Group 4 - Uber Simulation

<u>Manjot Singh</u> (017557462)

Manjot played a key role in designing and developing the backend architecture for the project using Django. They set up the core project structure, defined models to manage drivers, customers, rides, billing, and administrative tasks, and built RESTful APIs with the Django REST Framework. They also integrated a dynamic pricing algorithm into the billing system and ensured seamless database operations by connecting MySQL for relational data and MongoDB for multimedia storage. Furthermore, they configured API endpoints to handle CRUD operations and thoroughly tested the backend functionality using Django's built-in development server.

<u>Sai Prasad Thalluri</u> (017512781)

Sai was responsible for developing the frontend of the application using ReactJS. They created components for drivers, customers, rides, billing, admin tasks, and dynamic pricing, focusing on delivering a user-friendly and intuitive interface for all stakeholders. To ensure smooth navigation, they implemented React Router and connected the frontend to the backend APIs using Axios for real-time data exchange. Additionally, they worked on responsive design to enhance usability across devices and implemented proper form validation and error handling to improve the overall user experience.

<u>Darpankumar Jiyani</u> (017536623)

Darpan took charge of setting up and integrating the databases for the project. They designed the MySQL schema to manage relational data and set up MongoDB to store multimedia files, such as driver images. To enhance performance, they configured Redis to cache frequently used SQL queries. Additionally, they prepared and preprocessed the Uber fares dataset and trained the dynamic pricing model using machine learning algorithms. They also focused on ensuring database security, implementing proper indexing, and validating data to maintain integrity and efficiency.

<u>Saqib Chowdhury</u> (017514978)

Saqib was responsible for deployment and setting up real-time communication systems. They implemented Kafka for real-time messaging, creating producer and consumer scripts to manage updates such as driver availability and ride statuses. They containerized the backend with Docker and configured a Docker Compose setup to streamline integration with MySQL. For deployment, they used Kubernetes on AWS, writing deployment and service YAML files to ensure smooth orchestration. Their efforts focused on ensuring the system's scalability, fault tolerance, and load balancing to handle varying loads effectively.

Object Management Policy:

- Use **JWT-based authentication** for secure user access to resources.
- Implement **Role-Based Access Control (RBAC)** to manage permissions for Admin, Driver, and Rider roles.
- Validate incoming requests to ensure **data consistency** at both API and database levels.
- Verify **driver and car availability** before confirming ride bookings.
- Use **PATCH** requests for partial updates to prevent data overwrites.
- Log critical operations such as **POST**, **PUT**, **DELETE**, and **PATCH** for audit trails.
- Retain booking data for a predefined period to meet business requirements.
- Allow users to delete saved addresses via endpoints like `/riders/saved-addresses/{id}`.
- Apply **rate-limiting** to restrict users to 100 requests per minute.
- Set **booking quotas per user** to avoid system overload and ensure fair usage.
- Use appropriate **HTTP status codes** for clear communication (e.g., `200`, `400`, `401`, `404`, `500`).
- Provide clear, actionable, and user-friendly **error messages** in API responses.
- Encrypt sensitive data **at rest and in transit** to protect user information.
- Sanitize and validate all user inputs to prevent **injection attacks**.
- Maintain relationships between resources (e.g., drivers linked to cars, riders linked to bookings) to preserve **referential integrity**.
- Implement cascading updates and deletions to keep data relationships consistent.
- Introduce **pagination** for large datasets in endpoints like `/rides/list-booking`.
- Use **caching** for frequently accessed resources (e.g., `/app/site`) to improve performance.

How we handled heavyweight resources:

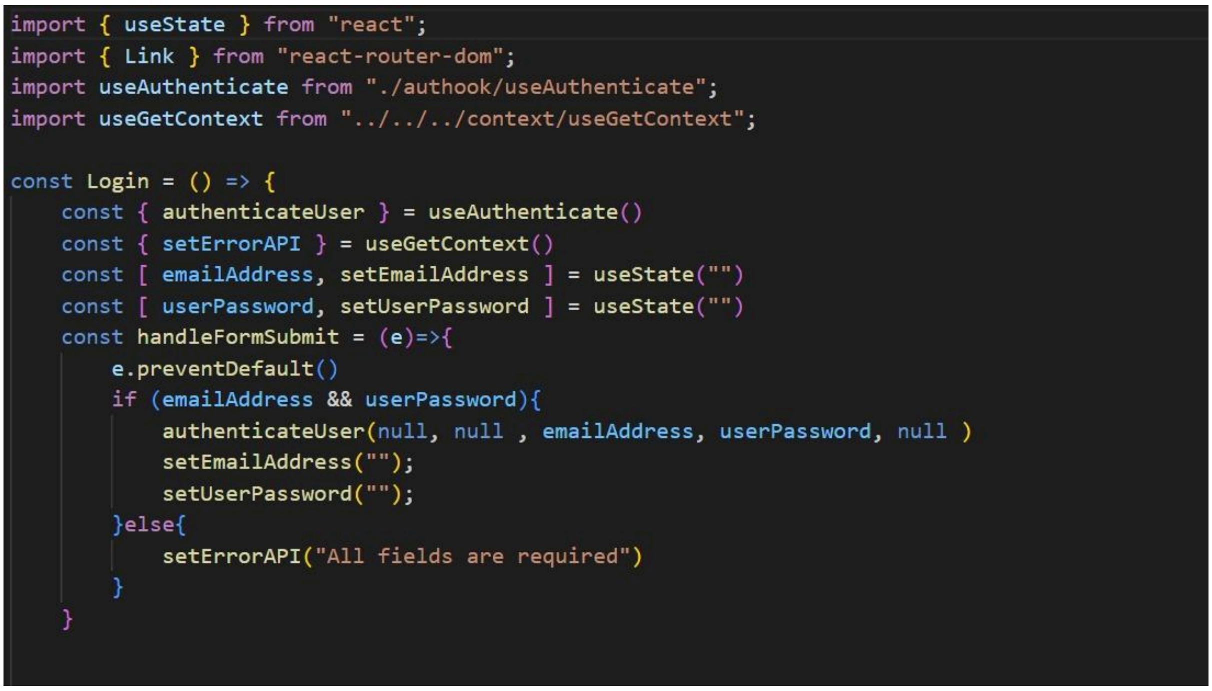
Heavy resources, like big datasets, multimedia files, and tasks that need a lot of computing power, are handled carefully to keep the system fast and scalable. Large datasets are stored using MySQL for structured data and MongoDB for unstructured data like images or videos. Redis is used to cache commonly accessed queries, so the main database doesn't get overloaded. For tasks that need heavy processing, like training machine learning models or calculating dynamic prices, the work is done in the background or offline, and the results are saved for quick access when needed.

Policy we used to decide when to write data into the database:

The policy for deciding when to write data into the database is based on the criticality and frequency of the data being processed. For real-time operations, such as ride bookings or status updates, data is written immediately to ensure system accuracy and consistency. Less critical or high-frequency data, like user activity logs, is batched and written periodically to reduce database load. Transactional data, such as billing records or payments, is written synchronously to ensure integrity and avoid loss. Additionally, temporary data used for caching or intermediate calculations is stored in Redis, while persistent, long-term data is committed to the database after thorough validation to maintain data quality.

