Project Topic:

Taxi Management System

PROJECT REPORT

Table of Contents:

PROBLEM STATEMENT:

For various purposes people need to travel on the daily basis. And there are various mediums used for travelling. But getting the transportation medium at peoples convenience is a difficult thing to achieve. So, our problem statement is to build a web application for Taxi Management System which provide the easy platform for taxi booking.

OBJECTIVES:

1. To create a location based reactive system where customers can book taxis that are in the proximity to their location and can plan a trip to new location.

2. To save time of both the customers and drivers during transit.

3. To provide excellent customer service, with instant response and easy payment gateway.

MOTIVATION:

Transportation is the basic and one of the core needs of working people in our society. There are many public transports available although people avoid to travel from them due to rush, crowd, traffic congestion, unavailability of seats, lack of maintenance, delay, etc. And due to this pandemic situation lot of people used to plan their journey via taxis to avoid contact with infected people. Also, we can find lot of people searching for a taxi in their locality as well as same case with taxi drivers, searching for customers. This motivated us, to develop an efficient and easy to use taxi management system platform which will provide solutions to all problems.

SUMMARY OF SRS:

The SRS stands for Software Requirements Specification. It includes the introduction of system like purpose, product scope, document convections, references. It also gives overall description, product perspective, operating environment, design and implementation constraints, assumptions, and dependencies. The User Interfaces, hardware interfaces, software interfaces, communication interfaces are included. Also, system features and other non-functional requirements are listed.

- **Introduction to SRS**:

  The purpose of SRS is to describe the features of Taxi Management System (TMS), so as to serve as a guide to developers and act as a software validation document for clients. It is intended to completely specify the functionalities which system provides. It describes the product scope in the form of functionalities provided by TMS like, register, sign in, send and receive booking requests, give ratings and feedbacks, easier payment gateway, etc.

- **Overall Description:**

  The product perspective is to build a platform where people can book taxi to plan their trip based on their locations and to create responsive, user-friendly and easy to use interface for application which will be available for 24 X 7. The product functions include taxi booking, viewing trip history, making payment, submitting ratings and feedbacks, etc. The user classes for the system are customers and drivers with basic knowledge of the system. The operating environment includes minimum 4 GB RAM, 120 GB free hard drive space and compatible processor.

- **External Interface Requirements:**

  The user interfaces encompasses, GUI which is build using React frontend framework and JSX model. This UI will be responsive to size of screen, ensure user get proper messages informing them about the status of task. The hardware interfaces should support TCP/IP protocols. The software interfaces require MySql, MERN stack, compatible processor. The communicating interfaces includes support of HTTP protocol for querying, NodeJS for communication with database.

- **System Features:**
  1. Register/ Sign in functionality
  2. Sending and Receiving trip requests.
  3. View booking status and trip history
  4. Payment gateway with ratings and grievances.
  5. Checking shifts and taxi details.
  6. Updating location and choice to accept trip requests
  7. Maintaining taxi's owner information and garage status.
  8. Creating driver accounts and viewing feedbacks.

- **Other Non-functional Requirements:**

These requirements include reliability, portability, compatibility, proper and functional website, secure payment gateway, alerts based on activities with respect to system.

ENTITY RELATIONSHIP DIAGRAM:

ER diagrams are composed of three main elements: entities, attributes, and relationships. Entities - typically displayed in a rectangle. Attributes - displayed in a circle or an oval, the attributes refer to the characteristics of an entity. Relationships - illustrate how two or more entities interact with each other.

ER Diagram
TAXI MANAGEMENT SYSTEM
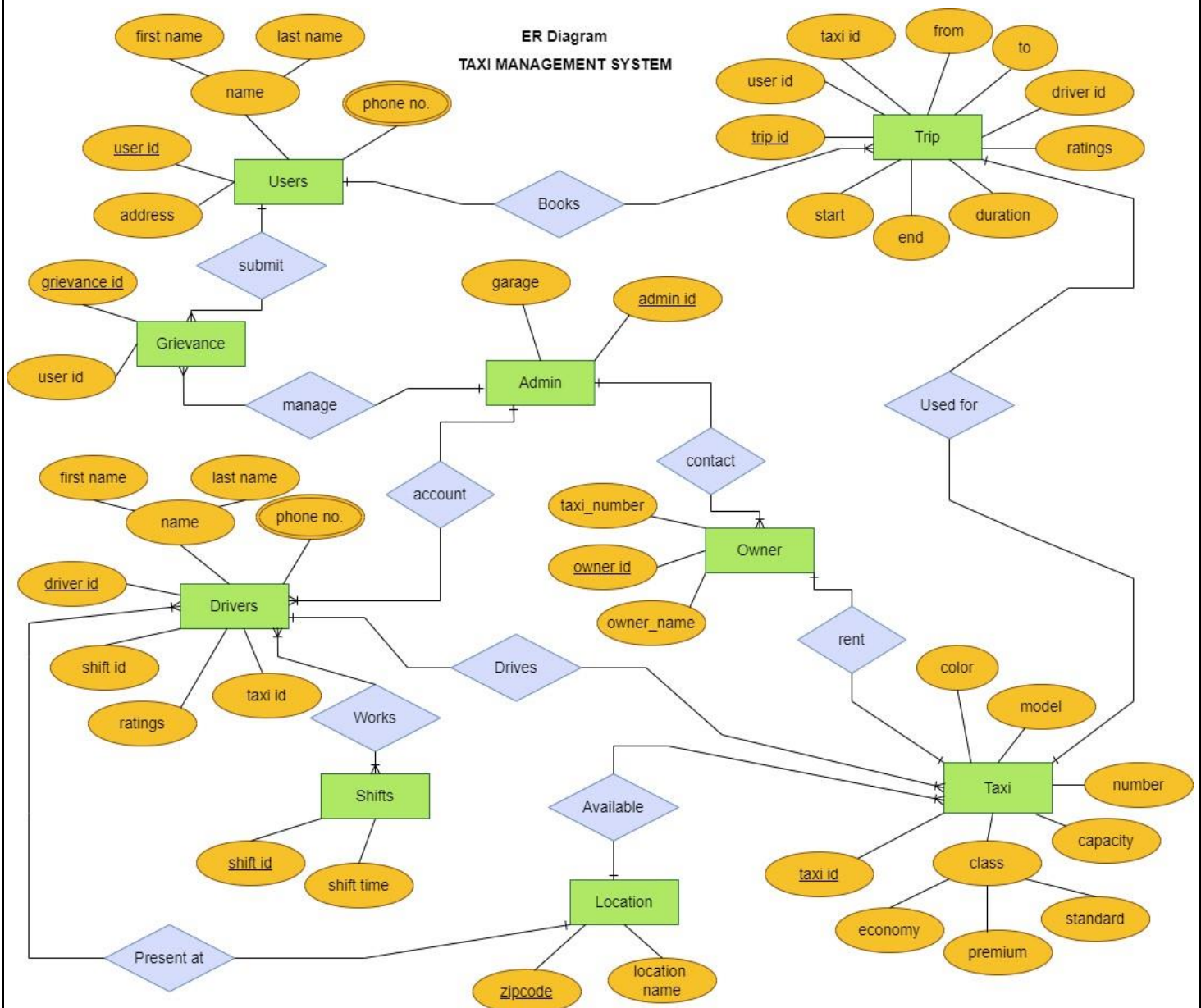
UML AND EXPLAINATION:

1. Use Case Diagram:

It is used to represent dynamic behaviour of System. It encapsulates the system's functionality by incorporating use cases, actors, and their relationship. It depicts external view of the system and gathers the system needs.

System's boundaries represented using a rectangle, use cases using ovals. Actors are the users of a system. An "include" relationship indicates that one use case is needed by another in order to perform a task. An "extend" relationship indicates alternative options under a certain use case.

