LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS6P05NI Final Year Project

Assessment Weightage & Type

40% FYP Final Report

Semester

2020/21 Autumn

Student Name: Meru Sangroula

London Met ID: 18029653

College ID: NP01CP4A180038

Internal Supervisor – Sukrit Shakya

External Supervisor – Madhukar Shrestha

Assignment Submission Date: April 26, 2021

Assignment Due Date: April 26, 2021

Word Count (Where Required): 9010

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Abstract

Technology has evolved in such a fast speed that all the previous old technique and method are now replaced by newly management system. Previously, all the business records and transactions were done in papers. Slowly technology became more advance and software like excel were used for business records and transactions. It is very hard to manually record all the details and is the records are huge then its quite next to impossible. Making a software which can maintain all the records no matter how big the data makes business much easier and makes maintaining details much smoother. My application will be for taxi business where the records can be easily handled which will consume very less time.

Talking about evolution of technology, nowadays AI (Artificial Intelligence) technology is helping humans a lot in saving life. From Train to rocket, AI is everywhere. My project will also have an AI feature called Drowsiness Detection Software which detects the eye of the driver while driving and if the driver sleeps (gets drowsy), then the alarm starts to wake the driver and prevent driver from accidents.

Making a project needs proper research, suitable methodology and focus. This project follows RUP (Rational Unified Process) Methodology along with programming languages, frameworks and AI scripts.

Table of Contents

CHAPTER 1: INTRODUCTION 1
1.1 PROJECT DESCRIPTION 1
1.2 CURRENT SCENARIO 2
1.3 PROBLEM DOMAIN AND PROJECT AS A SOLUTION 2
1.4 AIM AND OBJECTIVES 3
1.5 STRUCTURE OF THE REPORT 3
1.5.1 BACKGROUND 3
1.5.2 DEVELOPMENT 3
1.5.3 TESTING AND ANALYSIS 3
1.5.4 CONCLUSION 4
CHAPTER 2: BACKGROUND 4
2.1 ABOUT THE END USERS 4
2.2 UNDERSTANDING THE SOLUTION 4
2.2.1 Overview of the System 4
2.2.2 Building Process 5
2.3 SIMILAR PROJECTS 8
2.3.1 JUGNOO TAXI 8
2.3.2 Yelowsoft 10
2.3.3 Taxi Management System by AudenBerg Technologies India 11
2.3.4 Drowsiness Detector by Pyimagesearch 12
2.4 COMPARISIONS 14
CHAPTER 3: DEVELOPMENT 15
3.1 CONSIDERED METHODOLOGIES 15
3.2 SELECTED METHODOLOGY 19

3.3 SURVEY RESULTS 21
3.3.1 PRE-SURVEY RESULTS 21
3.3.2 POST-SURVEY RESULTS 25
3.4 REQUIREMENT ANALYSIS 27
3.4.1 Web Application 27
3.4.2 Mobile Application 28
3.4.3 Drowsiness Detection Software 29
3.5 DESIGN 29
3.5.1 Entity Relationship Diagram 30
3.5.2 Use Case Diagram 31
3.5.3 Context Level Diagram 32
3.5.4 Data Flow Diagram 33
3.5.5 Sequence Diagram 34
3.5.6 Communication Diagram 36
3.7 IMPLEMENTATION 38
3.7.1 Iteration 1: User Registration 38
3.7.2 Iteration 2: Taxi Registration 47
3.7.3 Iteration 3: Drowsiness Software 54
3.7.4 Iteration 4: Income 58
3.7.5 Iteration 5: Login Authentication 67
3.7.6 Iteration 6: Drowsiness API 71
3.7.7 Iteration 7: Charts 72
3.7.8 Iteration 8: Mobile Application 78
CHAPTER 4: TESTING AND ANALYSIS 85
4.1 TEST PLAN 85

4.1.1 UNIT TESTING, TEST PLAN 85
4.1.2 SYSTEM TESTING, TEST PLAN 85
4.2 UNIT TESTING 86
4.2.1 Owner Login and Login Validation 86
4.2.2 Register Taxi 89
4.2.3 View Taxi 90
4.2.4 Update Taxi 92
4.2.5 Delete Taxi 93
4.2.6 Register Driver 95
4.2.7 View Driver 97
4.2.8 Update Driver 98
4.2.9 Delete Driver 100
4.2.10 Add Income 102
4.2.11 View Income 103
4.2.12 Update Income 104
4.2.13 Delete Income 105
4.2.14 Driver Age Chart 107
4.2.15 Owner Logout 109
4.2.16 Driver Login and Password Change 111
4.2.17 Driver Login (Mobile) 113
4.2.18 Driver Drowsiness Data (Mobile) 115
4.2.19 Owner Login in mobile app 118
4.2.20 Driver Login Validation (Mobile) 120
4.2.21 Eye Detection Test (Drowsiness Detection) 122
4.2.22 Drowsiness Alert Test (Drowsiness Detection) 123