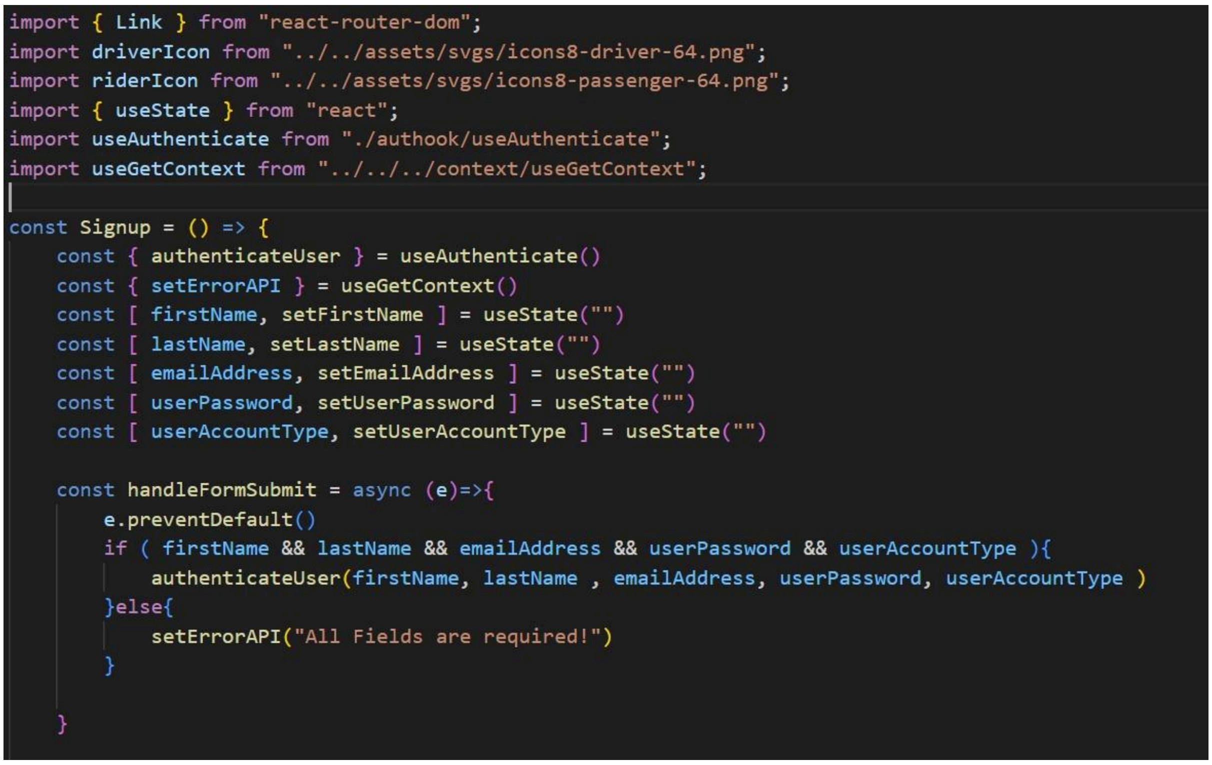
Code Snippets of our application

Login.js



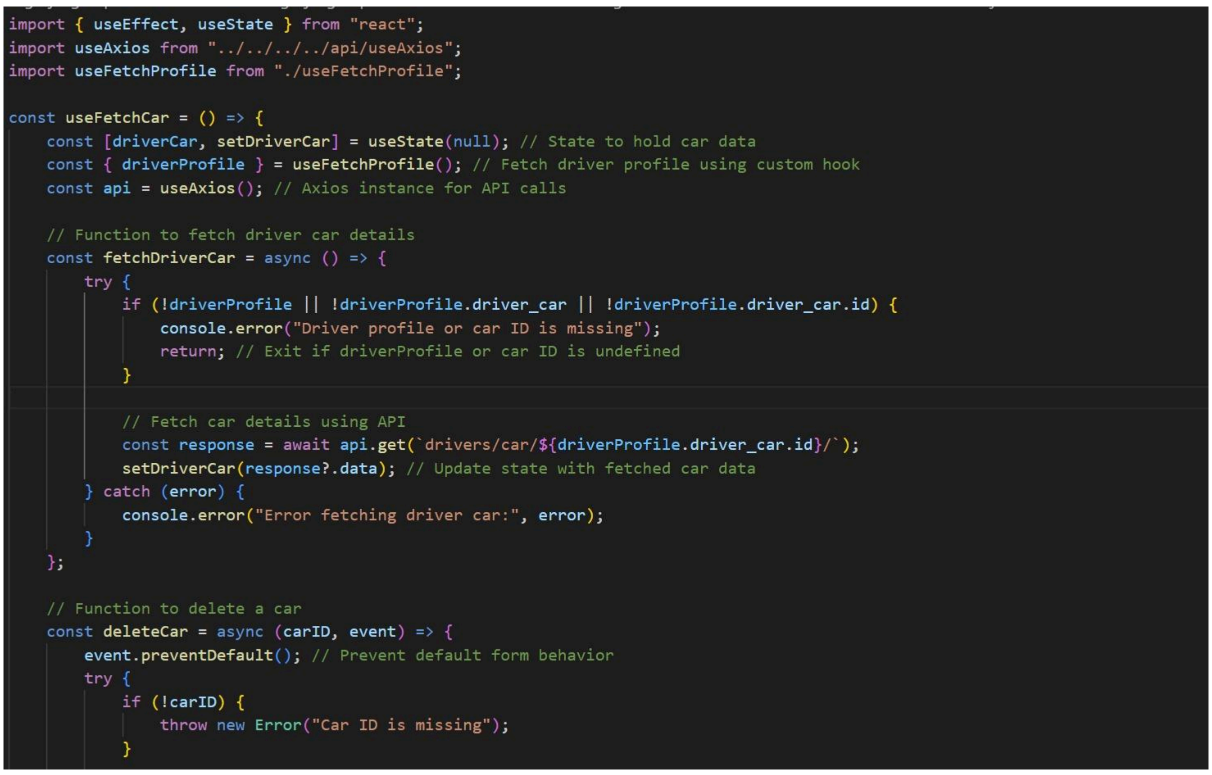
```
import { useState } from "react";
import { Link } from "react-router-dom";
import useAuthenticate from "./authook/useAuthenticate";
import useGetContext from "../../../context/useGetContext";

const Login = () => {
    const { authenticateUser } = useAuthenticate()
    const { setErrorAPI } = useGetContext()
    const [ emailAddress, setEmailAddress ] = useState("")
    const [ userPassword, setUserPassword ] = useState("")
    const handleFormSubmit = (e)=>{
        e.preventDefault()
        if (emailAddress && userPassword){
            authenticateUser(null, null , emailAddress, userPassword, null )
            setEmailAddress("");
            setUserPassword("");
        }else{
            setErrorAPI("All fields are required")
        }
    }
```

Singup.js

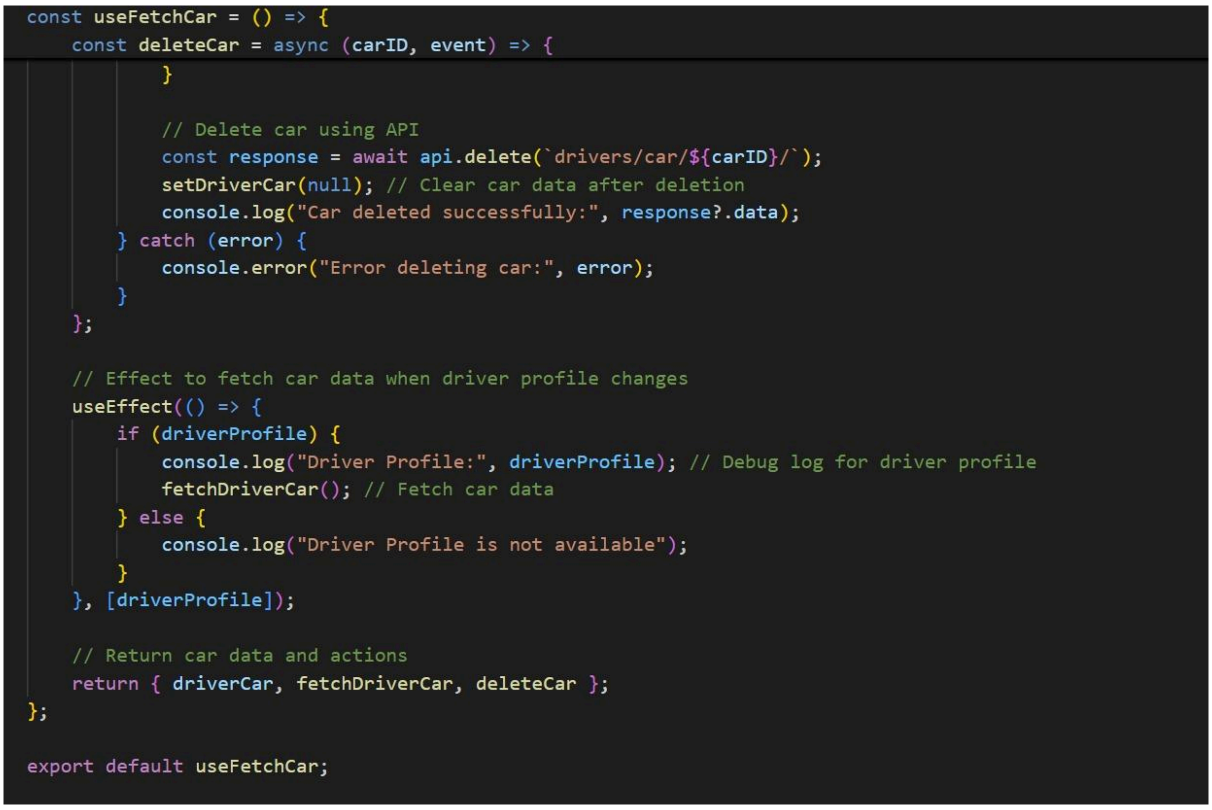


```
import { Link } from "react-router-dom";
import driverIcon from "../../assets/svgs/icons8-driver-64.png";
import riderIcon from "../../assets/svgs/icons8-passenger-64.png";
import { useState } from "react";
import useAuthenticate from "./authook/useAuthenticate";
import useGetContext from "../../../context/useGetContext";

const Signup = () => {
    const { authenticateUser } = useAuthenticate()
    const { setErrorAPI } = useGetContext()
    const [ firstName, setFirstName ] = useState("")
    const [ lastName, setLastName ] = useState("")
    const [ emailAddress, setEmailAddress ] = useState("")
    const [ userPassword, setUserPassword ] = useState("")
    const [ userAccountType, setUserAccountType ] = useState("")

    const handleFormSubmit = async (e)=>{
        e.preventDefault()
        if ( firstName && lastName && emailAddress && userPassword && userAccountType ){
            authenticateUser(firstName, lastName , emailAddress, userPassword, userAccountType )
        }else{
            setErrorAPI("All Fields are required!")
        }

    }
```

useFetchCar.js

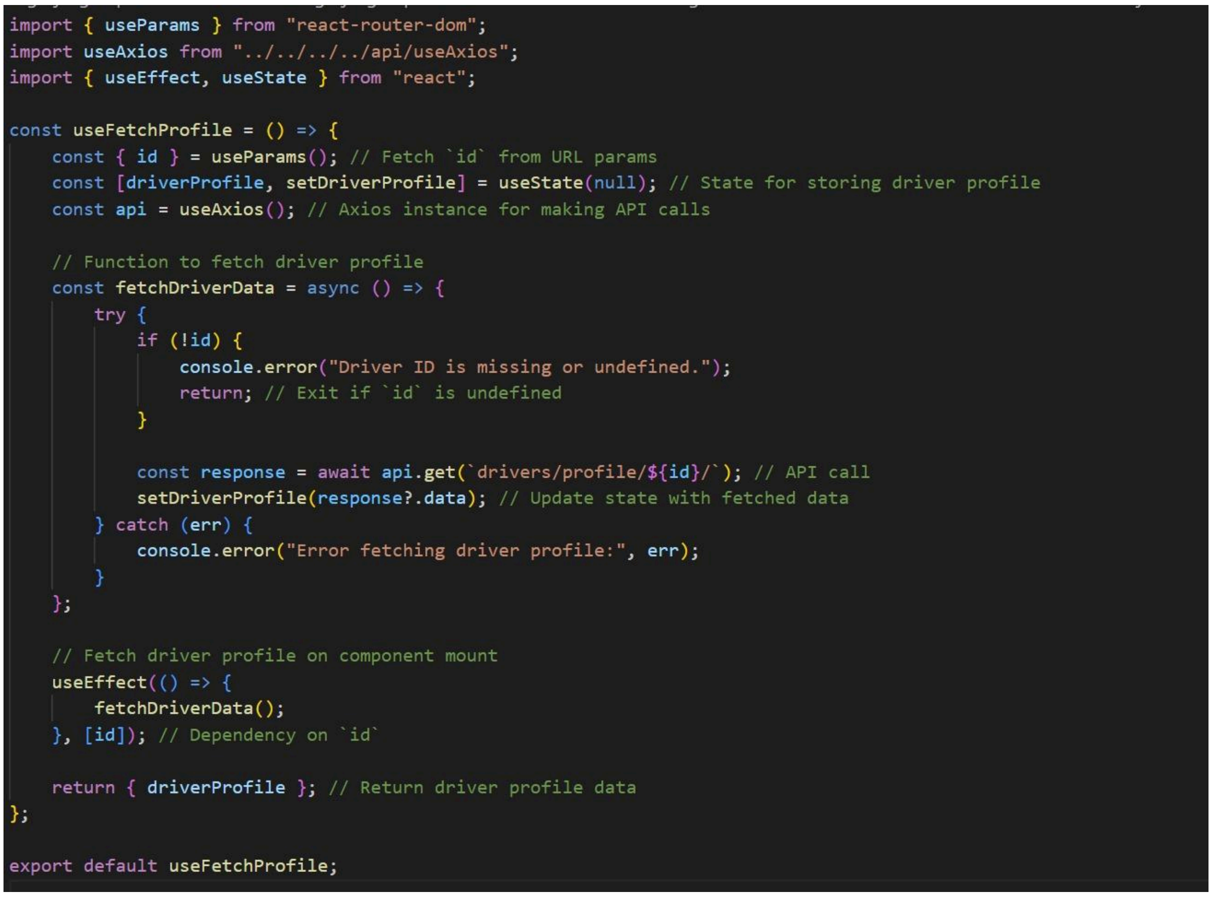


```
import { useEffect, useState } from "react";
import useAxios from "../../../../api/useAxios";
import useFetchProfile from "./useFetchProfile";

const useFetchCar = () => {
    const [driverCar, setDriverCar] = useState(null); // State to hold car data
    const { driverProfile } = useFetchProfile(); // Fetch driver profile using custom hook
    const api = useAxios(); // Axios instance for API calls

    // Function to fetch driver car details
    const fetchDriverCar = async () => {
        try {
            if (!driverProfile || !driverProfile.driver_car || !driverProfile.driver_car.id) {
                console.error("Driver profile or car ID is missing");
                return; // Exit if driverProfile or car ID is undefined
            }

            // Fetch car details using API
            const response = await api.get(`drivers/car/${driverProfile.driver_car.id}/`);
            setDriverCar(response?.data); // Update state with fetched car data
        } catch (error) {
            console.error("Error fetching driver car:", error);
        }
    };

    // Function to delete a car
    const deleteCar = async (carID, event) => {
        event.preventDefault(); // Prevent default form behavior
        try {
            if (!carID) {
                throw new Error("Car ID is missing");
            }
```