Use Case Diagram
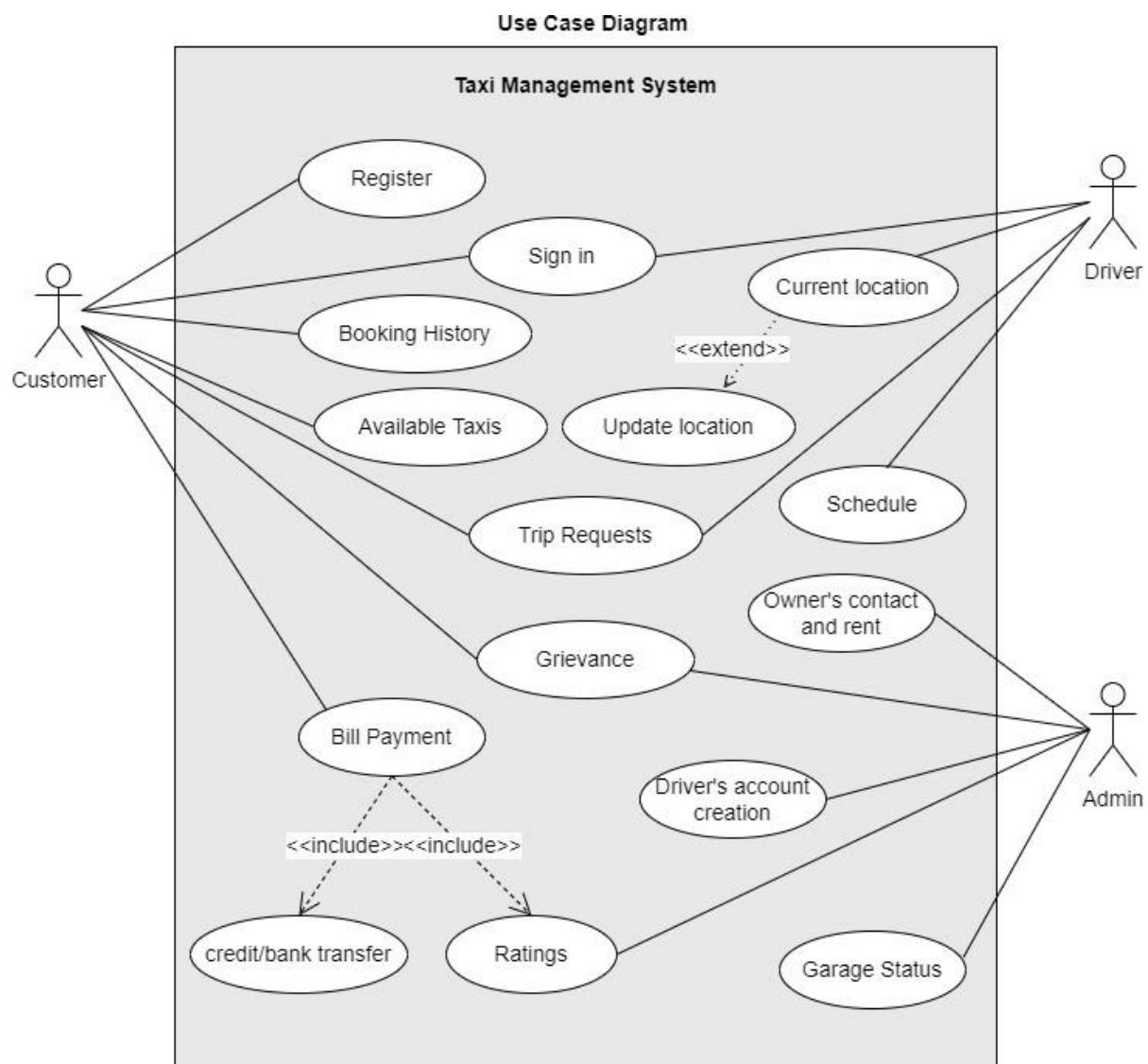
2. **Class Diagram**:

The class diagram depicts a static view of an application. A class consist of its objects and also may inherit from other classes. It Analyse and design static view of an application and also describes the major responsibilities of system.

A class represent a concept which encapsulates state(attributes) and behavior (operations). Each attribute has a type. Each operation has a signature. The class name is the only mandatory information.

Class Diagram
TAXI MANAGEMENT SYSTEM
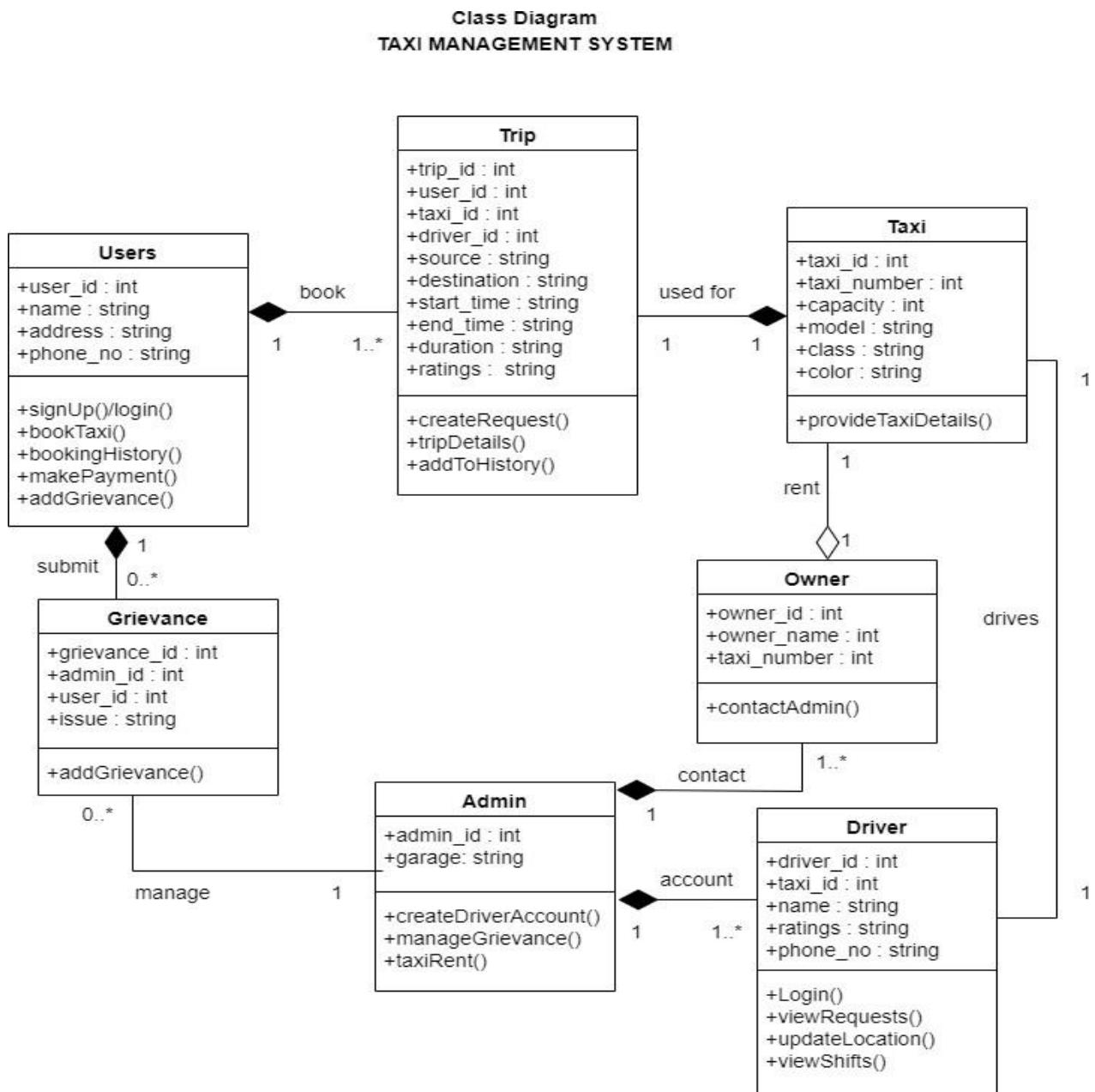
3. **Sequence Diagram**:

   The sequence diagram represents the flow of messages in the system. The purpose of Sequence diagram is to model high level interaction among active objects within a system and to model interaction among objects inside a collaboration realizing the use case.