4.2.23 Update Drowsiness Data (Web App) 124
4.3 SYSTEM TESTING 125
4.3.1 Sending Drowsiness API (Django) 125
4.3.2 Sending Drowsiness API and User Token for Mobile 127
4.3.3 Database Connection and Data Visualization 128
4.3.4 URL Run Time 130
4.4 CRITICAL ANALYSIS 131
4.4.1 Testing Analysis 131
4.4.2 Error and Error Handling 131
CHAPTER 5: CONCLUSION 136
5.1 LEGAL, SOCIAL AND ETHICAL ISSUES 136
5.1.1 LEGAL ISSUES 136
5.1.2 SOCIAL ISSUES 136
5.1.3 ETHICAL ISSUES 137
5.2 ADVANTAGES 137
5.3 LIMITATIONS 138
5.3.1 Drowsiness Detection Software 138
5.3.2 Web Application 138
5.3.3 Mobile Application 138
5.4 FUTURE WORK 139
CHAPTER 6: REFERENCES 140
CHAPTER 7: BIBLIOGRAPHY 143
CHAPTER 8: APPENDIX 144
8.1 APPENDIX A: PRE-SURVEY 144
8.1.1 PRE-SURVERY FORM 144

8.1.2 SAMPLE OF FILLED PRE-SURVEY FORMS 147
8.2 APPENDIX B: POST-SURVEY 150
8.2.1 POST-SURVEY FORM 150
8.2.2 SAMPLE OF FILLED POST-SURVEY FORMS 152
8.3 APPENDIX C: CODES 154
8.3.1 CODE OF THE UI 154
8.4 APPENDIX D: DESIGNS 160
8.4.1 GANTT CHART 160
8.4.2 WORK BREAKDOWN STRUCTURE 161
8.4.3 MILESTIONE 162
8.4.4 HIGH-LEVEL USECASE 163
8.4.5 WIREFRAME 165
8.5 APPENDIX E: SOFTWARE REQUIREMENTS SPECIFICATION **(SRS)** DOCUMENT 174
8.5.1 Introduction 174
8.5.2 Overall Description 175
8.5.3 Functional Requirements 179
8.5.4 External Interfaces Requirements 180
8.7 APPENDIX F: USER FEEDBACK 185
8.7.1 USER FEEDBACK FORM 185
8.7.2 SAMPLE OF FILLED USER FEEDBACK FORMS 187
8.7.3 WEB APPLICATION MANUAL (USER) 189
8.8 APPENDIX G: FUTURE WORK 191
8.8.1 READING FOR THE FUTURE WORK 191

Table of figures

Figure 1 Jugnoo Taxi 9
Figure 2 Yelowsoft (Yelowsoft, 2020) 10
Figure 3 Taxi Management System by AudenBerg Technologies (getsetproject, 2020) 11
Figure 4 Drowsiness Detector (Pyimagesearch, 2017) 12
Figure 5 Waterfall Model (McCormick, 2012) 15
Figure 6 Prototype Methodology (K, 2020) 17
Figure 7 Iterative Methodology 18
Figure 8 RUP Methodology (testbytes , 2019) 19
Figure 9 Majority of taxis 22
Figure 10 Taxi Management System for smooth business 22
Figure 11 Eye Tracking Feature 23
Figure 12 Main cause of read accidents 23
Figure 13 Eye Tracking Feature Opinions 24
Figure 14 View Date Time Feature 24
Figure 15 Owner Dashboard rating 25
Figure 16 Drowsiness Detection Software rating 26
Figure 17 Overall Project feedback 26
Figure 18 Overall Mobile Application rating 27
Figure 19 Requirements.txt 28
Figure 20 ERD Diagram 30
Figure 21 Use Case Diagram 31
Figure 22 Context Level Diagram 32
Figure 23 DFD Diagram 33
Figure 24 Register Driver 34
Figure 25 Register Taxi 34
Figure 26 Register Income 34
Figure 27 Owner Login 35
Figure 28 Driver Login 35
Figure 29 Driver Registration (Communication) 36
Figure 30 Taxi Registration (Communication) 36

Figure 31 Income Registration (Communication) 37
Figure 32 Owner Login (Communication) 37
Figure 33 Driver Login (Communication) 38
Figure 34 Owner (models.py) 39
Figure 35 Driver (models.py) 39
Figure 36 Register Driver (views.py) 41
Figure 37 Generic views for drivers 41
Figure 38 driver URLs 42
Figure 39 register_driver.html (1) 42
Figure 40 register_driver.html (2) 43
Figure 41 Driver Registration Form 43
Figure 42 view_driver.html 44
Figure 43 View Driver 44
Figure 44 update_driver_detail.html 45
Figure 45 Update Driver Page 46
Figure 46 delete.html 46
Figure 47 Delete Page 47
Figure 48 Taxi (models.py) 47
Figure 49 taxi_registration_view(views.py) 48
Figure 50 taxi registration (generic views) 48
Figure 51 taxi URLs 49
Figure 52 register_taxi.html 50
Figure 53 Taxi registration page 50
Figure 54 view_taxi.html 51
Figure 55 View Taxi page 51
Figure 56 update_taxi_detail.html 52
Figure 57 Update taxi page 52
Figure 58 Delete Taxi Page 53
Figure 59 drowsiness.py (1) 54
Figure 60 drowsiness.py (2) 55
Figure 61 drowsiness.py (3) 55