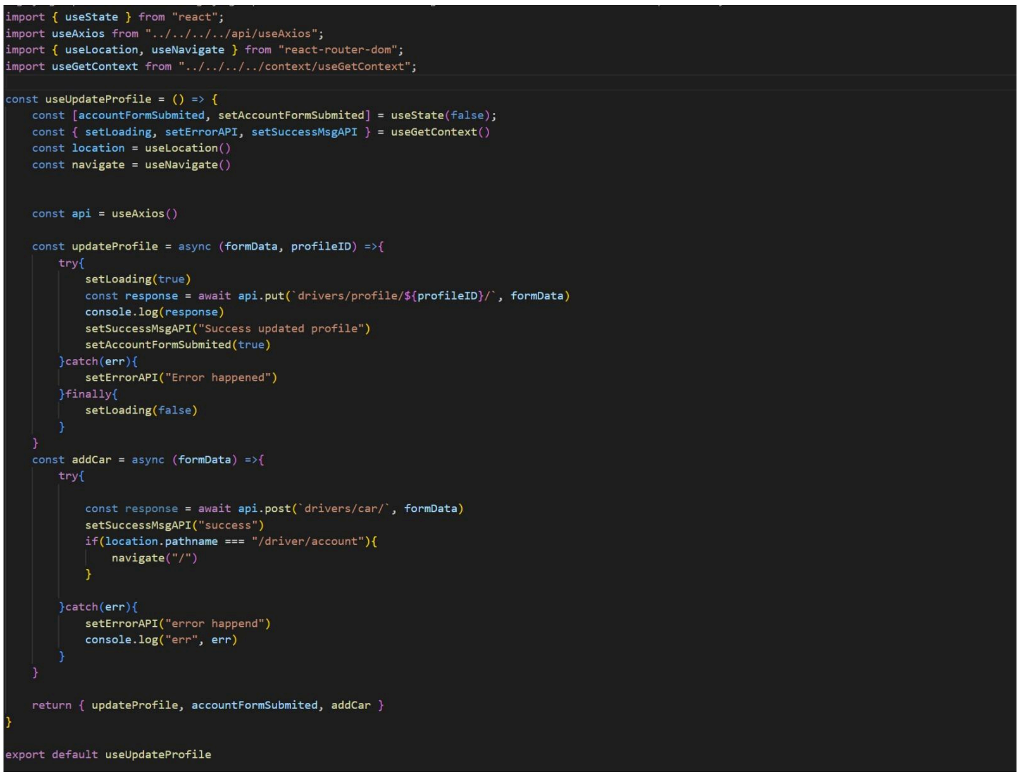
```
const useFetchCar = () => {
    const deleteCar = async (carID, event) => {
            }

            // Delete car using API
            const response = await api.delete(`drivers/car/${carID}/`);
            setDriverCar(null); // Clear car data after deletion
            console.log("Car deleted successfully:", response?.data);
        } catch (error) {
            console.error("Error deleting car:", error);
        }
    };

    // Effect to fetch car data when driver profile changes
    useEffect(() => {
        if (driverProfile) {
            console.log("Driver Profile:", driverProfile); // Debug log for driver profile
            fetchDriverCar(); // Fetch car data
        } else {
            console.log("Driver Profile is not available");
        }
    }, [driverProfile]);

    // Return car data and actions
    return { driverCar, fetchDriverCar, deleteCar };
};

export default useFetchCar;
```

useFetchProfile:



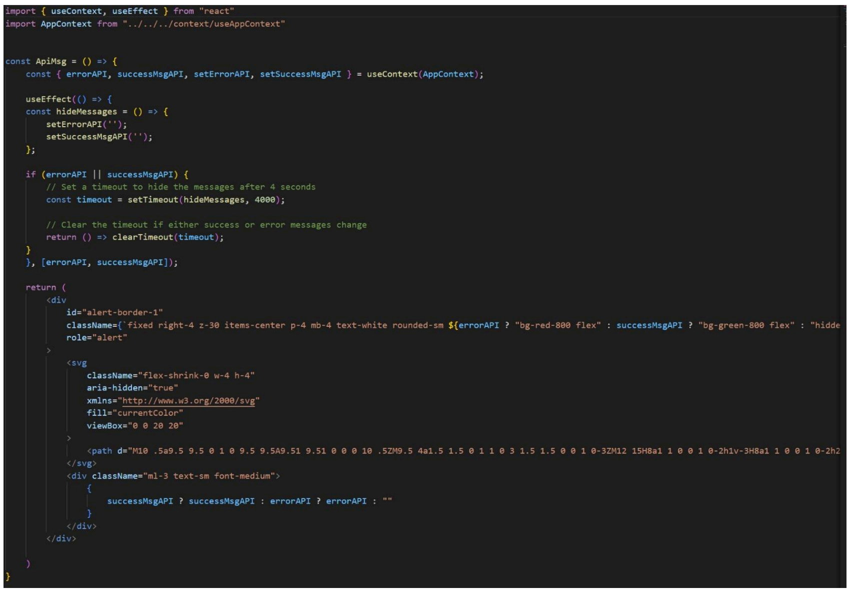
```javascript
import { useParams } from "react-router-dom";
import useAxios from "../../../../api/useAxios";
import { useEffect, useState } from "react";

const useFetchProfile = () => {
    const { id } = useParams(); // Fetch `id` from URL params
    const [driverProfile, setDriverProfile] = useState(null); // State for storing driver profile
    const api = useAxios(); // Axios instance for making API calls

    // Function to fetch driver profile
    const fetchDriverData = async () => {
        try {
            if (!id) {
                console.error("Driver ID is missing or undefined.");
                return; // Exit if `id` is undefined
            }

            const response = await api.get(`drivers/profile/${id}/`); // API call
            setDriverProfile(response?.data); // Update state with fetched data
        } catch (err) {
            console.error("Error fetching driver profile:", err);
        }
    };

    // Fetch driver profile on component mount
    useEffect(() => {
        fetchDriverData();
    }, [id]); // Dependency on `id`

    return { driverProfile }; // Return driver profile data
};

export default useFetchProfile;
```

useUpdateProfile.js



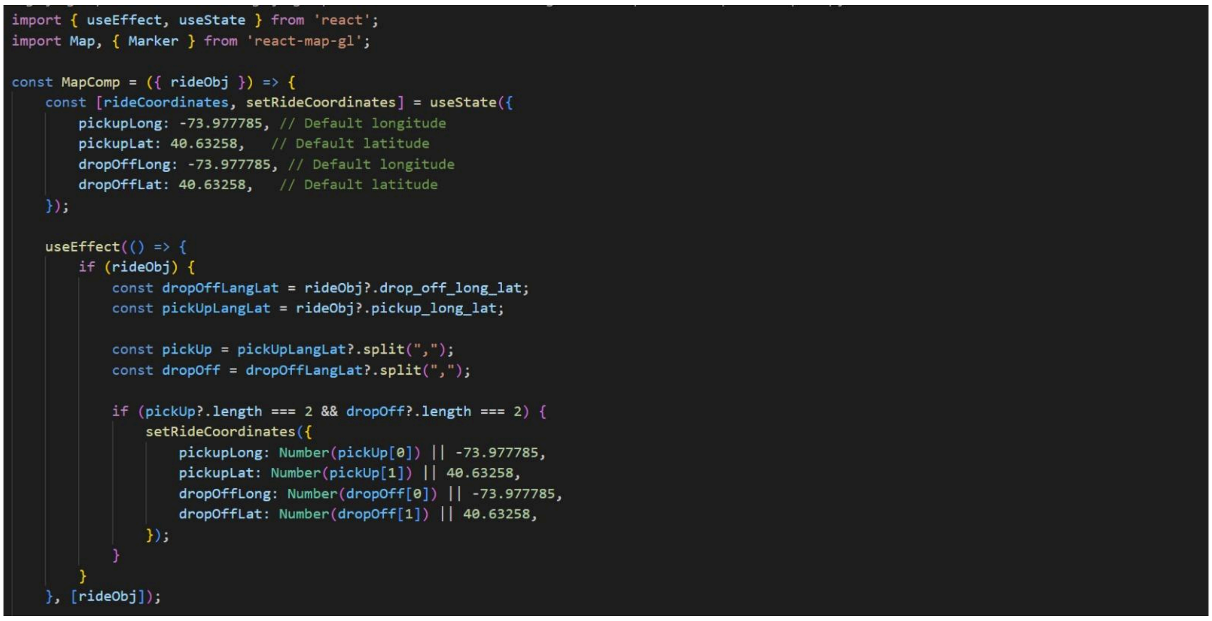
```javascript
import { useState } from "react";
import useAxios from "../../../../api/useAxios";
import { useLocation, useNavigate } from "react-router-dom";
import useGetContext from "../../../../context/useGetContext";

const useUpdateProfile = () => {
    const [accountFormSubmited, setAccountFormSubmited] = useState(false);
    const { setLoading, setErrorAPI, setSuccessMsgAPI } = useGetContext()
    const location = useLocation()
    const navigate = useNavigate()


    const api = useAxios()

    const updateProfile = async (formData, profileID) =>{
        try{
            setLoading(true)
            const response = await api.put(`drivers/profile/${profileID}/`, formData)
            console.log(response)
            setSuccessMsgAPI("Success updated profile")
            setAccountFormSubmited(true)
        }catch(err){
            setErrorAPI("Error happened")
        }finally{
            setLoading(false)
        }
    }
    const addCar = async (formData) =>{
        try{

            const response = await api.post(`drivers/car/`, formData)
            setSuccessMsgAPI("success")
            if(location.pathname === "/driver/account"){
                navigate("/")
            }

        }catch(err){
            setErrorAPI("error happend")
            console.log("err", err)
        }
    }

    return { updateProfile, accountFormSubmited, addCar }
}

export default useUpdateProfile
```

ApiMSG.js

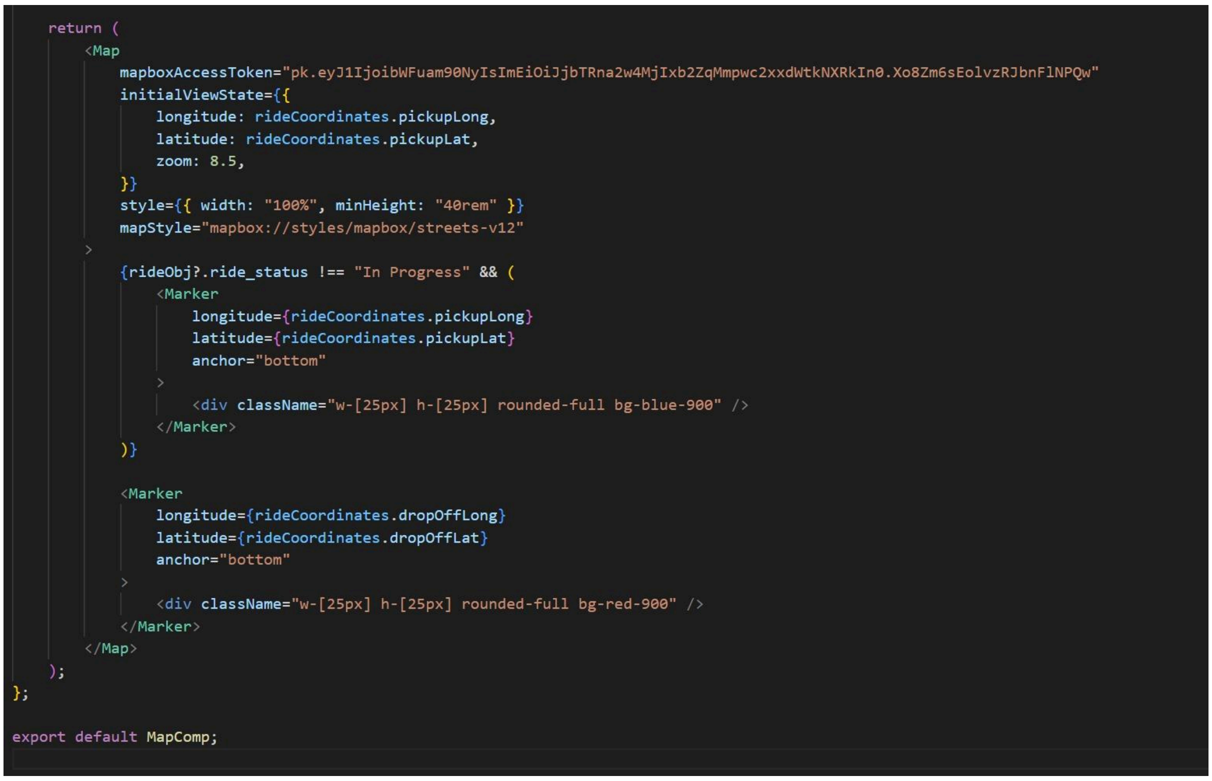


```
import { useContext, useEffect } from "react"
import AppContext from "../../../context/useAppContext"


const ApiMsg = () => {
    const { errorAPI, successMsgAPI, setErrorAPI, setSuccessMsgAPI } = useContext(AppContext);

    useEffect(() => {
    const hideMessages = () => {
        setErrorAPI('');
        setSuccessMsgAPI('');
    };

    if (errorAPI || successMsgAPI) {
        // Set a timeout to hide the messages after 4 seconds
        const timeout = setTimeout(hideMessages, 4000);

        // Clear the timeout if either success or error messages change
        return () => clearTimeout(timeout);
    }
    }, [errorAPI, successMsgAPI]);

    return (
        <div
            id="alert-border-1"
            className={`fixed right-4 z-30 items-center p-4 mb-4 text-white rounded-sm ${errorAPI ? "bg-red-800 flex" : successMsgAPI ? "bg-green-800 flex" : "hidde
            role="alert"
        >
            <svg
                className="flex-shrink-0 w-4 h-4"
                aria-hidden="true"
                xmlns="http://www.w3.org/2000/svg"
                fill="currentColor"
                viewBox="0 0 20 20"
            >
                <path d="M10 .5a9.5 9.5 0 1 0 9.5 9.5A9.51 9.51 0 0 0 10 .5ZM9.5 4a1.5 1.5 0 1 1 0 3 1.5 1.5 0 0 1 0-3ZM12 15H8a1 1 0 0 1 0-2h1v-3H8a1 1 0 0 1 0-2h2
            </svg>
            <div className="ml-3 text-sm font-medium">
                {
                    successMsgAPI ? successMsgAPI : errorAPI ? errorAPI : ""
                }
            </div>
        </div>

    )
}
```

