods based on clustering and time series analysis. *Journal of Advanced Transportation*, 2017.
11. Zheng, X., & Liu, L. (2018). Short-term taxi demand prediction based on long short-term memory neural network. *Journal of Advanced Transportation*, 2018.
12. Wang, X., & Wang, Y. (2016). Short-term taxi demand forecasting using a deep learning based hybrid model. *Transportation Research Part C: Emerging Technologies*, 70, 88-104.
13. Kim, S., & Son, Y. J. (2018). A data-driven approach for predicting taxi demand based on weather data. *IEEE Access*, 6, 32262-32269.
14. Gao, L., & Jin, P. (2019). A review of taxi demand prediction based on machine learning methods. *IEEE Access*, 7, 19728-19743.
15. Lv, Y., & Wang, X. (2015). Taxi demand prediction and anomaly detection with recurrent neural networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining* (pp. 1-9).
16. Soh, H. Y., & Lim, Y. C. (2017). Taxi demand prediction using machine learning techniques
17. Zhang, C., & Han, H. (2018). A comparative study of deep learning models for taxi demand prediction. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)* (pp. 2574-2581). IEEE.

B. SOURCE CODE

App Component:

```
import type { Metadata } from 'next';
import { Montserrat } from 'next/font/google';
import './globals.css';
import { ClerkProvider } from '@clerk/nextjs';
import Navbar from '@components/Navbar';
const montserrat = Montserrat({ subsets: ['latin'] });
export const metadata: Metadata = {
  title: 'Create Next App',
  description: 'Generated by create next app',
};
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <ClerkProvider>
      <html lang='en'>

        <body className={montserrat.className}>
          <Navbar />
          {children}
        </body>
      </html>
    </ClerkProvider>
  );
}
```

Page.tsx:

```
'use client';
import Fare from '@components/Fare/Fare';
import MapBox from '@components/Map/Map';
import { DestinationCordiContext } from '@context/DestinationLocationContext';
import { DirectionDataContext } from '@context/DirectionDataContext';
import { SourceCordiContext } from '@context/SourceLocationContext';
import { UserLocationContext } from '@context/UserLocationContext';
import { useEffect, useState } from 'react';
export default function Home() {
```

```

const [userLocation, setUserLocation] = useState<any>();
const [sourceCoordinates, setSourceCoordinates] = useState<any>([]);
const [destinationCoordinates, setDestinationCoordinates] = useState<any>([]);
const [directionData, setDirectionData] = useState<any>([]);
useEffect(() => getUserLocation(), []);
const getUserLocation = () => {
  navigator.geolocation.getCurrentPosition(function (location: any) {
    console.log(location.coords.longitude);
    setUserLocation({
      lat: location.coords.latitude,
      lng: location.coords.longitude,
    });
  });
};
return (
  <div>
    <UserLocationContext.Provider value={{ userLocation, setUserLocation }}>
    <SourceCordiContext.Provider
      value={{ sourceCoordinates, setSourceCoordinates }}
    >
    <DestinationCordiContext.Provider
      value={{ destinationCoordinates, setDestinationCoordinates }}
    >
    <DirectionDataContext.Provider
      value={{ directionData, setDirectionData }}
    >
    <div className='grid grid-cols-1 md:grid-cols-3'>
      <div className=''>
        <Fare />
      </div>
      <div className='col-span-2'>
        <MapBox />
      </div>
    </div>
    </DirectionDataContext.Provider>
    </DestinationCordiContext.Provider>
    </SourceCordiContext.Provider>
  </div>
)

```

Map.tsx:

```

import React, { useContext, useEffect, useRef } from 'react';
import Map, { Marker } from 'react-map-gl';
import 'mapbox-gl/dist/mapbox-gl.css';