Figure 62 drowsiness.py (4) 56
Figure 63 Drowsiness Detection Software 58
Figure 64 Income (models.py) 58
Figure 65 add_income_view(views.py) 59
Figure 66 Income generic views 60
Figure 67 Income URLs 60
Figure 68 add_daily_income.html 62
Figure 69 Add Daily Income Page 62
Figure 70 Totalearning.html (1) 63
Figure 71 Totalearning.html (2) 64
Figure 72 Total Earnings Page 64
Figure 73 update_income.html 65
Figure 74 Update Income Page 65
Figure 75 Delete Page 66
Figure 76 Login View (view.py) 67
Figure 77 Login URLs 68
Figure 78 Login.html (1) 68
Figure 79 Login.html (2) 69
Figure 80 Login Page (1) 69
Figure 81 Login Page (2) 70
Figure 82 Sending POST request 71
Figure 83 Postman API 71
Figure 84 Age Chart (views.py) 72
Figure 85 Gender Chart (views.py) 73
Figure 86 Chart URLs 73
Figure 87 index.html (1) 74
Figure 88 index.html (2) 75
Figure 89 index.html (3) 76
Figure 90 Chart 77
Figure 91 models.dart 78
Figure 92 verify.dart 78

Figure 93 body.dart (1) 79
Figure 94 body.dart (2) 80
Figure 95 body.dart (3) 82
Figure 96 Login Screen (1) 82
Figure 97 Login Screen (2) 83
Figure 98 Drowsy Data in mobile 84
Figure 99 Owner Login Test (1) 86
Figure 100 Owner Login Test (2) 87
Figure 101 Owner Login Test (3) 87
Figure 102 Owner Login Validation Test 88
Figure 103 Taxi Register Test (1) 89
Figure 104 Taxi Register Test (2) 89
Figure 105 View Registered Taxi (Test 1) 90
Figure 106 View Registered Taxi (Test 2) 91
Figure 107 Update Registered Taxi (Test 1) 92
Figure 108 Update Registered Taxi (Test 2) 92
Figure 109 Delete Taxi Test (1) 93
Figure 110 Delete Taxi Test (2) 94
Figure 111 Delete Taxi Test (3) 94
Figure 112 Register Driver Test (1) 95
Figure 113 Register Driver Test (2) 96
Figure 114 View Driver Test (1) 97
Figure 115 View Driver Test (2) 97
Figure 116 Update Driver Test (1) 98
Figure 117 Update Driver Test (2) 99
Figure 118 Delete Driver Test (1) 100
Figure 119 Delete Driver Test (2) 100
Figure 120 Add Income Test (1) 102
Figure 121 Add Income Test (2) 102
Figure 122 View Income Test 103
Figure 123 Update Income Test (1) 104

Figure 124 Update Income Test (2) 104
Figure 125 Delete Income Test (1) 105
Figure 126 Delete Driver Test (2) 106
Figure 127 Delete Driver Test (3) 106
Figure 128 Driver Age Chart Test (1) 107
Figure 129 Driver Age Chart Test (2) 108
Figure 130 Owner Logout Test (1) 109
Figure 131 Owner Logout Test (2) 109
Figure 132 Owner Logout Test (3) 110
Figure 133 Driver Login Test (1) 111
Figure 134 Driver Login Test (2) 111
Figure 135 Change Password Test (1) 112
Figure 136 Change Password Test (2) 112
Figure 137 Driver Login Test (1) 113
Figure 138 Driver Login Test (2) 115
Figure 139 Driver Drowsiness Data Test (1) 116
Figure 140 Driver Drowsiness Data Test (1) 117
Figure 141 Owner Login in mobile app Test (1) 118
Figure 142 Owner Login in mobile app Test (2) 119
Figure 143 Driver Login Validation in mobile Test (1) 120
Figure 144 Driver Login Validation in mobile Test (2) 121
Figure 145 Running Drowsiness Detection Software and tracking eye Test (1) 122
Figure 146 Running Drowsiness Detection Software and tracking eye Test (2) 123
Figure 147 Drowsiness alert after feeling drowsiness Test 123
Figure 148 Updated Drowsiness Data Test (1) 124
Figure 149 Updated Drowsiness Data Test (2) 124
Figure 150 Sending Drowsiness API Test (1) 125
Figure 151 Sending Drowsiness API Test (2) 126
Figure 152 Sending Drowsiness API and User Token Test (1) 127
Figure 153 Sending Drowsiness API and User Token Test (2) 127
Figure 154 Database connection and data visualization Test (1) 128

Figure 155 Database connection and data visualization Test (2) 129
Figure 156 URL Run Time 130
Figure 157 Integrity Error 132
Figure 158 Integrity Error Solution 132
Figure 159 Invalid View 133
Figure 160 Flutter API Error 134
Figure 161 Flutter API Error Solved 135
Figure 162 Pre-Survey (1) 144
Figure 163 Pre-Survey (2) 145
Figure 164 Pre-Survey (3) 146
Figure 165 Pre-Survey Sample (1) 147
Figure 166 Pre-Survey Sample (2) 148
Figure 167Pre-Survey Sample (3) 149
Figure 168 Post Survey (1) 150
Figure 169 Post Survey (2) 151
Figure 170 Post Survey Sample (1) 152
Figure 171 Post Survey Sample (2) 153
Figure 172 Logout.html code 154
Figure 173 Password Change (1) 154
Figure 174 Password Change (2) 155
Figure 175 password change view 156
Figure 176 base.html (1) 157
Figure 177 base.html (2) 157
Figure 178 Drowsiness.html (1) 158
Figure 179 Drowsiness.html (2) 159
Figure 180 Gantt Chart 160
Figure 181 Work Breakdown Structure 161
Figure 182 Milestone 162
Figure 183 Wireframe 1 Dashboard 165
Figure 184 Wireframe 2 Driver Registration 166
Figure 185 Wireframe 3 Taxi Registration 167

