Implementation Document

for

Route Mate

Version <1.2>

Prepared by

Group 8:		Group Name: Access_denied
Palagiri Tousif Ahamad	220744	atousif22@iitk.ac.in
Tamidala Venkata Sai Pawan Chanukya Reddy	221120	venkatasai22@iitk.ac.in
Vetcha Pankaj Nath	221188	vpankaj22@iitk.ac.in
Sudhamandu Sai Nikhil	221095	ssain22@iitk.ac.in
Mohammed Anas	220654	mohdanas22@iitk.ac.in
Himanshu Shekhar	220454	shimanshu22@iitk.ac.in
Swayamsidh Pradhan	221117	spradhan22@iitk.ac.in
Kawaljeet Singh	220514	kawaljeets22@iitk.ac.in
Anya Rajan	220191	anyarajan22@iitk.ac.in
Daksh Kumar Singh	220322	dakshks22@iitk.ac.in

Course: CS253

Mentor TA: *Sumit*

Date: 15-03-2024

CONTENTS II
REVISIONS II

1 IMPLEMENTATION DETAILS 1

2 CODEBASE 3

3 COMPLETENESS 4

APPENDIX A - GROUP LOG 6

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
version 1.0	Sai Nikhil	Implementation details and completeness of the product are added to the document.	13/03/24
version 1.2	Sai Nikhil	Appendix about the Group meetings is added	14/03/24

1 Implementation Details

Programming Language, Framework and Libraries for Frontend:

RouteMate is written in **Javascript**, **HTML** and **CSS.**We have used **React**, a javascript based open-source framework widely used for Web development.React offers the following advantages.

1.Component-Based Architecture: React follows a component-based architecture, allowing developers to break down complex UIs into smaller, reusable components. This modular approach makes code organization easier, enhances maintainability and reusability.

2.Virtual DOM: React utilizes a virtual DOM to efficiently update the UI. Instead of directly manipulating the real DOM for every change, React creates a lightweight copy of the DOM in memory and updates it based on the changes.

3.Declarative Syntax: React uses a declarative programming paradigm, where developers describe the desired UI state. This approach simplifies the process of building and reasoning about UIs.

4.One-Way Data Binding: React enforces one-way data flow, meaning data flows in a single direction from parent to child components. This approach reduces the likelihood of bugs and makes data flow more predictable.

Considering the above advantages of React, it is no wonder that some of the most popular apps today, Instagram, Facebook and Netflix, are built in React.

Programming Language, Framework and Libraries for Backend:

The Backend part of RouteMate is written in **Javascript** making use of Node.js runtime environment.

1.Express.js is used for building web servers.

2.WebSockets is used to implement the Chat feature on the Web.

3.Passport.js is used for implementing the Authentication protocol.

Database:

We have used PostgreSQL, an open source Relational database management system(RDBMS) known for its reliability, robustness and extensibility. The advantages of using PostgreSQL are listed below

- Open-source and community driven
- Relational database Management system
- Supports for ACID transactions and multi-version concurrency control(MVCC)
- Supports foreign key constraints for maintaining data integrity

2 Codebase

GIT Repository link : https://github.com/Python-Pi/CS253-CSR

To begin, clone the Git repository. Ensure Git is installed before proceeding.

git clone https://github.com/Python-Pi/CS253-CSR.git

Setting Up React Front-end server

- Navigate to the client directory.

  `cd /client`

- Install required Node.js modules.

  `npm i`

- Initiate the server

  `npm start`

- Upon successful installation of all necessary modules, create a .env file within the client directory. Add the following line:

  `REACT_APP_IP = '{IP ADDRESS OF YOUR MACHINE}'.`

Setting Up PostgreSQL Database

- Install PostgreSQL and setup username and password
- Create a database named cs253.

  `CREATE DATABASE CS253`

- Execute all the query commands in '`query.sql`'

Running Node Back-end server

- Navigate to the server directory.

  `cd /server`

- Install required Node.js modules.

  `npm i`

- Initiate the server.

  `npm start`

- Upon successful installation of all necessary modules, create a '*.env'* file within the client directory. Add the following lines:

  `PG_HOST = 'localhost'`

  `PG_USER = '{PostgreSQL Username}'`

```
PG_PASSWORD = '{PostgreSQL Password}'
PG_PORT = '5432'
SESSION_SECRET = '{Any sufficiently random string}'
PG_DATABASE = 'cs253'
IP = '{YOUR IP ADDRESS}'
PORT = '3000'
EMAIL_ID = ‘siriussnape880@gmail.com’
EMAIL = ‘snij qpys ijhw boeo’
```

Note :

The default command for running Node server is set to nodemon server.js in package.json. If you don't have nodemon installed, change it to node server.js.

3 Completeness

Ideas Implemented from SRS

The users are trusted and verified people from IIT Kanpur.

- We have Implemented both Personalised trips and travel groups based on destination locations.
- There is a chat section dedicated to each of the personalised trips and for each train running on that day.
- There is a unique profile for each trip created by the user and for the group travel
- Each train shows the number of users who have confirmed their trips and who have their tentative journeys on the particular date.
- Users can create new trips which they are planning to go and can include other users who are willing to join.
- A separate Blog for each of the personalised trips and a common blog for a particular train.

Future Improvements

- Implementing the expense splitting feature in which users can get to know the amount of money they need to pay to other users in case of uneven expenses during the trip.
- Reset password functionality via email authentication using SMTP server.
- Adding the same facilities for buses and flights which were previously implemented for trains.
- In case of home travel, suggesting possible groups for cab sharing.
- Users can get a chance to upload Images and videos in the blogs to convey their experiences more vividly allowing for a deeper connection with the audience
- Users will have their own profile and have the option to connect with others in the site which helps in makings groups and suggests user about the trips that they are part of.

Appendix A - Group Log

S.No	Meet Date	Topics Discussed	Duration	Members Attended
1	10th Feb 2024	Discussed what language to work upon and Frameworks to use	120 mins	10
2	12th Feb 2024	Finalised upon the languages and Framework	90 mins	10
3	15th Feb 2024	We discussed how to implement the features of the product and had a rough distribution of the work	120 mins	10
4	17th Feb 2024	We finalised upon the work distribution and started to make individual components	60 mins	10
5	27th Feb 2024	We cleared each others doubts about the implementation and improved upon the each others idea of implementing	90 mins	10
6	02nd Mar 2024	We discussed about common parts of the components so that the final product looks uniform	60 mins	10
7	08th Mar 2024	Everyone brought their individual works and we decided upon the common factors which should look same throughout the product	90 mins	10
8	10th Mar 2024	We made a git repo of the project and pushed our individual works into it and laid rules to edit the repo	60 mins	10

9	12th Mar 2024	We started integrating parts of Frontend	120 mins	10
10	13th Mar 2024	We finished integrating Frontend parts and started working upon integrating frontend with backend.	150 mins	10
11	14th Mar 2024	We made final touch ups to the product	60 mins	10
12	15th Mar 2024	Final meet before submission	60 mins	10