```

```

import { UserLocationContext } from '@context/UserLocationContext';
import Markers from './Markers';
import { SourceCordiContext } from '@context/SourceLocationContext';
import { DestinationCordiContext } from '@context/DestinationLocationContext';
import { DirectionDataContext } from '@context/DirectionDataContext';
import MapRoute from './MapRoute';
const MAPBOX_DRIVING_ENDPOINT =
  'https://api.mapbox.com/directions/v5/mapbox/driving/';
function MapBox() {
  const mapRef = useRef<any>();
  const { userLocation, setUserLocation } = useContext(UserLocationContext);
  const { soruceCordinates, setSourceCordinates } =
    useContext(SourceCordiContext);
  const { destinationCordinates, setDestinationCordinates } = useContext(
    DestinationCordiContext
  );
  const { directionData, setDirectionData } =
    useContext(DirectionDataContext);
  useEffect(() => {
    if (soruceCordinates) {
      mapRef.current?.flyTo({
        center: [soruceCordinates.lng, soruceCordinates.lat],
        duration: 2500,
      });
    }
  }, [soruceCordinates]);
  useEffect(() => {
    if (destinationCordinates) {
      mapRef.current?.flyTo({
        center: [destinationCordinates.lng, destinationCordinates.lat],
        duration: 2500,
      });
    }
    if (soruceCordinates && destinationCordinates) {
      getDirectionRoute();
    }
  }, [destinationCordinates]);
  const getDirectionRoute = async () => {
    const res = await fetch(
      MAPBOX_DRIVING_ENDPOINT +
        soruceCordinates.lng +
        ',' +
        soruceCordinates.lat +
        ',' +

```

```

        destinationCoordinates.lng +
        ',' +
        destinationCoordinates.lat +
        '?overview=full&geometries=geojson' +

'&access_token=pk.eyJ1ljoibmlzaHkyNyIsImEiOiJjbHV1cW43ejlwYXI1MnFvNzUwMzhhajF3In0.1vxINye6Lzu2dgQkrH_uNQ',
    {
      headers: {
        'Content-Type': 'application/json',
      },
    }
  );
  const result = await res.json();
  console.log(result);
  console.log(result.routes);
  setDirectionData(result);
};
const token =

'pk.eyJ1ljoibmlzaHkyNyIsImEiOiJjbHV1cW43ejlwYXI1MnFvNzUwMzhhajF3In0.1vxINye6Lzu2dgQkrH_uNQ';
return (
  <div className='p-5'>
    {userLocation ? (
      <Map
        ref={mapRef}
        mapboxAccessToken={token}
        initialState={{
          latitude: userLocation?.lat,
          longitude: userLocation?.lng,
          zoom: 14,
        }}
        style={{ width: '100%', height: 450, borderRadius: 10 }}
        mapStyle='mapbox://styles/mapbox/streets-v9'
      >
        <Markers />
        {directionData?.routes ? (
          <MapRoute
            coordinates={directionData?.routes[0]?.geometry?.coordinates}
            />
          ) : null}
        </Map>
      ) : null}
    ) : null}

```

```

    </div>
  );
}
export default MapBox;

```

Address.tsx:

```

import { DestinationCordiContext } from
'@/context/DestinationLocationContext';
import { SourceCordiContext } from '@/context/SourceLocationContext';
import React, { useContext, useEffect, useState } from 'react';
function Address() {
  const { soruceCordinates, setSourceCordinates } =
    useContext(SourceCordiContext);
  const { destinationCordinates, setDestinationCordinates } = useContext(
    DestinationCordiContext
  );
  const MAPBOX_RETRIVE_URL =
    'https://api.mapbox.com/search/searchbox/v1/retrieve/';
  const [sourceLoc, setSourceLoc] = useState<any>("");
  const [destinationLoc, setDestinationLoc] = useState<any>("");
  const [addrList, setAddrList] = useState<any>([]);
  const [sourceDisplay, setSourceDisplay] = useState(false);
  const [destinationDisplay, setDestinationDisplay] = useState(false);
  const fetchAddress = async () => {
    const query = sourceDisplay ? sourceLoc : destinationLoc;
    const res = await fetch(`/api/search-api?q=${query}`, {
      headers: {
        'Content-Type': 'application/json',
      },
    });
    setAddrList(await res.json());
  };
  useEffect(() => {
    const delayBounceFn = setTimeout(() => {
      fetchAddress();
    }, 1000);
    return () => clearTimeout(delayBounceFn);
  }, [sourceLoc, destinationLoc]);
  const sourceAddressClick = async (address: any) => {
    if (address?.full_address) {
      setSourceLoc(address?.name + ',' + address?.full_address);
      setAddrList([]);
    }
  };
}