MapComp.js

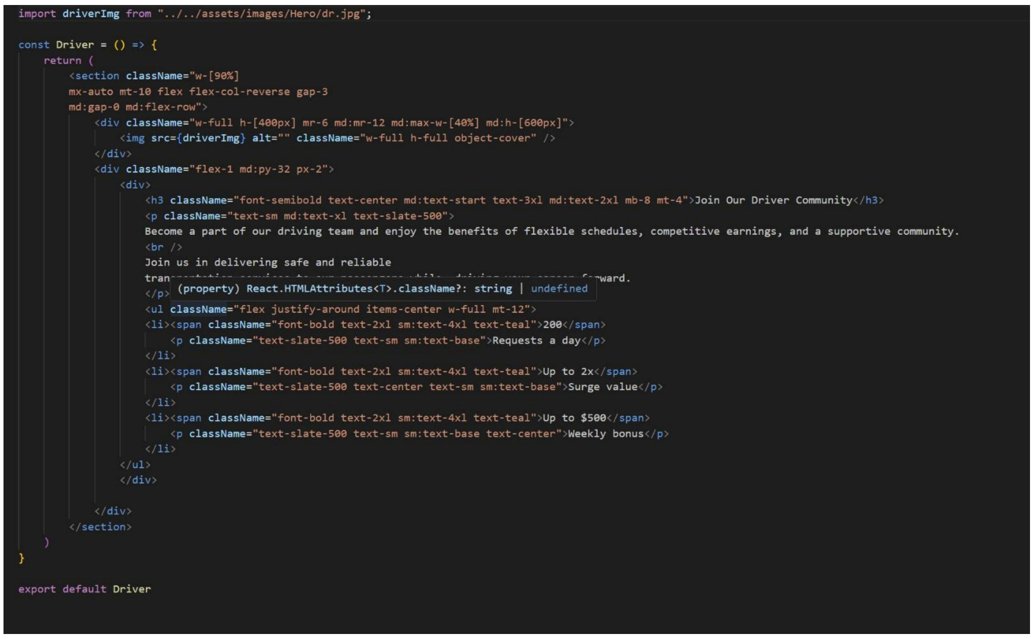


```
import { useEffect, useState } from 'react';
import Map, { Marker } from 'react-map-gl';

const MapComp = ({ rideObj }) => {
    const [rideCoordinates, setRideCoordinates] = useState({
        pickupLong: -73.977785, // Default longitude
        pickupLat: 40.63258,   // Default latitude
        dropOffLong: -73.977785, // Default longitude
        dropOffLat: 40.63258,   // Default latitude
    });

    useEffect(() => {
        if (rideObj) {
            const dropOffLangLat = rideObj?.drop_off_long_lat;
            const pickUpLangLat = rideObj?.pickup_long_lat;

            const pickUp = pickUpLangLat?.split(",");
            const dropOff = dropOffLangLat?.split(",");

            if (pickUp?.length === 2 && dropOff?.length === 2) {
                setRideCoordinates({
                    pickupLong: Number(pickUp[0]) || -73.977785,
                    pickupLat: Number(pickUp[1]) || 40.63258,
                    dropOffLong: Number(dropOff[0]) || -73.977785,
                    dropOffLat: Number(dropOff[1]) || 40.63258,
                });
            }
        }
    }, [rideObj]);
```



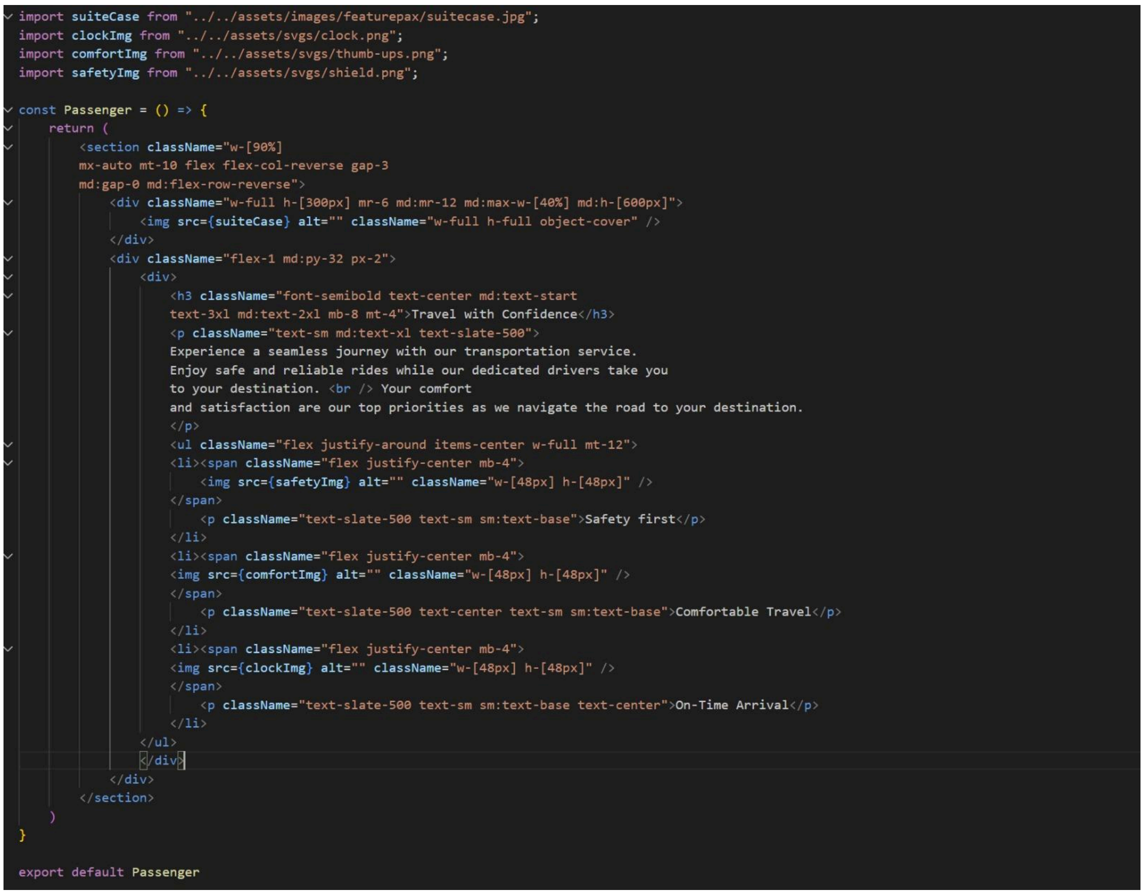
```
    return (
        <Map
            mapboxAccessToken="pk.eyJ1IjoibWFuam90NyIsImEiOiJjbTRna2w4MjIxb2ZqMmpwc2xxdWtkNXRkIn0.Xo8Zm6sEolvzRJbnFlNPQw"
            initialViewState={{
                longitude: rideCoordinates.pickupLong,
                latitude: rideCoordinates.pickupLat,
                zoom: 8.5,
            }}
            style={{ width: "100%", minHeight: "40rem" }}
            mapStyle="mapbox://styles/mapbox/streets-v12"
        >
            {rideObj?.ride_status !== "In Progress" && (
                <Marker
                    longitude={rideCoordinates.pickupLong}
                    latitude={rideCoordinates.pickupLat}
                    anchor="bottom"
                >
                    <div className="w-[25px] h-[25px] rounded-full bg-blue-900" />
                </Marker>
            )}

            <Marker
                longitude={rideCoordinates.dropOffLong}
                latitude={rideCoordinates.dropOffLat}
                anchor="bottom"
            >
                <div className="w-[25px] h-[25px] rounded-full bg-red-900" />
            </Marker>
        </Map>
    );
};

export default MapComp;
```

driver.js



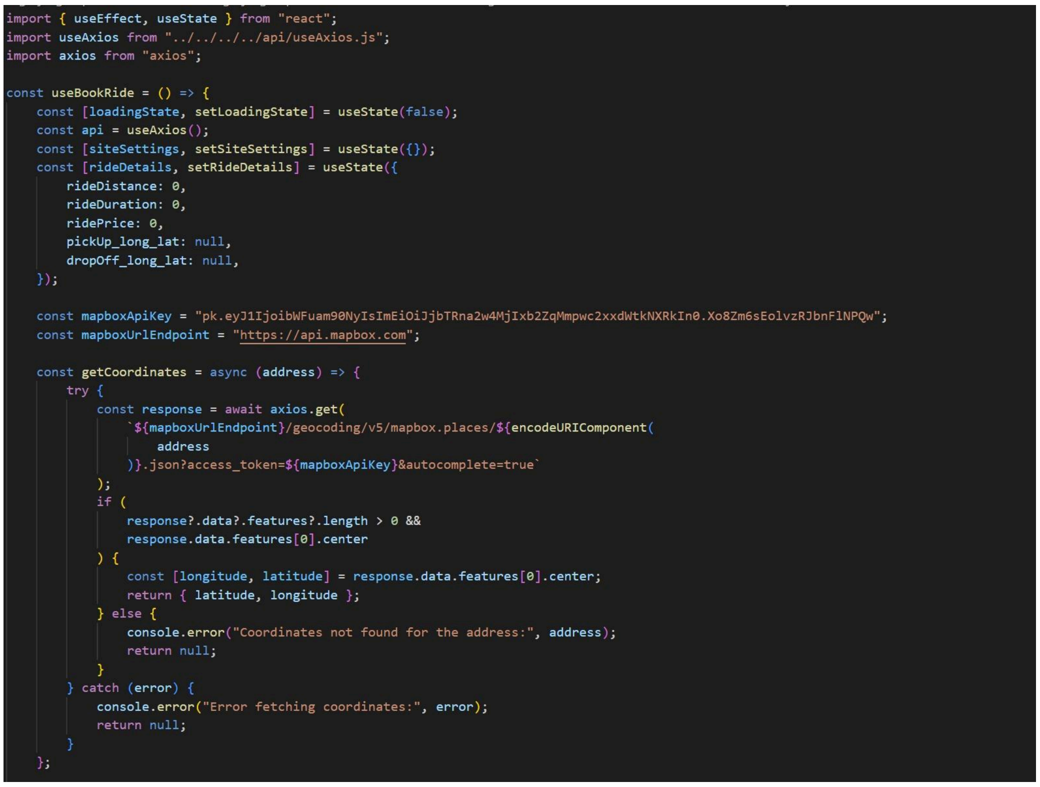
```
import driverImg from "../../assets/images/Hero/dr.jpg";

const Driver = () => {
    return (
        <section className="w-[90%]
        mx-auto mt-10 flex flex-col-reverse gap-3
        md:gap-0 md:flex-row">
            <div className="w-full h-[400px] mr-6 md:mr-12 md:max-w-[40%] md:h-[600px]">
                <img src={driverImg} alt="" className="w-full h-full object-cover" />
            </div>
            <div className="flex-1 md:py-32 px-2">
                <div>
                    <h3 className="font-semibold text-center md:text-start text-3xl md:text-2xl mb-8 mt-4">Join Our Driver Community</h3>
                    <p className="text-sm md:text-xl text-slate-500">
                    Become a part of our driving team and enjoy the benefits of flexible schedules, competitive earnings, and a supportive community.
                    <br />
                    Join us in delivering safe and reliable
                    tran[illegible]ward.
                    </p> (property) React.HTMLAttributes<T>.className?: string | undefined
                    <ul className="flex justify-around items-center w-full mt-12">
                    <li><span className="font-bold text-2xl sm:text-4xl text-teal">200</span>
                        <p className="text-slate-500 text-sm sm:text-base">Requests a day</p>
                    </li>
                    <li><span className="font-bold text-2xl sm:text-4xl text-teal">Up to 2x</span>
                        <p className="text-slate-500 text-center text-sm sm:text-base">Surge value</p>
                    </li>
                    <li><span className="font-bold text-2xl sm:text-4xl text-teal">Up to $500</span>
                        <p className="text-slate-500 text-sm sm:text-base text-center">Weekly bonus</p>
                    </li>
                </ul>
                </div>

            </div>
        </section>
    )
}

export default Driver
```