Figure 186 Wireframe 4 Driver Information 167
Figure 187 Wireframe 5 Taxi Information 168
Figure 188 Wireframe 6 Update Driver Registration 169
Figure 189 Wireframe 7 Update Taxi Details 170
Figure 190 Wireframe 8 Add Daily Income 170
Figure 191 Wireframe 9 View Total Earning 171
Figure 192 Wireframe 10 Mobile login 172
Figure 193 Wireframe 11 View Drowsiness Data 173
Figure 194 System Architecture 175
Figure 195 Use Case 176
Figure 196 Dashboard 180
Figure 197 Taxi Registration 181
Figure 198 Total Earning 181
Figure 199 Driver Drowsiness 182
Figure 200 User Feedback form (1) 185
Figure 201User Feedback form (2) 187
Figure 202 Filled User Feedback form (1) 187
Figure 203 Filled User Feedback form (2) 188
Figure 204 Web application Manual 189
Figure 205 GPS Tracking System 191
Figure 206 Payment Gateway 192

Table of tables

Table 1 Project Comparison 14
Table 2 RUP Phase deliverables and success criteria 21
Table 3 Owner Login Test 87
Table 4 Owner Login Validation Test 88
Table 5 Taxi Register Test 90
Table 6 View Registered Taxi Test 91
Table 7 Update Registered Taxi Test 93
Table 8 Delete Taxi Test 95
Table 9 Register Driver Test 96
Table 10 View Driver Test 98
Table 11 Update Driver Test 99
Table 12 Delete Driver Test 101
Table 13 Add Income Test 103
Table 14 View Income Test 103
Table 15 Update Income Test 105
Table 16 Delete Driver Test 107
Table 17 Driver Age Chart Test 108
Table 18 Owner Logout Test 110
Table 19 Change Password Test 113
Table 20 Driver Login Test 115
Table 21 Driver Drowsiness Data Test 117
Table 22 Owner Login in mobile app Test 119
Table 23 Driver Login Validation in mobile Test 121
Table 24 Eye Detection Test 123
Table 25 Drowsiness Alert Test 124
Table 26 Updated Drowsiness Data Test 125
Table 27 Sending Drowsiness API Test 126
Table 28 Sending Drowsiness API and User Token Test 128
Table 29 Database connection and data visualization Test 129
Table 30 URL Run Time 130

Table 31 Workout Feature..... 179
Table 32 Performance Requirement 184
Table 33 Safety and Security Requirement..... 184

Table of Abbreviation

SN	ACRONYMS	FULLFORMS
1.	UI	User Interface
2.	UX	User Experience
3.	HTML	Hypertext Mark-up Language
4.	CSS	Cascading Style Sheet
5.	Js	Java Script
6.	App	Application
7.	GUI	Graphical User Interface

CHAPTER 1: INTRODUCTION

1.1 PROJECT DESCRIPTION

Taxi service has always been an integral part of the transportation business. Back in the year 2000, the government of Nepal banned the registration of new taxis inside Kathmandu valley declaring that 7500 taxis would be enough but after the sudden increment of the population inside Kathmandu valley, both the general public and the taxi business got affected (SUBEDI, 2020). In the present COVID situation, the general public feels safer to go somewhere in a taxi rather than in a public vehicle. This situation is massively beneficial to the taxi business. There are also some online companies that run the taxi business like Pathao, Taximandu, Sarathi, etc. But all these services only operate inside the valley. Most of the Taxi business in Nepal are owned by organizations or privately and they hire drivers to drive their taxis. My project focuses on those owners who are having a very hard time managing and keeping track of their taxi business. Apart from that, my project will also focus on the safety of drivers with regards to road accidents.

Microsleep is temporary sleep which is also called drowsiness which could last for just a fraction of a second or could even last up to 30 seconds. Microsleep is one of the main causes of road accidents. It is a common experience of many drivers. Usually, to prevent drowsiness, drivers stop the cars for some time and take a quick nap. This process can be useful but is inconvenient. Let's take an example of a driver feeling drowsy while driving in the middle of heavy traffic. The driver has no chance of taking a nap. Automobile companies like Audi, Volvo, etc. have built-in AI that solves this problem. Not only are these cars expensive but thinking about bringing those AI cars into Nepal is next to impossible because of the tax and the increment of price. This is where my computer vision software comes into play.

Computer vision can be defined as a field of study that seeks to develop techniques to help computers see and understand the content of digital images such as photographs and videos (Brownlee, 2019). It basically helps computers to see.

1.2 CURRENT SCENARIO

It is no new fact that Nepali taxi drivers are overworked. Driving all day long makes them feel sleepy of course but we cannot blame them because after all, they need the money. In 2017-18, 2541 road deaths were officially reported in Nepal, which is equivalent to a fatality rate of 8.59 per 100,000 population (International Bank for Reconstruction and Development/The World Bank , 2020). According to the latest WHO data published in 2018, road traffic accidents in Nepal reached 3.07% of total deaths. Nepal ranks 76th in the world for road traffic accidents (World Health Ranking, 2020).

To keep track of your own business can be very stressful. Let me give an example of a Taxi business owner who must keep track of his/her 30 Taxi. There is a very less chance of remembering all his/her taxis number and the detail of the driver as well. Many new drivers could also join in the middle. There might also be difficulty in tracking day to day income which will be generated at the end of the day by all the drivers. There could be also a huge income related problem in a taxi company opened in partnership. Ever So, focusing on this problem I have decided to make a web application for taxi business owners that can ease this workload from them.