   A sequence diagram is structured in such a way that it represents a timeline which begins at the top and descends gradually to mark the sequence of interactions. Each object has a column and the messages exchanged between them are represented by arrows.

Sequence Diagram

TAXI MANAGEMENT SYSTEM
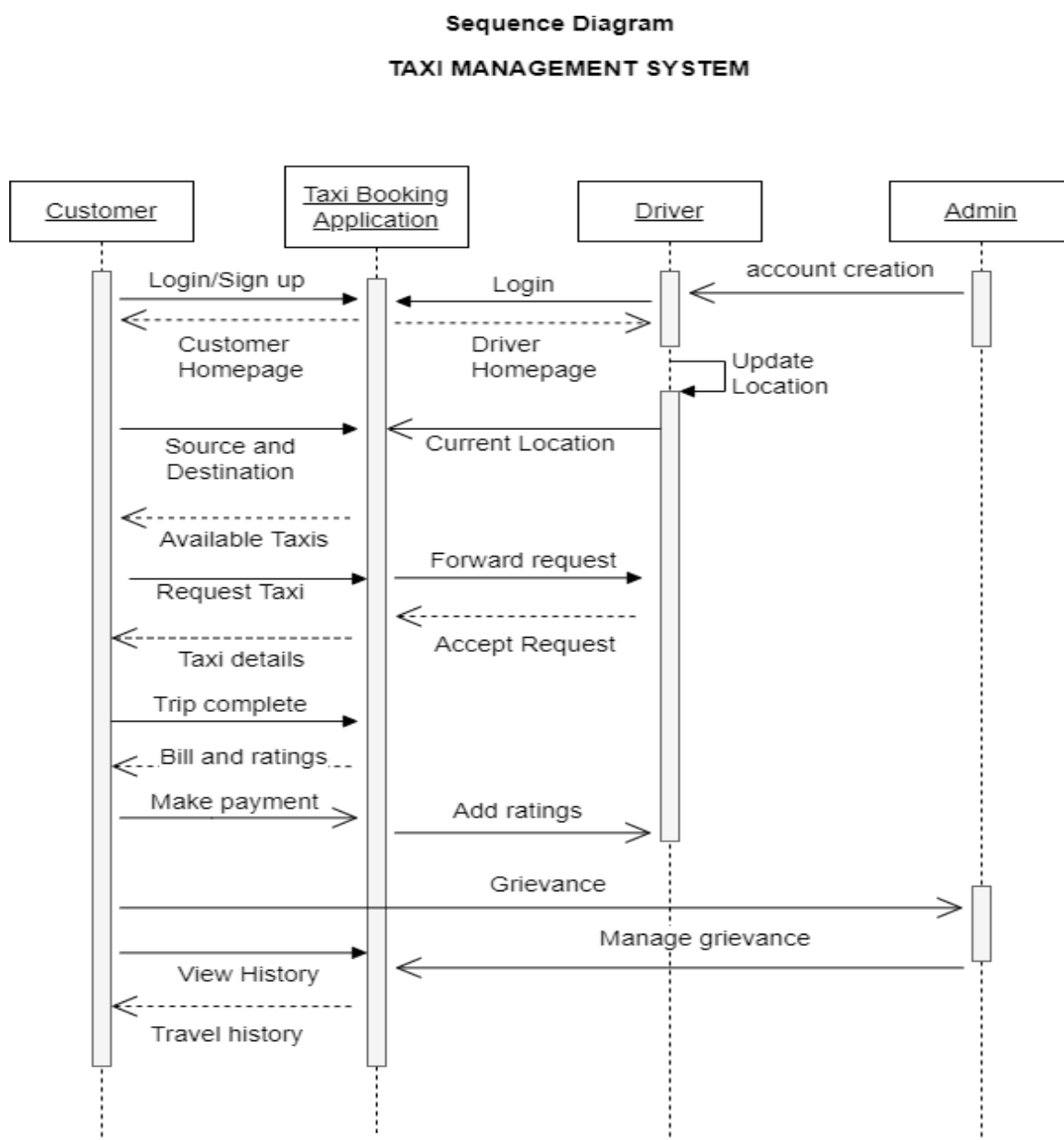
4. **State Diagram**:

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It captures the software system's behaviour. It models the behaviour of a class, a subsystem, a package, and a complete system.

Its initial state usually marked by a solid black circle, middle states are portrayed as boxes with rounded corners, transitions are external straight lines each with an arrow at one end connect various pairs of boxes and final state is portrayed as a large black dot with a circle around it.

State Diagram

TAXI MANAGEMENT SYSTEM

Homepage

Login

customer | driver

Customer Login/Sign Up

Driver Login

Request Taxi

Update location

view history

current location

select Taxi

other details

book taxi

view requests

end trip

taxi and shift details

payment

accept requests

Log out

Log out
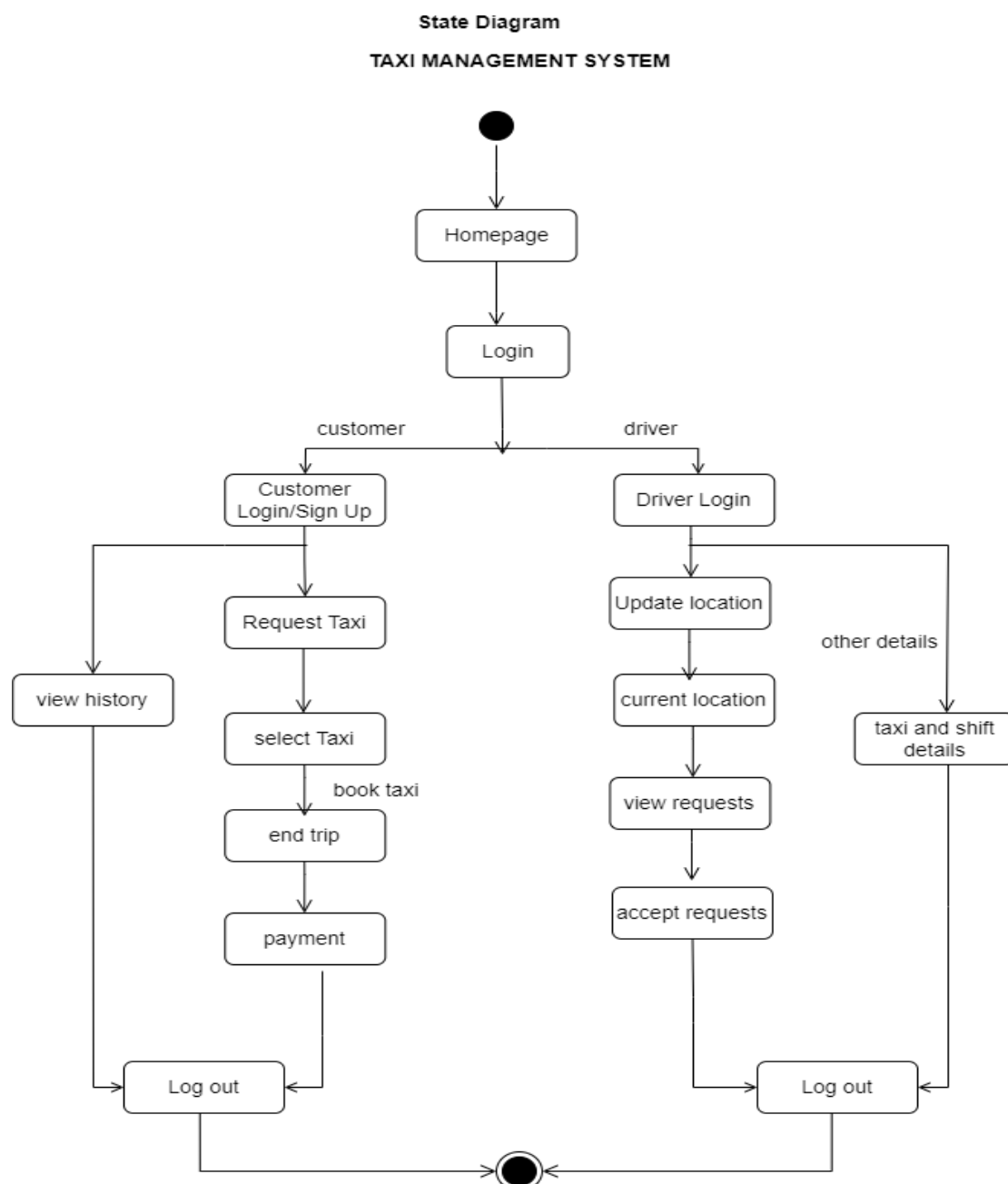
5. **Activity Diagram**:

The Activity diagram is used to demonstrate the flow of control within system. It models concurrent and sequential activities. It helps in envisioning the workflow from one activity to another.

It consists of action, decision node, control flows, start and end node as a components.

Activity Diagram

TAXI MANAGEMENT SYSTEM
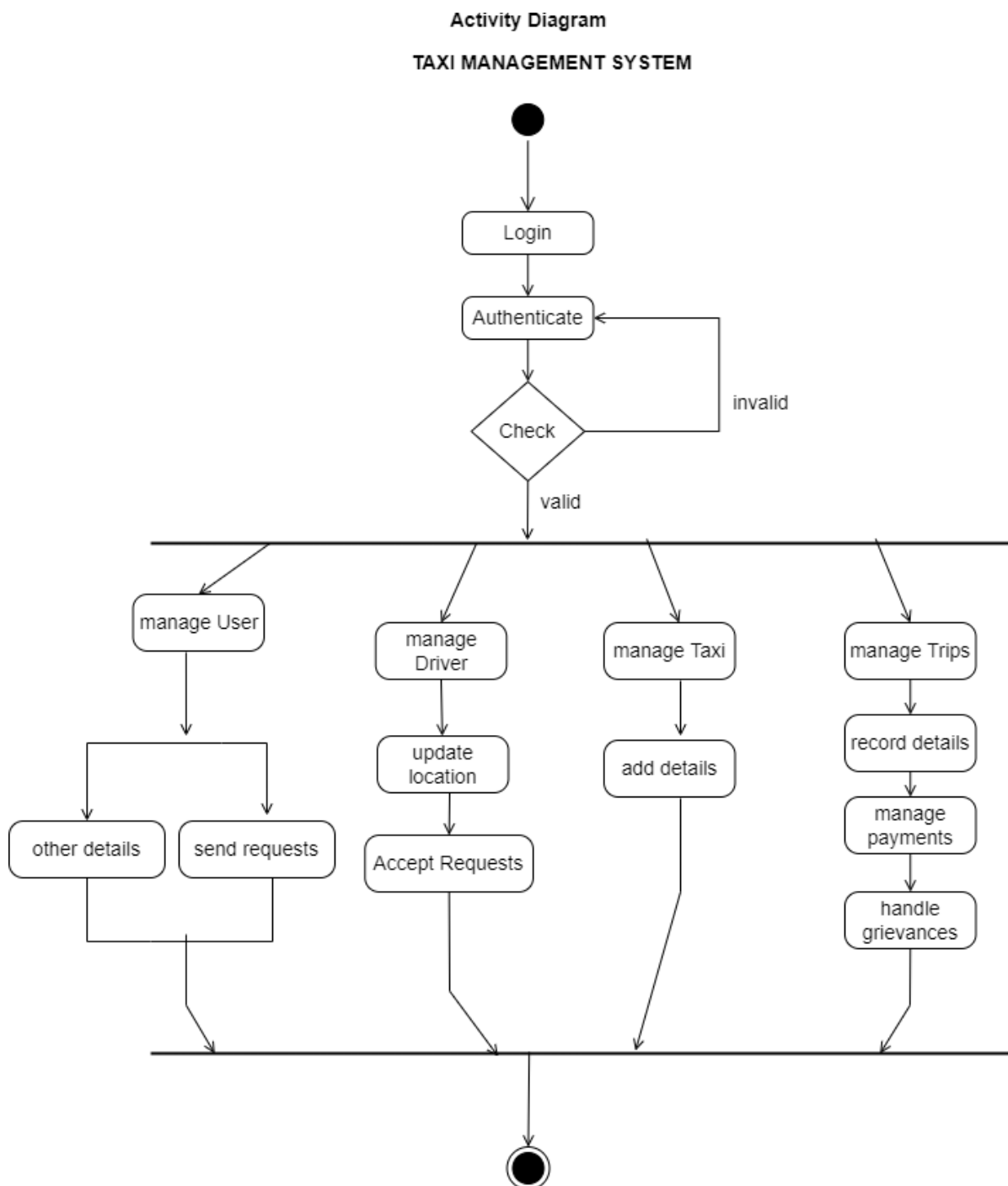
6. **Component Diagram**:

The component diagram is used to break down a large object-oriented system into the smaller components. It models the physical view of system. The purpose of Component diagram is to explore each component of system and to depicts the relationship and organization of components.

Its component symbol represent an entity required to execute a function. Node symbol represents hardware or software objects. Port symbol specifies a separate interaction point between the component and the environment. Dependency symbol shows that one part of your system depends on another.

Component Diagram

TAXI MANAGEMENT SYSTEM
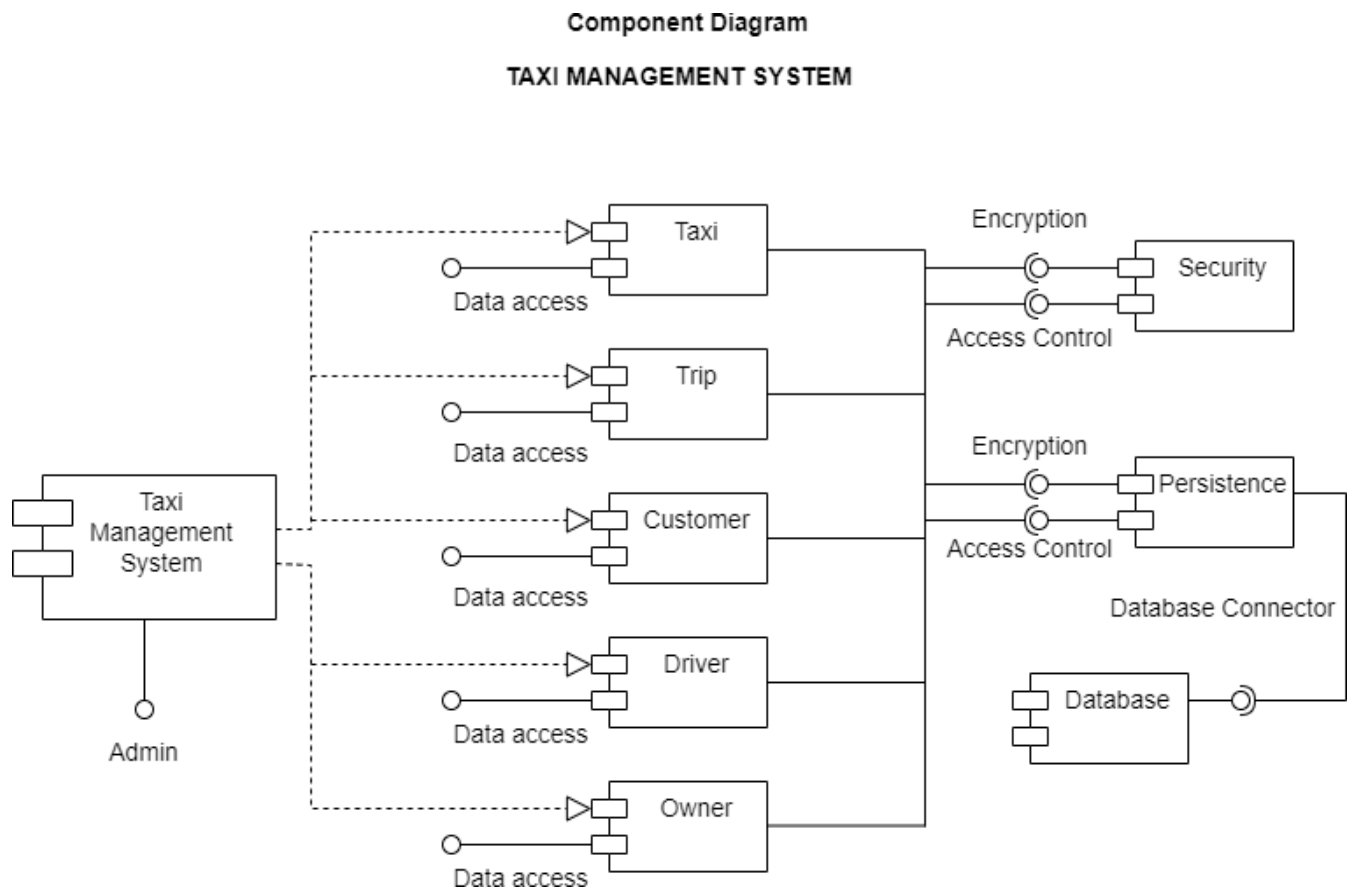
IMPLEMENTATION:

1. Home Page:

2. Admin Login:

3. Customer Login:

4. Driver Login:


```javascript
import React, { Component } from 'react'
import DriverPage from './driverPage'
import { CardHeader, CardFooter, CardBody, } from 'reactstrap';
import {Input} from 'baseui/input'
import { Card} from 'baseui/card'
import { Button} from 'baseui/button'
import background from './images/driverbg.gif'
import 'react-notifications-component/dist/theme.css'
import ReactNotification, { store } from 'react-notifications-component';

import axios from 'axios'
export default class Driver extends Component {
    constructor(props){
        super(props)
        this.state = {
            name : '',
            pass : '',
            login : false,
        }
    }
    handlename = (e) => {
        this.setState({name : e.target.value})
    }

    handlepass = (e) => {
        this.setState({password : e.target.value})
    }
    login = () => {
        axios({
            method : 'post',
            url : 'http://localhost:5000/logindriver',
            data : {
                name : this.state.name,
                pass : this.state.password
            }
        }).then((res) => {
            if(res.status === 200 && res.data.length !== 0){
                const data = res.data.data[0]
                this.setState({
```
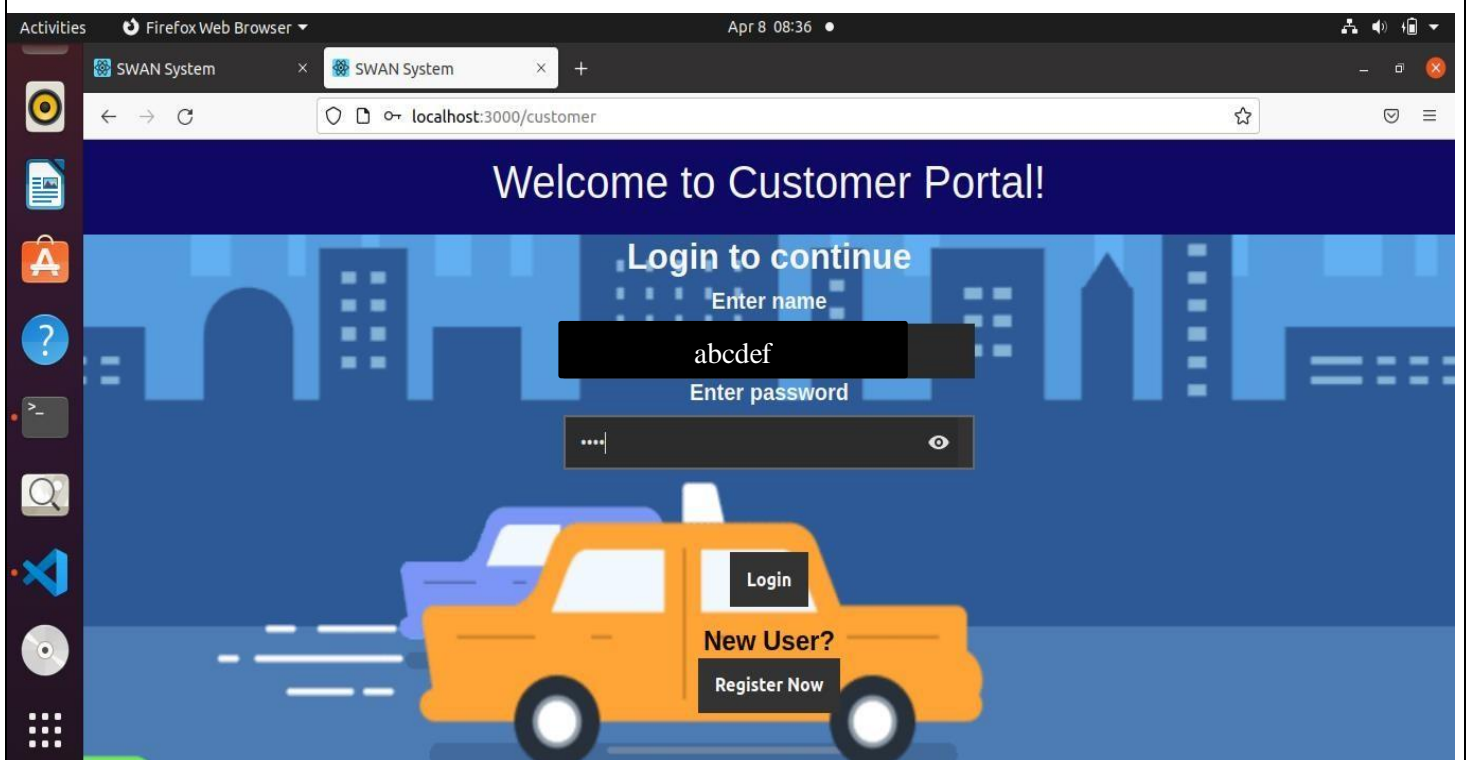

Hello, Driver!

Login to continue

Enter name:

abcdef

Enter password:

••••••

Login
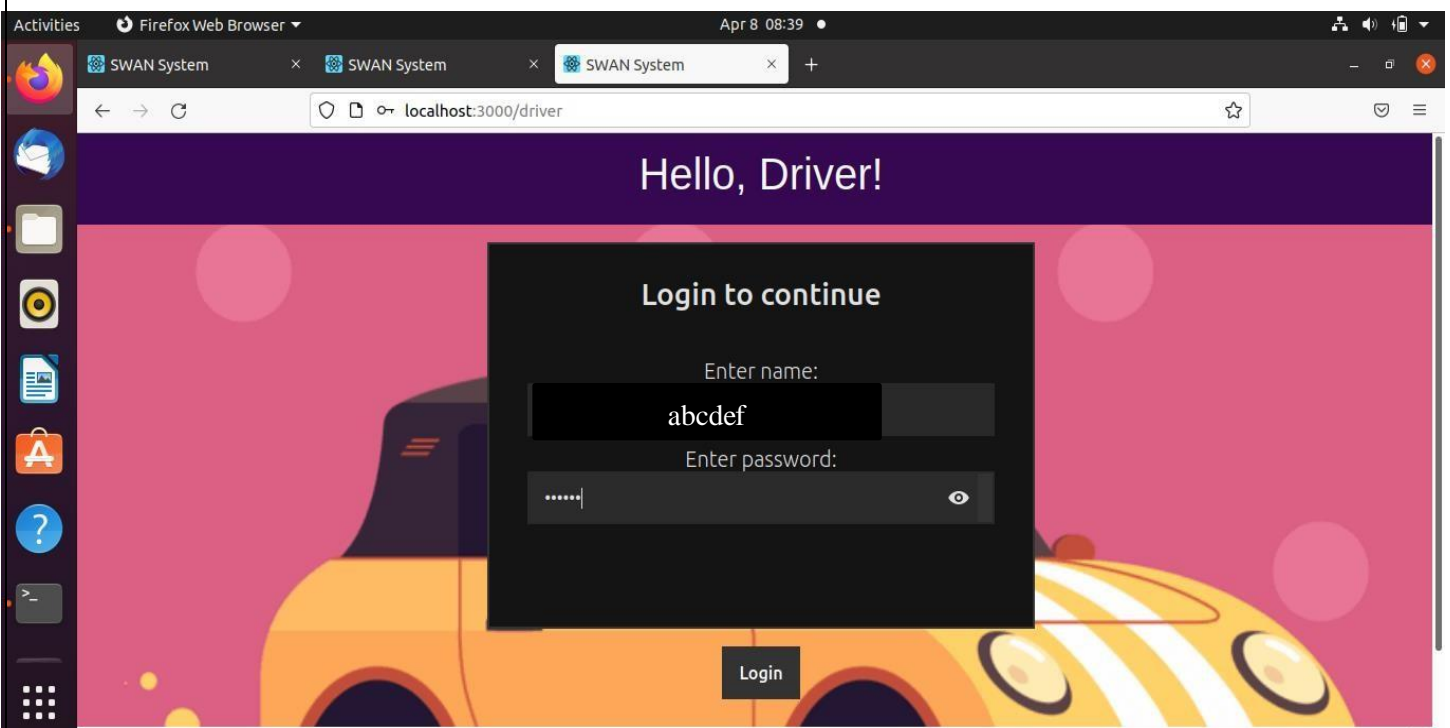
5. Admin Dashboard:


```
Activities          Visual Studio Code ▼                                        Apr 8 08:25  ●

                                                        adminPage.js - swan - Visual Studio Code

File  Edit  Selection  View  Go  Run  Terminal  Help

JS adminPage.js M ×

src > pages > JS adminPage.js > Admin > getgrievance > then() callback
  1   import React, { Component } from 'react'
  2   import { Card, Jumbotron, Spinner } from 'reactstrap'
  3   import {Button } from 'baseui/button'
  4   import {Input} from 'baseui/input'
  5   import { Modal, ModalBody, ModalFooter} from 'baseui/modal'
  6   import Header from './headerFile'
  7   import axios from 'axios'
  8   import ReactNotification, { store } from 'react-notifications-component';
  9   import 'react-notifications-component/dist/theme.css'
 10   import { ListItem, ListItemLabel } from 'baseui/list'
 11
 12   export default class Admin extends Component {