Passenger.js



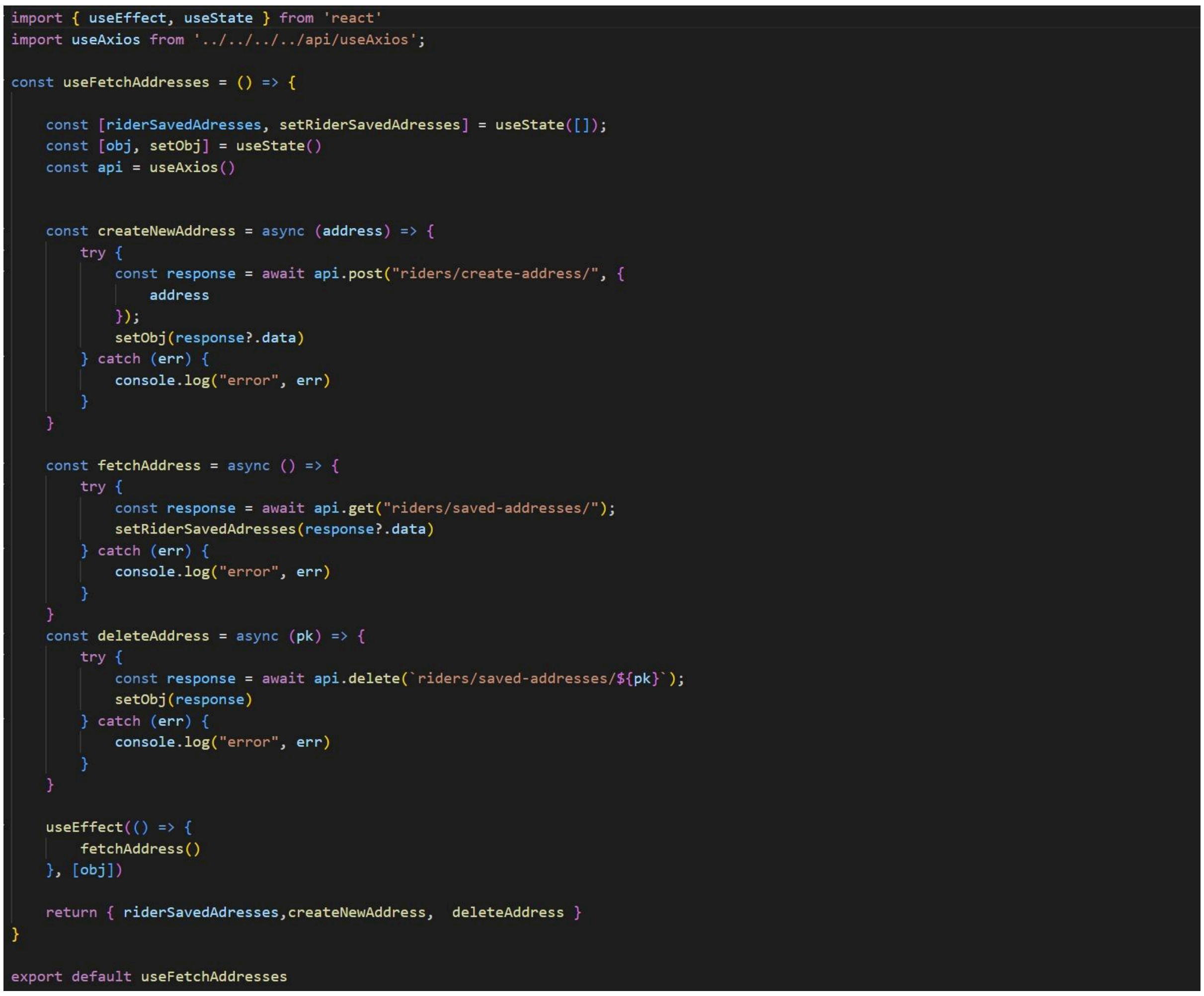
```
import suiteCase from "../../assets/images/featurepax/suitecase.jpg";
import clockImg from "../../assets/svgs/clock.png";
import comfortImg from "../../assets/svgs/thumb-ups.png";
import safetyImg from "../../assets/svgs/shield.png";

const Passenger = () => {
    return (
        <section className="w-[90%]
        mx-auto mt-10 flex flex-col-reverse gap-3
        md:gap-0 md:flex-row-reverse">
            <div className="w-full h-[300px] mr-6 md:mr-12 md:max-w-[40%] md:h-[600px]">
                <img src={suiteCase} alt="" className="w-full h-full object-cover" />
            </div>
            <div className="flex-1 md:py-32 px-2">
                <div>
                    <h3 className="font-semibold text-center md:text-start
                    text-3xl md:text-2xl mb-8 mt-4">Travel with Confidence</h3>
                    <p className="text-sm md:text-xl text-slate-500">
                    Experience a seamless journey with our transportation service.
                    Enjoy safe and reliable rides while our dedicated drivers take you
                    to your destination. <br /> Your comfort
                    and satisfaction are our top priorities as we navigate the road to your destination.
                    </p>
                    <ul className="flex justify-around items-center w-full mt-12">
                    <li><span className="flex justify-center mb-4">
                        <img src={safetyImg} alt="" className="w-[48px] h-[48px]" />
                    </span>
                        <p className="text-slate-500 text-sm sm:text-base">Safety first</p>
                    </li>
                    <li><span className="flex justify-center mb-4">
                    <img src={comfortImg} alt="" className="w-[48px] h-[48px]" />
                    </span>
                        <p className="text-slate-500 text-center text-sm sm:text-base">Comfortable Travel</p>
                    </li>
                    <li><span className="flex justify-center mb-4">
                    <img src={clockImg} alt="" className="w-[48px] h-[48px]" />
                    </span>
                        <p className="text-slate-500 text-sm sm:text-base text-center">On-Time Arrival</p>
                    </li>
                </ul>
                </div>
            </div>
        </section>
    )
}

export default Passenger
```

UseBookRide.js



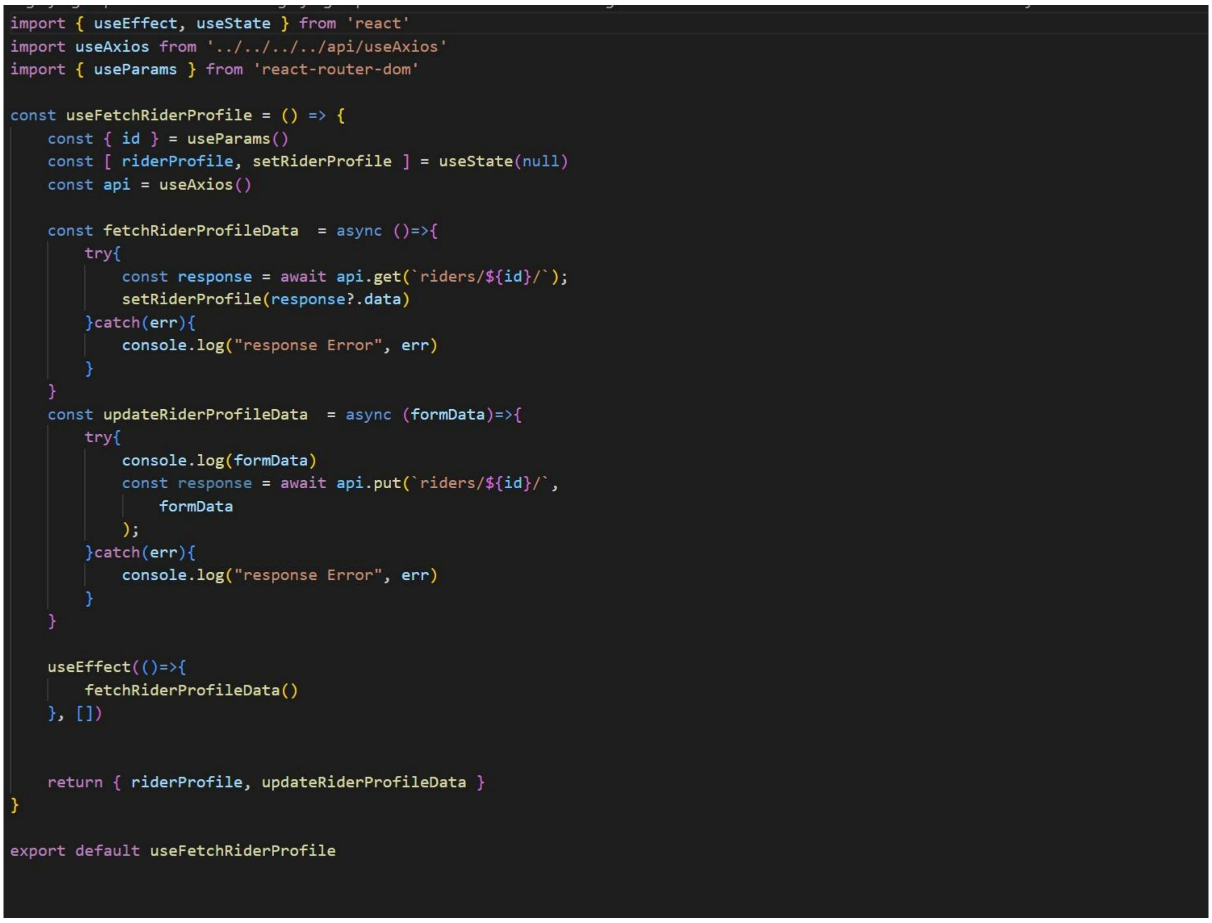
```
import { useEffect, useState } from "react";
import useAxios from "../../../../api/useAxios.js";
import axios from "axios";

const useBookRide = () => {
    const [loadingState, setLoadingState] = useState(false);
    const api = useAxios();
    const [siteSettings, setSiteSettings] = useState({});
    const [rideDetails, setRideDetails] = useState({
        rideDistance: 0,
        rideDuration: 0,
        ridePrice: 0,
        pickUp_long_lat: null,
        dropOff_long_lat: null,
    });

    const mapboxApiKey = "pk.eyJ1IjoibWFuam90NyIsImEiOiJjbTRna2w4MjIxb2ZqMmpwc2xxdWtkNXRkIn0.Xo8Zm6sEolvzRJbnFlNPQw";
    const mapboxUrlEndpoint = "https://api.mapbox.com";

    const getCoordinates = async (address) => {
        try {
            const response = await axios.get(
                `${mapboxUrlEndpoint}/geocoding/v5/mapbox.places/${encodeURIComponent(
                    address
                )}.json?access_token=${mapboxApiKey}&autocomplete=true`
            );
            if (
                response?.data?.features?.length > 0 &&
                response.data.features[0].center
            ) {
                const [longitude, latitude] = response.data.features[0].center;
                return { latitude, longitude };
            } else {
                console.error("Coordinates not found for the address:", address);
                return null;
            }
        } catch (error) {
            console.error("Error fetching coordinates:", error);
            return null;
        }
    };
```