1.3 PROBLEM DOMAIN AND PROJECT AS A SOLUTION

There are two parts to my project. The Taxi management system will be specially focused on the owners of the taxi business which will help the owners maintain all the records of their drivers and the taxi they are using. The product will be a web application dashboard with user-friendly UI. The other part of my project is computer vision security for drivers, this software will track the eye of the driver and if the driver feels distracted or feels drowsy, the software fires an alarm and alerts the driver. This software is the alternative to those expensive AI cars which will have inbuilt drowsiness detection software. This software will prevent drivers from road accidents and aims to save the life of drivers. Furthermore, drivers will have their own profile where they can login and see all their drowsiness data.

1.4 AIM AND OBJECTIVES

The aim of this project is to create a web application for the taxi owners for handling and recording their business data and distraction detection software for the drivers to given them extra security while driving and prevent them from accidents.

The objectives of this project are as follows:

- To reduce the chance of road accidents specially caused by distraction from the driver.
- To help drivers with their micro sleep problem while driving.
- To learn how to use facial landmarks and apply it in the project.
- To let driver, access their drowsiness data.
- To make a useful web app for the taxi owners where they can manage their business.

1.5 STRUCTURE OF THE REPORT

The following format below showcases how the report is structured and organized.

1.5.1 BACKGROUND

This part of the project shows the research and background related to the proposed project. It consists information about the end users, review of similar projects, understanding the solution, comparison between similar projects and review of various technology used.

1.5.2 DEVELOPMENT

This chapter consists information about the methodology considered and used and reasons behind it. It also contains survey results, requirements needed for the project, project design and implementation where project codes and working UI screenshot are shown.

1.5.3 TESTING AND ANALYSIS

This chapter contains various testing, test evaluations and explanation of error fixing during development.

1.5.4 CONCLUSION

This chapter contains information about different issues that could be violated. It also contains advantages, limitation, and future work of the project.

CHAPTER 2: BACKGROUND

2.1 ABOUT THE END USERS

The end user of this project will be the owner of taxi company. Both small and big taxi company which is run by owner are the targeted end user of the project. The general assumption as a project developer is that, till now all those targeted end users of this project have been doing the taxi business manually without any software. They are willing to fix this management problems and make their business run quick and smoothly without any confusions. A project like this can help all those taxi owners all over the Nepal struggling with business management as already discussed in problem statement section.

Taxi drivers are also the end user of this project. To prevent from road accident caused through microsleep, Drowsiness Detection Software will be installed on the taxi which is operated by the same company who buys the project. This will be the main feature of the project which will attract the end user even more. The taxi company owners also fear for their drivers as well as the health of the taxi. This feature will solve that problem too.

2.2 UNDERSTANDING THE SOLUTION

2.2.1 Overview of the System

This project delivers two different application (which are web application and mobile application) and drowsiness detection software and all are connected to each other. Web application is a taxi management system which is for the small or big taxi companies for easy working experience. Drowsiness detection software is for all the drivers of that taxi company which gives driver a security feature while driving and prevents from road accidents. Mobile application is also for driver where the driver will be able to see the drowsiness stats at the end of the day in his/her phone and can be self-aware.

2.2.1.1 How will Taxi Management System work?

The Taxi Management System is a web application where the taxi company owner can register drivers, taxi, income and do CRUD operations. This can be very useful because thinking in large scale, the owner can easily see the record of which taxi is drove by which driver and owner can also see the total income generated by the company.

The web application also has login feature for drivers where the driver can see his/her drowsiness data.

2.2.1.2 How will Drowsiness Detection Software work?

The idea behind this software is to detect the eye movement of the driver during driving and if the driver feels drowsy then the system will fire an alarm to wake up the driver and prevent the driver from accident. Once the alarm starts, the system automatically records the date and time of the current situation. The data will be connected to both web application and mobile application.

2.2.1.3 How will Mobile Application work?

The Mobile application is only for company driver. It consists of the same date and time mentioned above in 2.2.1.2. After the alarm starts and stops in the software, the driver can log in into the mobile application and can see the data. This data can help driver so be self-aware and track the sleep.

2.2.2 Building Process

The project will be using multiple tools and technologies for making this project. For web application, Django will be used. For mobile I have chosen flutter and for drowsiness detection software python and OpenCV will be used. All the software and hardware requirements are listed below:

2.2.1 Software

- Django

  Django is a python-based opensource web framework. It follows MTV(Model-Template-View) pattern. A framework is nothing more than the series of modules that facilitates development. I am using Django for making taxi management web

application and I choose Django because It is a very powerful web framework containing lots of built-in awesome features which will boost the production speed. Django offers authentication which can be used to build a fully functional authentication system by running a simple command. Along with that Django offers a lot of built-in features.

- OpenCV

  OpenCV is an open source computer vision and machine learning software. OpenCV is a library which has more than 2500 optimized algorithms that helps to detect, recognize faces. It can also track movements and many more. I am using OpenCV library for making Driver Distraction Detection software. I am using OpenCV specifically because it already has many face trained modules and will not complicate my project. Now-a-days many face recognized software are made using OpenCV.