 13       constructor(props){
 14           super(props)
 15           this.state = {
 16               open : false,
 17               loading : false
 18           }
 19       }
 20       getgarage = () => {
 21           axios({
 22               method : 'get',
 23               url : 'http://localhost:5000/api/getgarage'
 24           }).then(res => {
 25               console.log(res.data)
 26               this.setState({
 27                   garage : res.data.garage
 28               })
 29           }).catch(e => {
 30               console.log(e)
 31           })
 32       }
 33
 34       getgrievance = () => {
 35           axios({
 36               method : 'get',
 37               url : 'http://localhost:5000/api/getgrievance'
 38           }).then(res => {
 39               console.log(res.data)
```
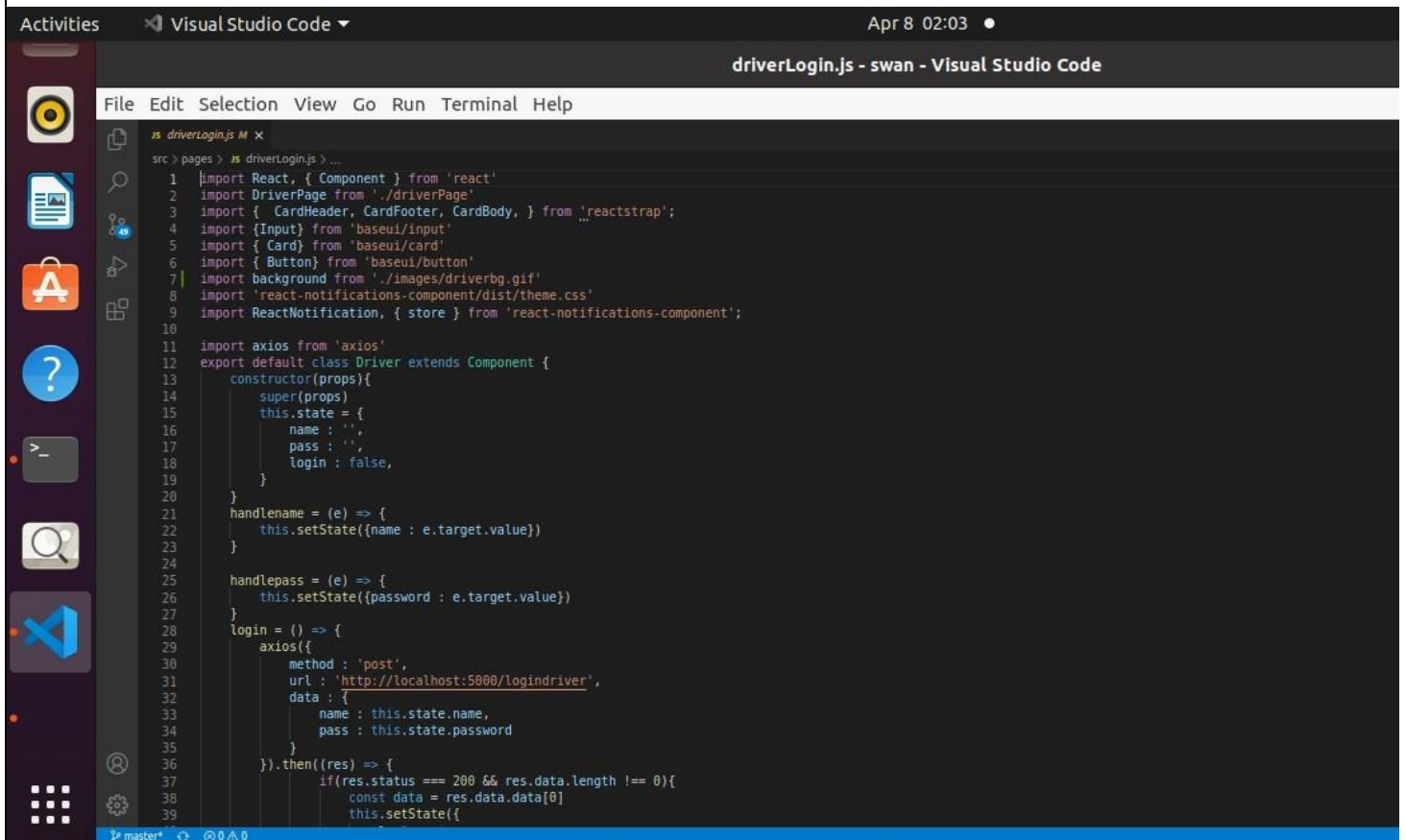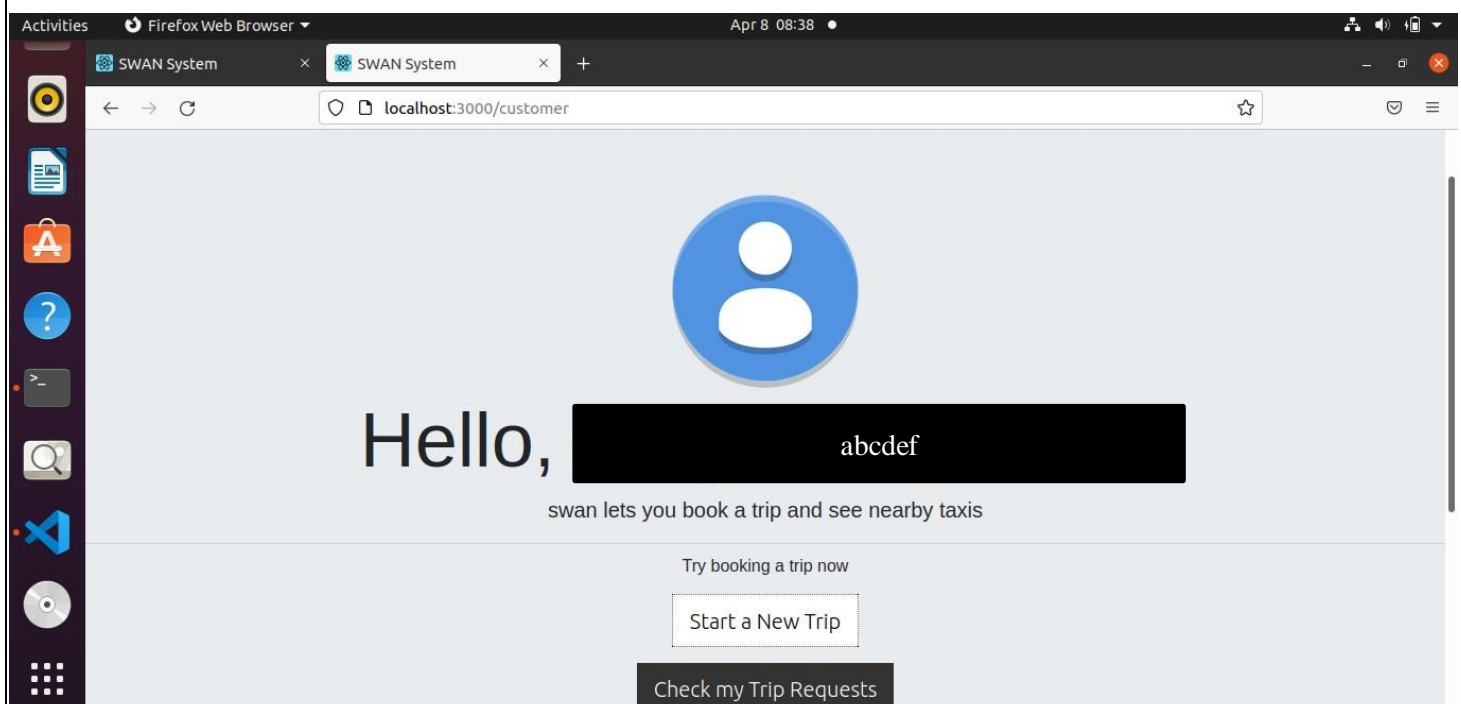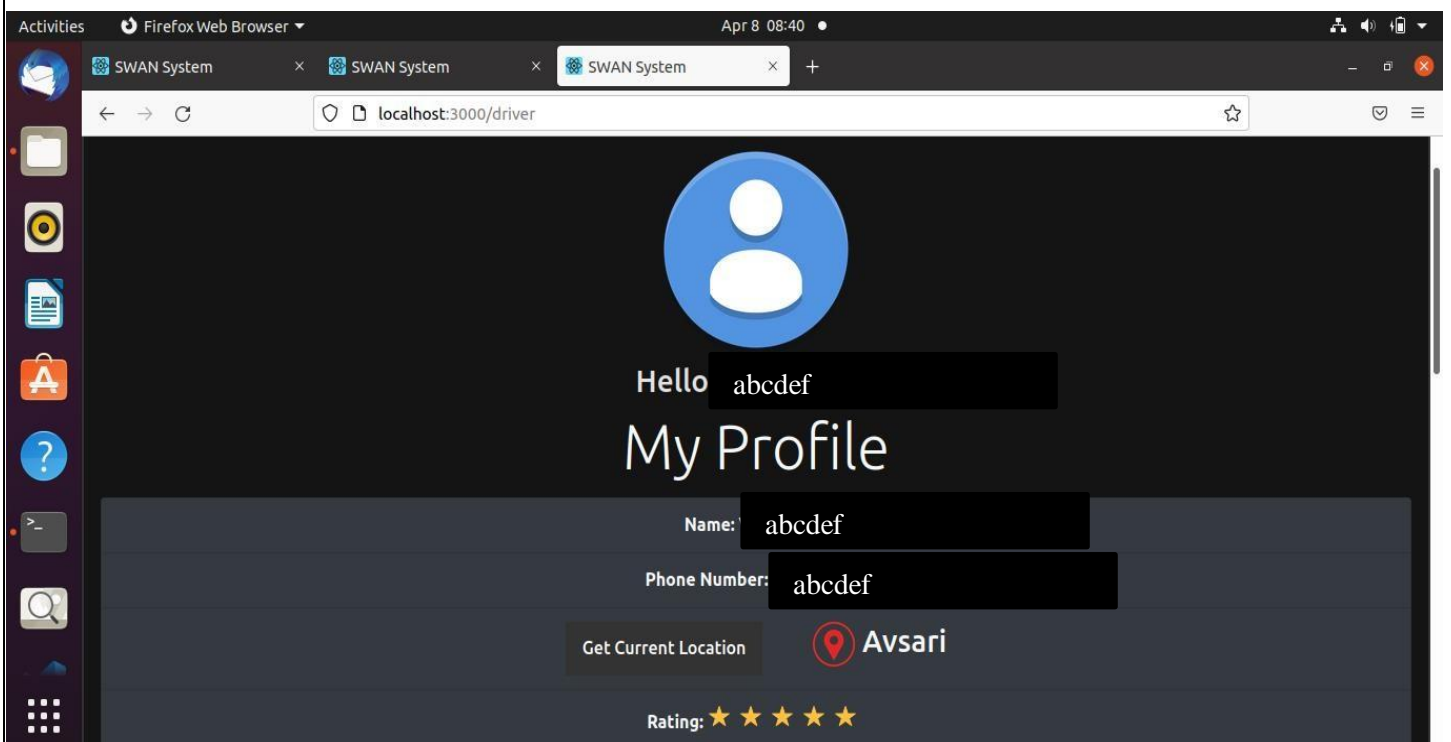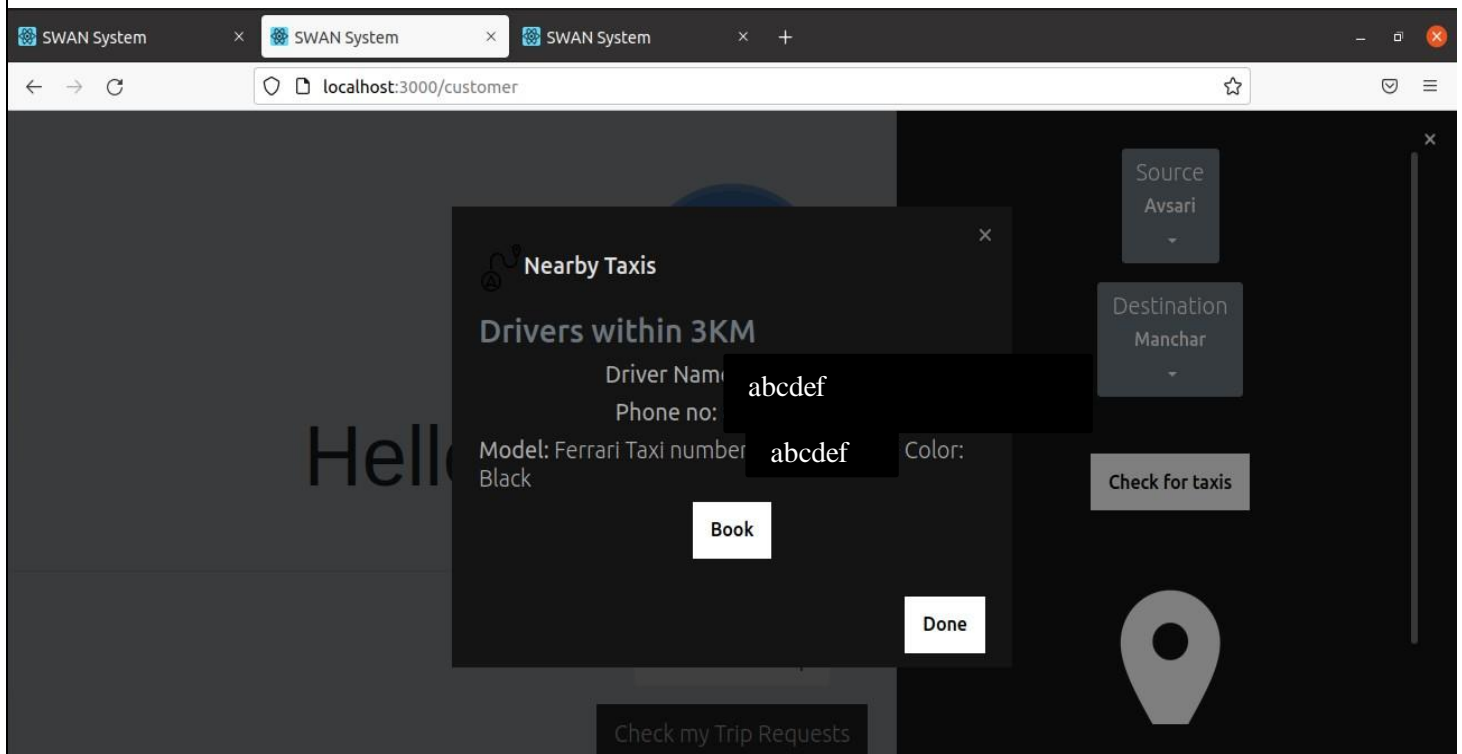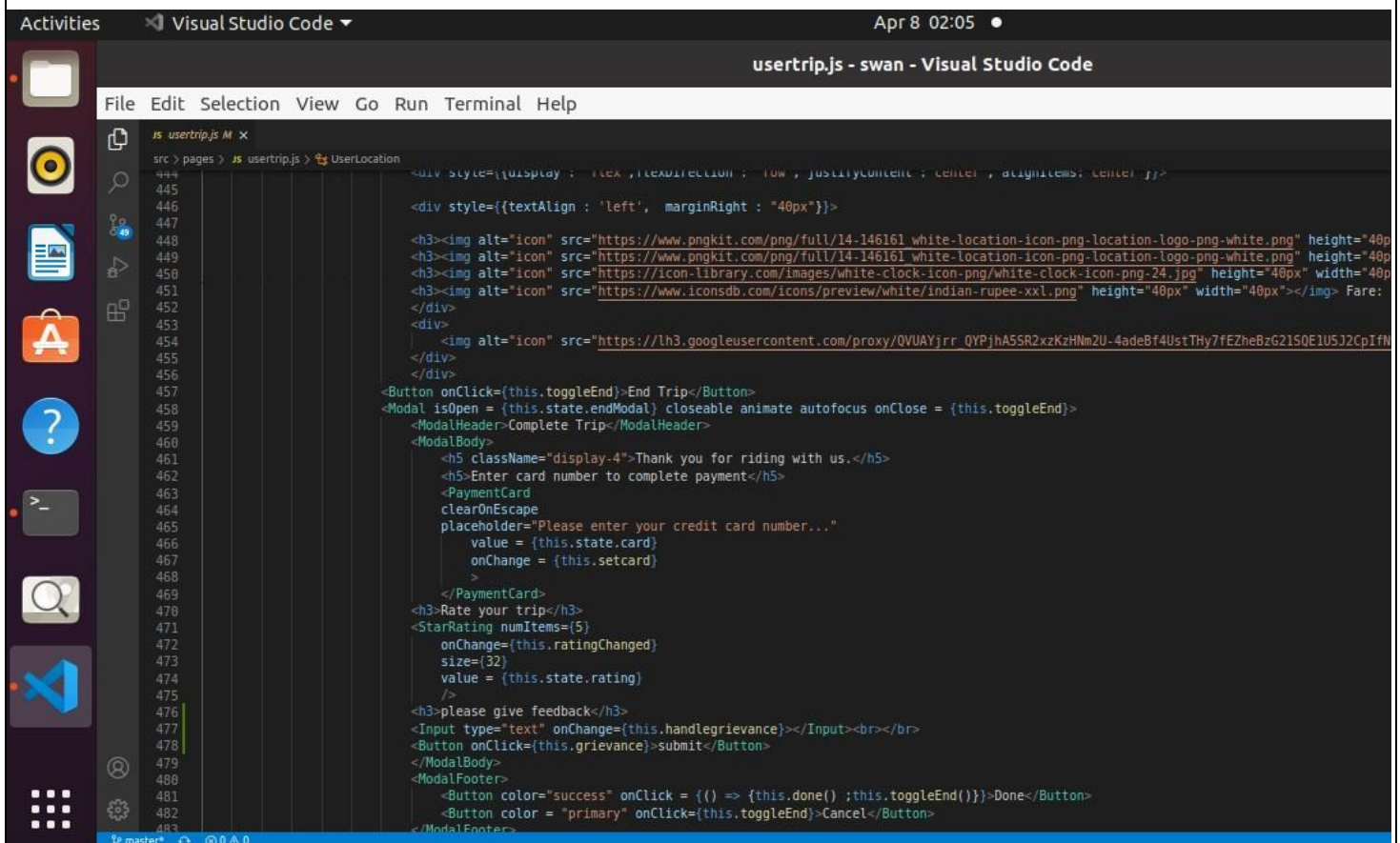