UseFetchAddresses.js



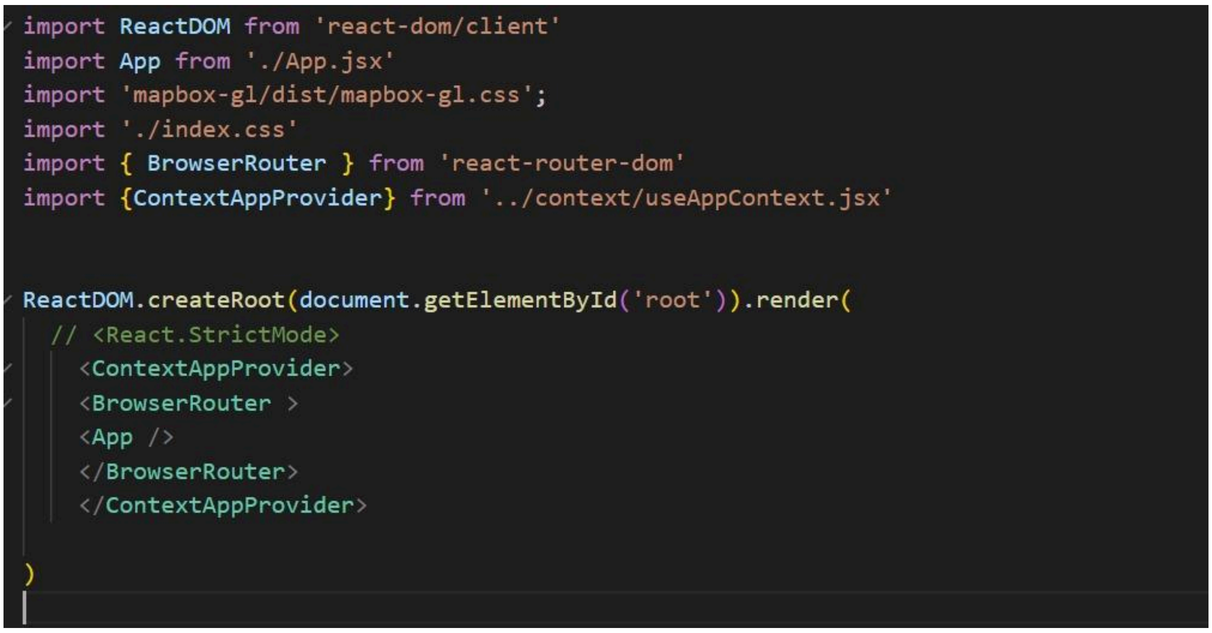
```javascript
import { useEffect, useState } from 'react'
import useAxios from '../../../../api/useAxios';

const useFetchAddresses = () => {

    const [riderSavedAdresses, setRiderSavedAdresses] = useState([]);
    const [obj, setObj] = useState()
    const api = useAxios()


    const createNewAddress = async (address) => {
        try {
            const response = await api.post("riders/create-address/", {
                address
            });
            setObj(response?.data)
        } catch (err) {
            console.log("error", err)
        }
    }

    const fetchAddress = async () => {
        try {
            const response = await api.get("riders/saved-addresses/");
            setRiderSavedAdresses(response?.data)
        } catch (err) {
            console.log("error", err)
        }
    }
    const deleteAddress = async (pk) => {
        try {
            const response = await api.delete(`riders/saved-addresses/${pk}`);
            setObj(response)
        } catch (err) {
            console.log("error", err)
        }
    }

    useEffect(() => {
        fetchAddress()
    }, [obj])

    return { riderSavedAdresses,createNewAddress,  deleteAddress }
}

export default useFetchAddresses
```

UseFetchRiderProfile.js



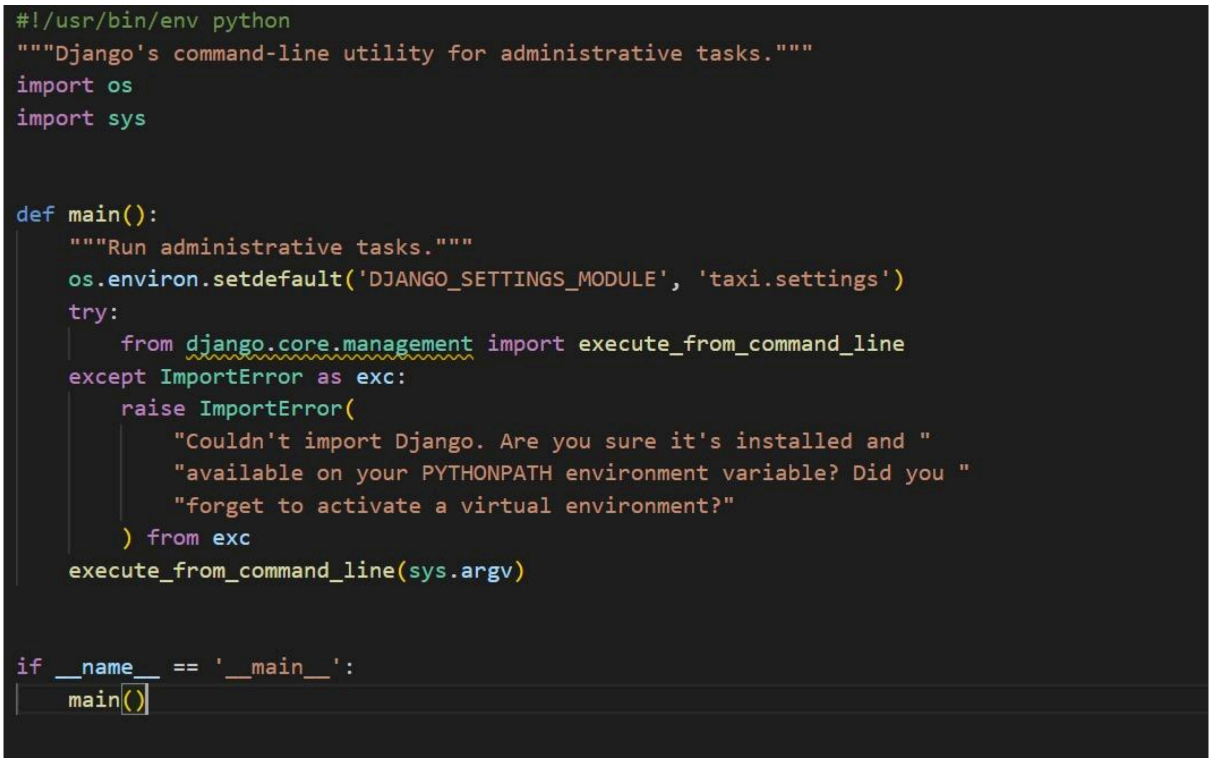
```javascript
import { useEffect, useState } from 'react'
import useAxios from '../../../../api/useAxios'
import { useParams } from 'react-router-dom'

const useFetchRiderProfile = () => {
    const { id } = useParams()
    const [ riderProfile, setRiderProfile ] = useState(null)
    const api = useAxios()

    const fetchRiderProfileData  = async ()=>{
        try{
            const response = await api.get(`riders/${id}/`);
            setRiderProfile(response?.data)
        }catch(err){
            console.log("response Error", err)
        }
    }
    const updateRiderProfileData  = async (formData)=>{
        try{
            console.log(formData)
            const response = await api.put(`riders/${id}/`,
                formData
            );
        }catch(err){
            console.log("response Error", err)
        }
    }

    useEffect(()=>{
        fetchRiderProfileData()
    }, [])


    return { riderProfile, updateRiderProfileData }
}

export default useFetchRiderProfile
```

main.js



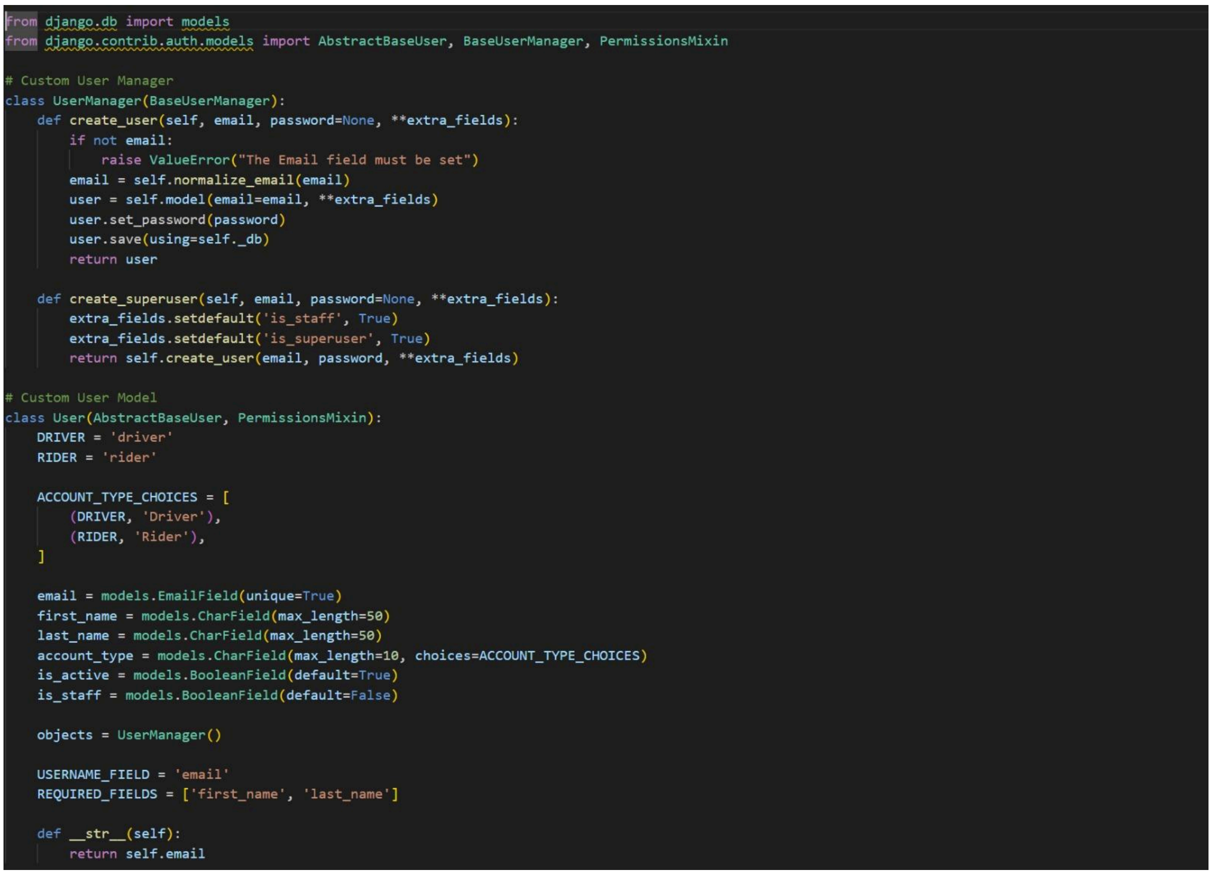
```javascript
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import 'mapbox-gl/dist/mapbox-gl.css';
import './index.css'
import { BrowserRouter } from 'react-router-dom'
import {ContextAppProvider} from '../context/useAppContext.jsx'


ReactDOM.createRoot(document.getElementById('root')).render(
  // <React.StrictMode>
    <ContextAppProvider>
    <BrowserRouter >
    <App />
    </BrowserRouter>
    </ContextAppProvider>

)
```