- PostgreSQL

  PostgreSQL is an advance, enterprise-class and open-source relational database system (postgresqltutorial, 2020). I am using PostgreSQL because of the following reasons:

  - Open Source DBMS
  - Diverse Community
  - Diversified extension features

- HTML, CSS, Bootstrap, Java Script for frontend.
- Flutter

  Flutter is Google's UI toolkit for building mobile, web and desktop application from a single codebase (flutter , 2021).

- Django RestAPI

  Django REST framework is a powerful and flexible toolkit for making Web APIs. I am using Django RestAPI because of following reasons:

  - Serialization supports both ORM and non-ORM data source

- Easy to use with Django
- Easy Installation and very understandable documentation.

2.2.2 Hardware

- Laptop

  Since my drowsiness feature is a working prototype, laptop is needed to run OpenCV code for driver distraction software. I will also use the built-in camera to detect the face of the driver.

- External Camera (Optional)

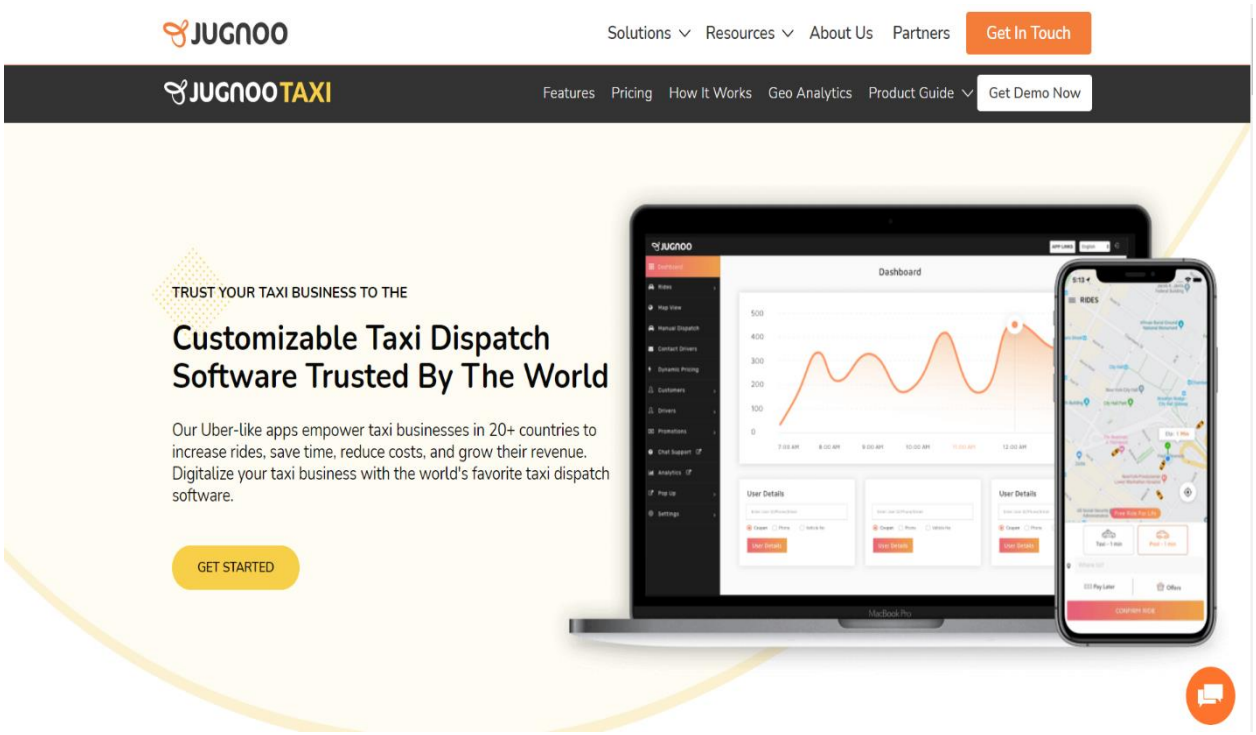
  Drowsiness feature need camera to detect face so, I need external camera which will be connected to my laptop or I can also detect face directly from the **built-in camera of laptop**.

- Mobile or Mobile emulator

  Since, I will be having a mobile application so mobile or emulator is needed for running the app.