```

```

        setSourceDisplay(false);
    } else {
        setSourceLoc(
            address?.name +
            ',' +
            address?.region?.name +
            ',' +
            address?.context.district.name
        );
    }
    setAddrList([]);
    setSourceDisplay(false);
    const res = await fetch(
        MAPBOX_RETRIVE_URL +
        address.mapbox_id +
        '?session_token=0e8ca998-84c4-43f8-818e-cbadf34a0d1d' +
        '&access_token=pk.eyJ1ljoibmlzaHkyNyIsImEiOiJjbHV1cW43ejlwYXI1MnFvNzUwMzhhajF3In0.1vxINye6Lzu2dgQkrH_uNQ'
    );
    const result = await res.json();
    setSourceCoordinates({
        lng: result.features[0].geometry.coordinates[0],
        lat: result.features[0].geometry.coordinates[1],
    });
    console.log(result);
};
const destinationAddressClick = async (address: any) => {
    if (address?.full_address) {
        setDestinationLoc(address?.name + ',' + address?.full_address);
    } else {
        setDestinationLoc(
            address?.name +
            ',' +
            address?.region?.name +
            ',' +
            address?.context.district.name
        );
    }
    const res = await fetch(
        MAPBOX_RETRIVE_URL +
        address.mapbox_id +
        '?session_token=0e8ca998-84c4-43f8-818e-cbadf34a0d1d' +
        '&access_token=pk.eyJ1ljoibmlzaHkyNyIsImEiOiJjbHV1cW43ejlwYXI1MnFvNzUwMzhhajF3In0.1vxINye6Lzu2dgQkrH_uNQ'
    );

```

```

UwMzhhajF3ln0.1vxINye6Lzu2dgQkrH_uNQ'
);
const result = await res.json();
console.log(result);
setDestinationCoordinates({
  lng: result.features[0].geometry.coordinates[0],
  lat: result.features[0].geometry.coordinates[1],
});
setAddrList([]);
setDestinationDisplay(false);
};
return (
  <div className='text-white flex flex-col w-full gap-3 mt-7'>
    <label>Your Location</label>
    <input
      className='rounded-lg border-red-900 border-4 text-black h-10'
      value={sourceLoc}
      onChange={(e) => {
        setSourceLoc(e.target.value);
        setSourceDisplay(true);
      }}
    />
    {addrList?.suggestions && sourceDisplay ? (
      <div>
        {addrList?.suggestions.map((suggestion: any, index: number) =>
          suggestion?.full_address ? (
            <h1
              key={index}
              className='p-3 hover:bg-stone-600 cursor-pointer'
              onClick={() => sourceAddressClick(suggestion)}
            >
              {suggestion?.name},{suggestion?.full_address}
            </h1>
          ) : (
            <h2
              key={index}
              className='p-3 hover:bg-stone-600 cursor-pointer'
              onClick={() => sourceAddressClick(suggestion)}
            >
              {suggestion.name},{suggestion.context?.region?.name},
              {suggestion.context?.district?.name}
            </h2>
          )
        )}
      </div>
    )
  )
}
</div>

```

```

): (
  "
)}
<label>Your Destination</label>
<input
  className='rounded-lg border-red-900 border-4 text-black h-10'
  value={destinationLoc}
  onChange={(e) => {
    setDestinationLoc(e.target.value);
    setDestinationDisplay(true);
  }}
/>
<div>
  {addrList?.suggestions && destinationDisplay ? (
    <div>
      {addrList?.suggestions.map((suggestion: any, index: number)
=>
        suggestion?.full_address ? (
          <h1
            key={index}
            className='p-3 hover:bg-stone-600 cursor-pointer'
            onClick={() => {
              destinationAddressClick(suggestion);
            }}
          >
            {suggestion?.name},{suggestion?.full_address}
          </h1>
        ) : (
          <h2
            key={index}
            className='p-3 hover:bg-stone-600 cursor-pointer'
            onClick={() => {
              destinationAddressClick(suggestion);
            }}
          >
            {suggestion.name},
            {suggestion.context?.region?.name
              ? suggestion.context?.region?.name
              : ""}
            ,{suggestion.context?.district?.name}
          </h2>
        )
      )}
    </div>
  ) : (

```

```
        "  
    })  
  </div>  
</div>  
);  
}  
export default Address;
```