manage.py



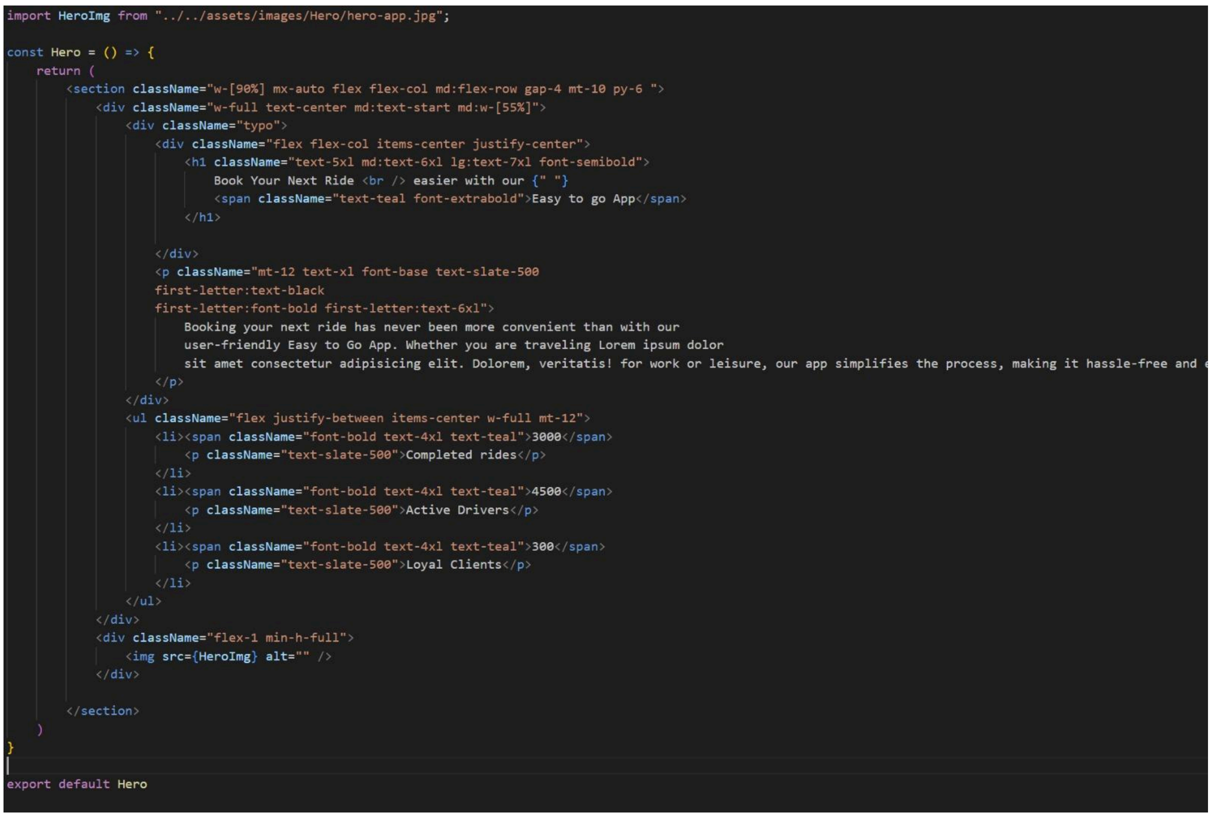
```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'taxi.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Code listing of our database creation class:



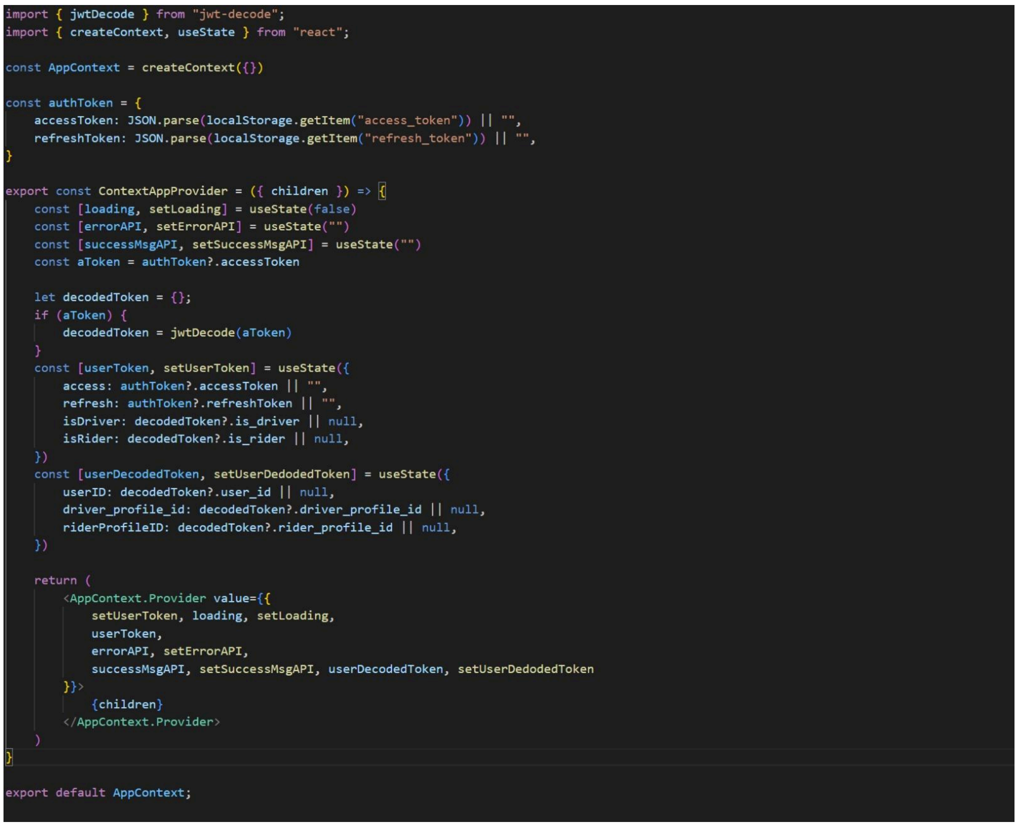
```python
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin

# Custom User Manager
class UserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError("The Email field must be set")
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        return self.create_user(email, password, **extra_fields)

# Custom User Model
class User(AbstractBaseUser, PermissionsMixin):
    DRIVER = 'driver'
    RIDER = 'rider'

    ACCOUNT_TYPE_CHOICES = [
        (DRIVER, 'Driver'),
        (RIDER, 'Rider'),
    ]

    email = models.EmailField(unique=True)
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    account_type = models.CharField(max_length=10, choices=ACCOUNT_TYPE_CHOICES)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['first_name', 'last_name']

    def __str__(self):
        return self.email
```

hero.js

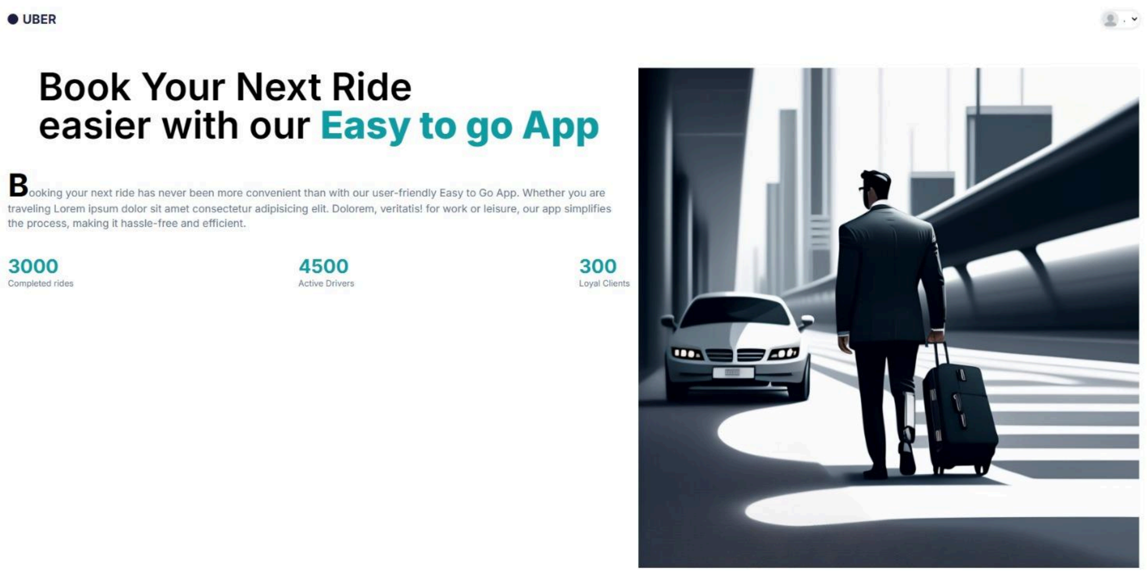


```
import HeroImg from "../../assets/images/Hero/hero-app.jpg";

const Hero = () => {
    return (
        <section className="w-[90%] mx-auto flex flex-col md:flex-row gap-4 mt-10 py-6 ">
            <div className="w-full text-center md:text-start md:w-[55%]">
                <div className="typo">
                    <div className="flex flex-col items-center justify-center">
                        <h1 className="text-5xl md:text-6xl lg:text-7xl font-semibold">
                            Book Your Next Ride <br /> easier with our {" "}
                            <span className="text-teal font-extrabold">Easy to go App</span>
                        </h1>

                    </div>
                    <p className="mt-12 text-xl font-base text-slate-500
                    first-letter:text-black
                    first-letter:font-bold first-letter:text-6xl">
                        Booking your next ride has never been more convenient than with our
                        user-friendly Easy to Go App. Whether you are traveling Lorem ipsum dolor
                        sit amet consectetur adipisicing elit. Dolorem, veritatis! for work or leisure, our app simplifies the process, making it hassle-free and
                    </p>
                </div>
                <ul className="flex justify-between items-center w-full mt-12">
                    <li><span className="font-bold text-4xl text-teal">3000</span>
                        <p className="text-slate-500">Completed rides</p>
                    </li>
                    <li><span className="font-bold text-4xl text-teal">4500</span>
                        <p className="text-slate-500">Active Drivers</p>
                    </li>
                    <li><span className="font-bold text-4xl text-teal">300</span>
                        <p className="text-slate-500">Loyal Clients</p>
                    </li>
                </ul>
            </div>
            <div className="flex-1 min-h-full">
                <img src={HeroImg} alt="" />
            </div>