2.3 SIMILAR PROJECTS

2.3.1 JUGNOO TAXI



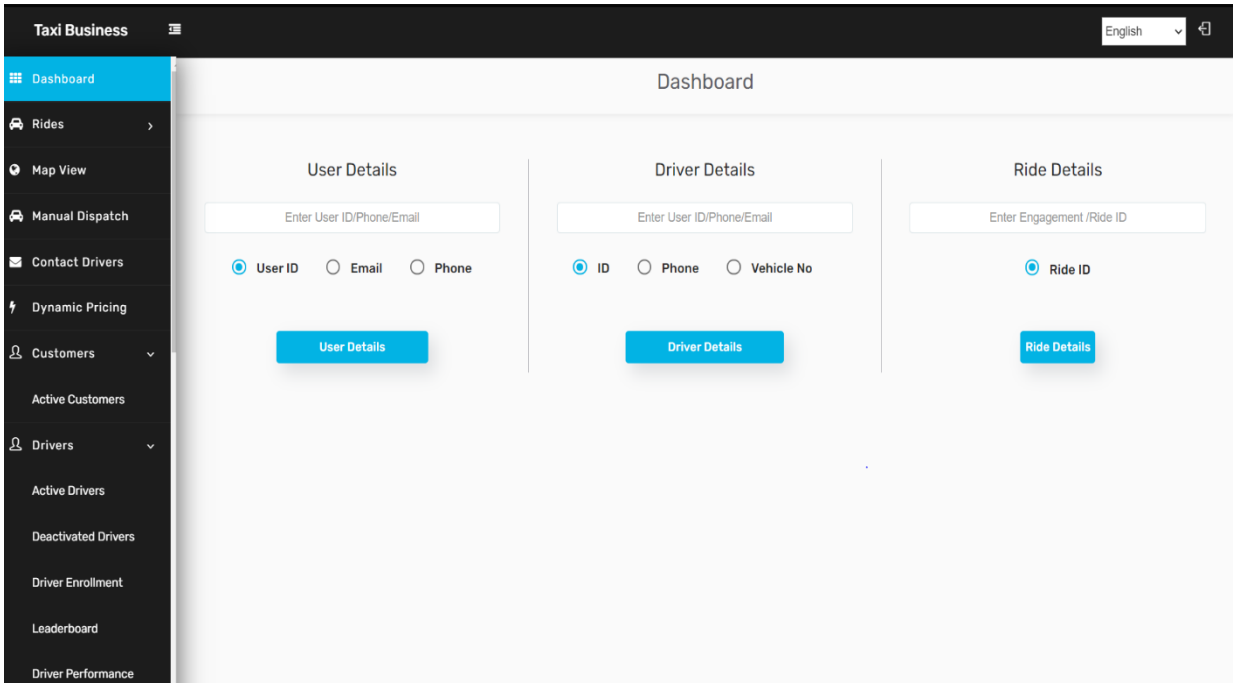


Figure 1 Jugnoo Taxi

Jugnoo Taxi is an Uber clone app which manages drivers, customers, bookings along with real time tracking features. It has app for both driver and customer along with admin dashboard. This software makes handling of taxi business much easier.

Jugnoo has also claimed to be India's 3rd Largest ride-hailing company that operates in 50+ cities (Jugnoo, 2020). It was founded on November 2014 and is the India's first taxi aggregator with the objective of making daily commute easy and reliable (Jugnoo, 2020). Since my project is focused on admin dashboard, following are the features provided by Jugnoo

Features:

- Secure Authentication
- Live Tracking
- Manage Drivers
- Real time fleet tracking