Hello Admin abcdef

Add driver

Garage status

Get Garage Info

Get Owner Info

view grievances

6. Customer Dashboard:

7. Driver Dashboard:

8. Choose Locations and Book Taxi:

9. Payment, Ratings and Feedback:

TESTING:

Security testing is a non-functional software testing technique used to determine if the information and data in a system is protected. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses.

In our project we used automated security testing plugin "HCL AppScan CodeSweep" for Static Application Security Testing (SAST) functionality for detecting vulnerabilities early in the development life cycle.

CONCLUSION:

To conclude, this is an efficient Taxi Management System application which enables its users to book the taxis which are present in the proximity to their location. Along with that, it provides its users with the complete details of driver and taxi. User's views about the system are also taken into consideration to improve the system in future. It also facilitates the drivers working with the system by providing easier ride requests and giving them freedom to choose their passenger. The admin can maintain the whole system, by recording the garage status, owner's information. Admin can add new driver to the system and can expand the network of whole system. So, this application can be used as location based reactive system for taxi booking.

FUTURE SCOPE:

This project can be extended further by adding several functionalities to it. Currently, this application provides the list of places to book taxi, instead of that Google maps functionality can be added to it in future. The feedback part of the application will be made optional for users. Further, we will add rent functionality which will help admin to pay the rent to respective taxi owner. Also, we will allow users and drivers to upload their profile photo so that they can easily recognize each other. This application will be improved further, to secure the user and driver information.

GITHUB LINK:

https://github.com/vishalanap/Taxi_Management_System

REFERENCES:

1. Taxi management system approach- Taxi Management Software Solutions – Taxi Management System - Taxi Fleet Management (aistechnolabs.com)
2. Connecting mysql server using nodejs - https://www.tutorialkart.com/nodejs/nodejs-mysqlconnect-to-database
3. Research paper - https://www.researchgate.net/publication/348304336_Design_and_Implementation_of_Taxi_Management_System