        </section>
    )
}

export default Hero
```

useAppContext.js



```
import { jwtDecode } from "jwt-decode";
import { createContext, useState } from "react";

const AppContext = createContext({})

const authToken = {
    accessToken: JSON.parse(localStorage.getItem("access_token")) || "",
    refreshToken: JSON.parse(localStorage.getItem("refresh_token")) || "",
}

export const ContextAppProvider = ({ children }) => {
    const [loading, setLoading] = useState(false)
    const [errorAPI, setErrorAPI] = useState("")
    const [successMsgAPI, setSuccessMsgAPI] = useState("")
    const aToken = authToken?.accessToken

    let decodedToken = {};
    if (aToken) {
        decodedToken = jwtDecode(aToken)
    }
    const [userToken, setUserToken] = useState({
        access: authToken?.accessToken || "",
        refresh: authToken?.refreshToken || "",
        isDriver: decodedToken?.is_driver || null,
        isRider: decodedToken?.is_rider || null,
    })
    const [userDecodedToken, setUserDedodedToken] = useState({
        userID: decodedToken?.user_id || null,
        driver_profile_id: decodedToken?.driver_profile_id || null,
        riderProfileID: decodedToken?.rider_profile_id || null,
    })

    return (
        <AppContext.Provider value={{
            setUserToken, loading, setLoading,
            userToken,
            errorAPI, setErrorAPI,
            successMsgAPI, setSuccessMsgAPI, userDecodedToken, setUserDedodedToken
        }}>
            {children}
        </AppContext.Provider>
    )
}

export default AppContext;
```



● UBER

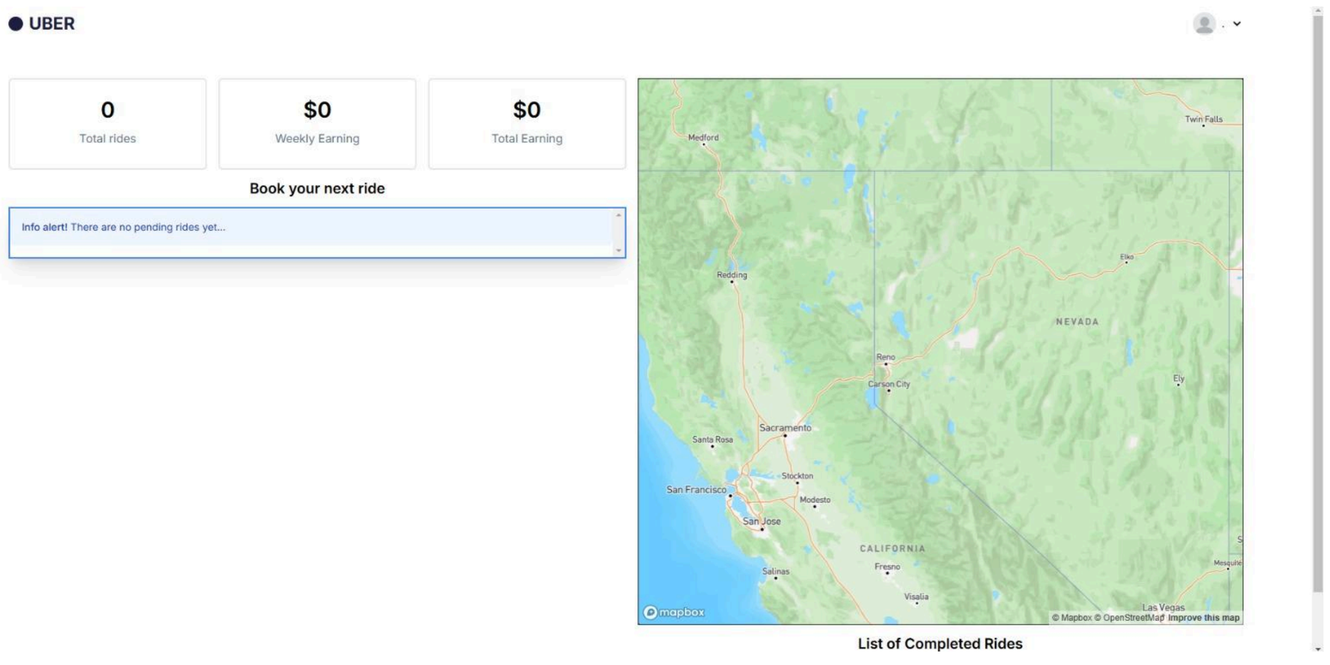
Book Your Next Ride
easier with our **Easy to go App**

Booking your next ride has never been more convenient than with our user-friendly Easy to Go App. Whether you are traveling Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolorem, veritatis! for work or leisure, our app simplifies the process, making it hassle-free and efficient.

3000
Completed rides

4500
Active Drivers

300
Loyal Clients



● UBER

0
Total rides

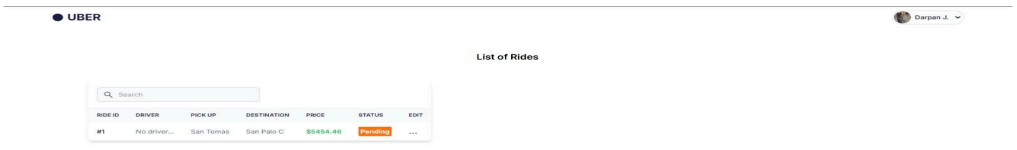
$0
Weekly Earning

$0
Total Earning

Book your next ride

Info alert! There are no pending rides yet...

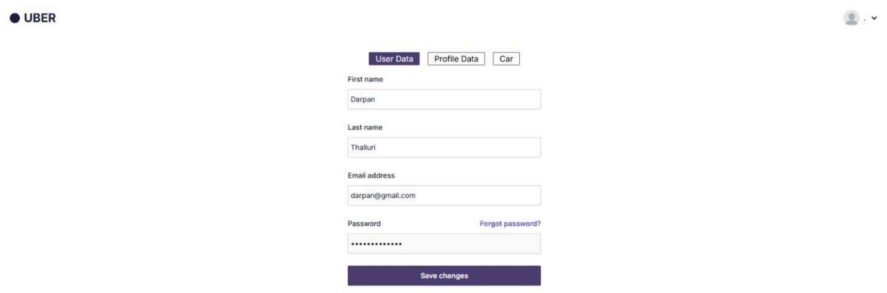
List of Completed Rides



List of Rides

Search

RIDE ID	DRIVER	PICK UP	DESTINATION	PRICE	STATUS	EDIT
#1	No driver...	San Tomas	San Palo C	$5454.46	Pending	...

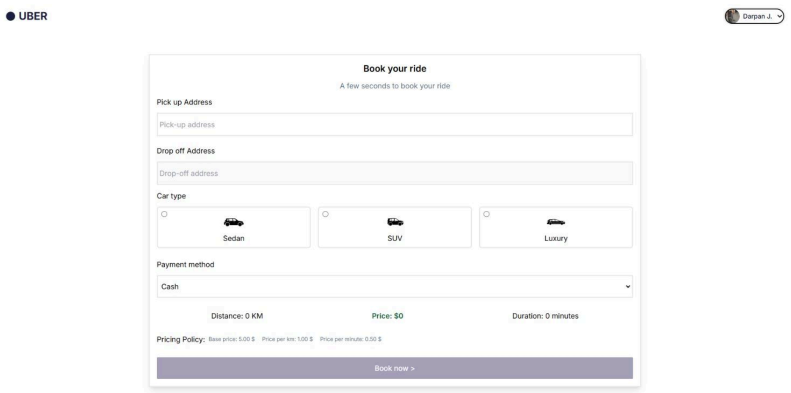


First name: Darpan

Last name: Thalluri

Email address: darpan@gmail.com

Password: •••••••••••••



Book your ride

A few seconds to book your ride

Pick up Address: Pick-up address

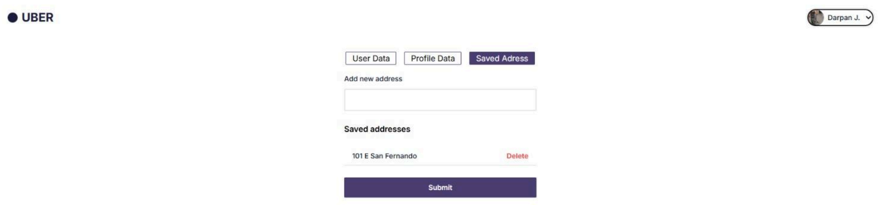
Drop off Address: Drop-off address

Car type: ○ Sedan ○ SUV ○ Luxury

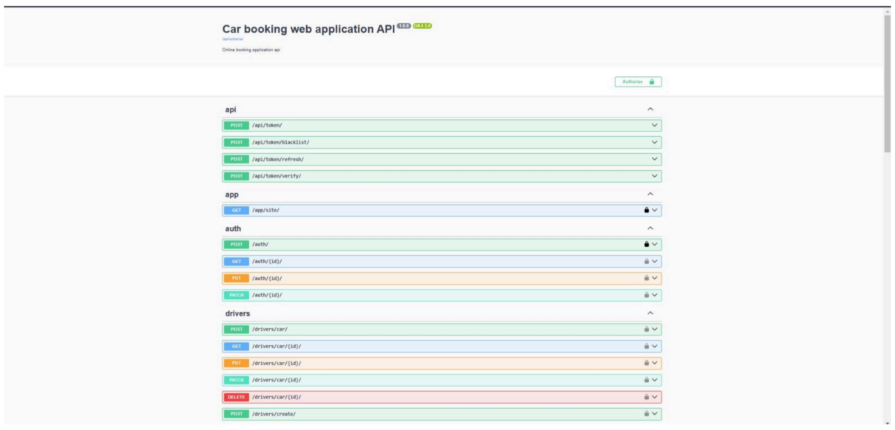
Payment method: Cash

Distance: 0 KM   Price: $0   Duration: 0 minutes

Pricing Policy: Base price: 5.00 $ Price per km: 1.00 $ Price per minute: 0.50 $



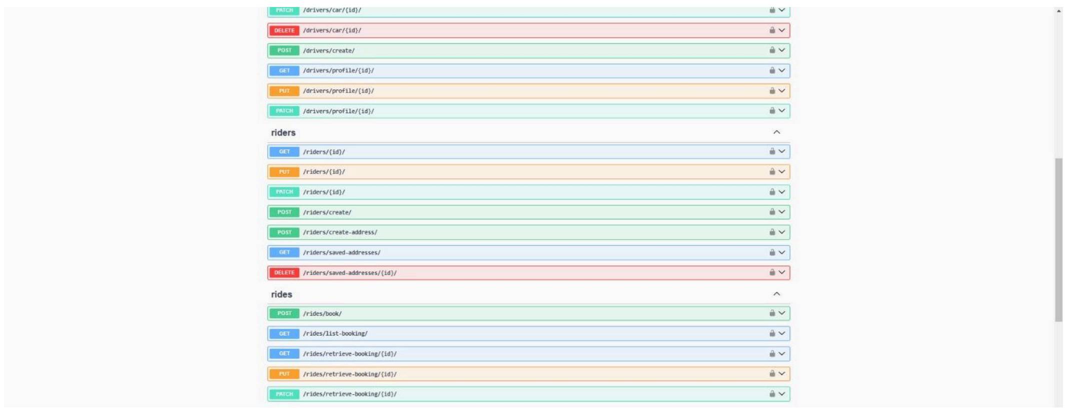
101 E San Fernando


Car booking web application API 1.0.0 OAS 3.0

/api/schema/

Online booking application api

```
api
POST    /api/token/
POST    /api/token/blacklist/
POST    /api/token/refresh/
POST    /api/token/verify/
app
GET     /app/site/
auth
POST    /auth/
GET     /auth/{id}/
PUT     /auth/{id}/
PATCH   /auth/{id}/
drivers
POST    /drivers/car/
GET     /drivers/car/{id}/
PUT     /drivers/car/{id}/
PATCH   /drivers/car/{id}/
DELETE  /drivers/car/{id}/
POST    /drivers/create/
```


```
PATCH   /drivers/car/{id}/
DELETE  /drivers/car/{id}/
POST    /drivers/create/
GET     /drivers/profile/{id}/
PUT     /drivers/profile/{id}/
PATCH   /drivers/profile/{id}/
riders
GET     /riders/{id}/
PUT     /riders/{id}/
PATCH   /riders/{id}/
POST    /riders/create/
POST    /riders/create-address/
GET     /riders/saved-addresses/
DELETE  /riders/saved-addresses/{id}/
rides
POST    /rides/book/
GET     /rides/list-booking/
GET     /rides/retrieve-booking/{id}/
PUT     /rides/retrieve-booking/{id}/
PATCH   /rides/retrieve-booking/{id}/
```

<u>Observations and Lesson learned:</u>

Through the development of this project, several key observations and lessons were learned. The integration of diverse technologies like Django, ReactJS, MySQL, MongoDB, Kafka, and AWS highlighted the importance of modularity and clear communication between components. Implementing real-time features with Kafka and Redis taught us the value of efficient resource management and caching to optimize performance. The use of a hybrid database model demonstrated how relational and non-relational databases can complement each other when handling different data types. Additionally, deploying the system on AWS with Docker and Kubernetes emphasized the need for scalability and fault tolerance in modern applications. Overall, the project reinforced the importance of planning, collaboration, and testing to build a robust, scalable, and efficient system.

Github Link

https://github.com/saiprasadthalluri/DATA236_Group4_project.git