2.3.2 Yelowsoft

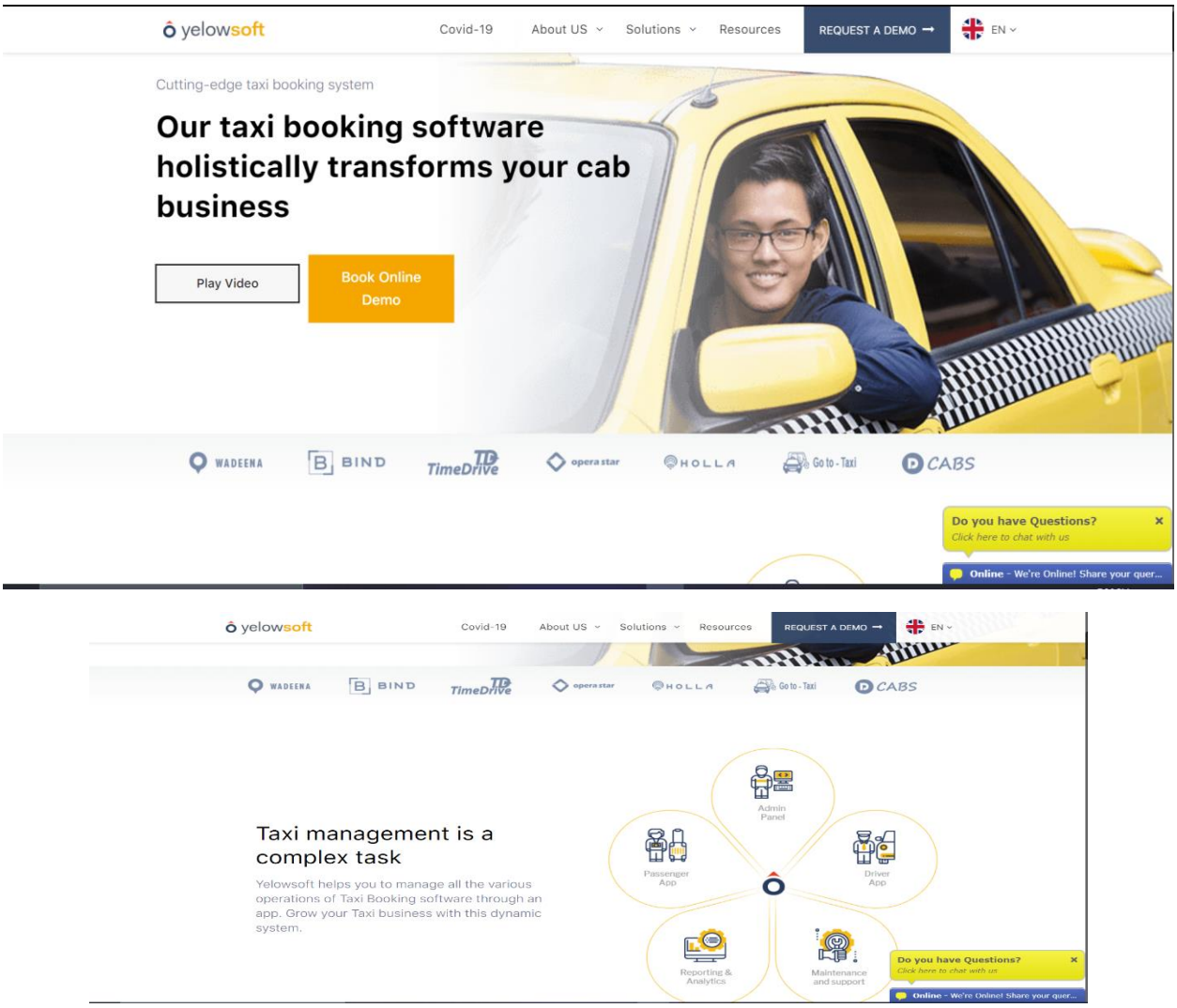


Figure 2 Yelowsoft (Yelowsoft, 2020)

Yelowsoft is a taxi dispatch software, founded in 2017. It has a centralized admin panel that manages all operations related to taxi business. It manages drivers as well as vehicle.

Features:

- Live Tracking
- Manages Drivers as well as vehicles
- Secure Authentication

2.3.3 Taxi Management System by AudenBerg Technologies India

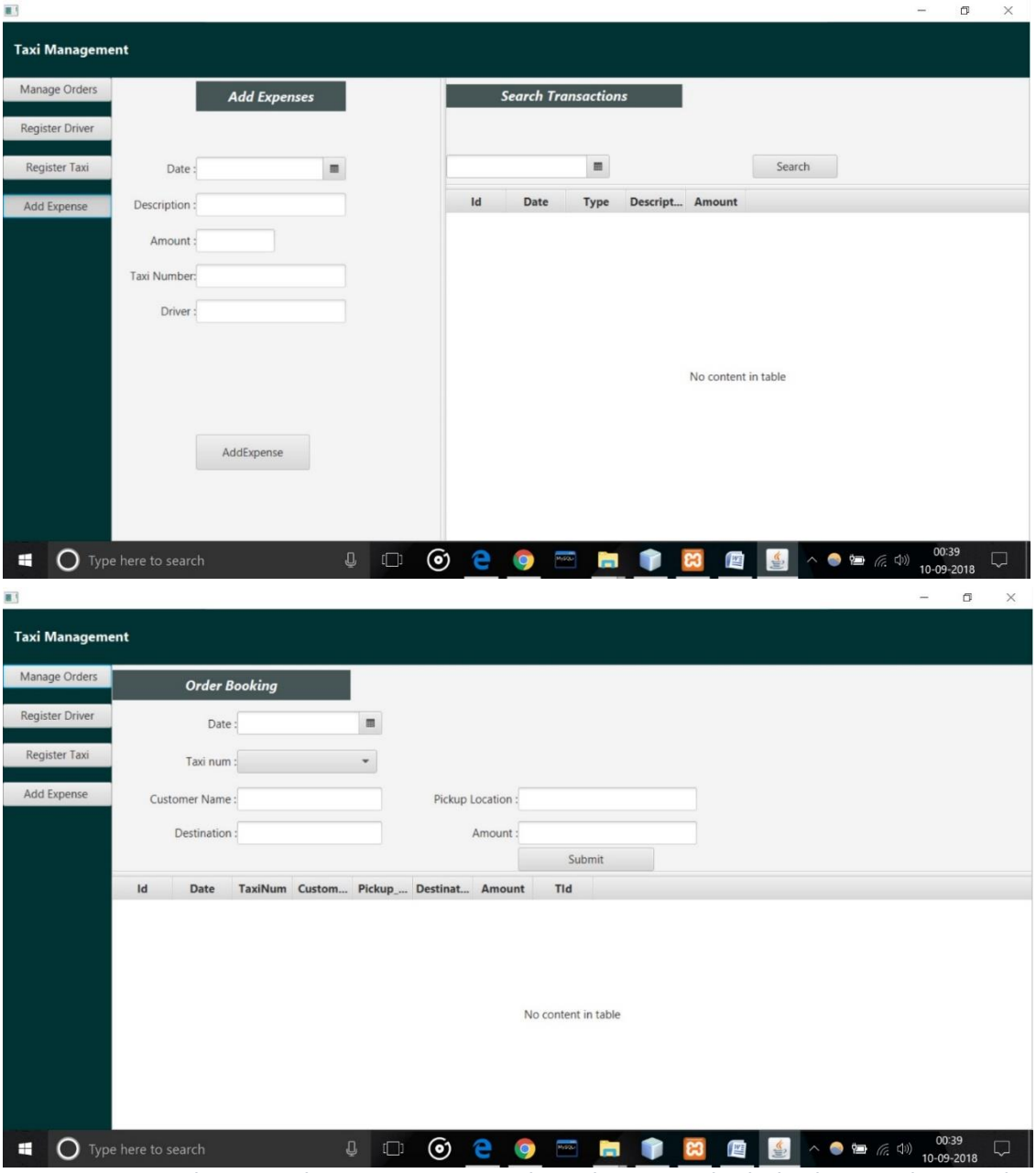


Figure 3 Taxi Management System by AudenBerg Technologies (getsetproject, 2020)

This is a simple taxi management software created by AudenBerg Technologies that provides decent admin dashboard for the taxi business.

Features:

- Register Driver
- Register Taxi

- Booking
- Income

2.3.4 Drowsiness Detector by Pyimagesearch

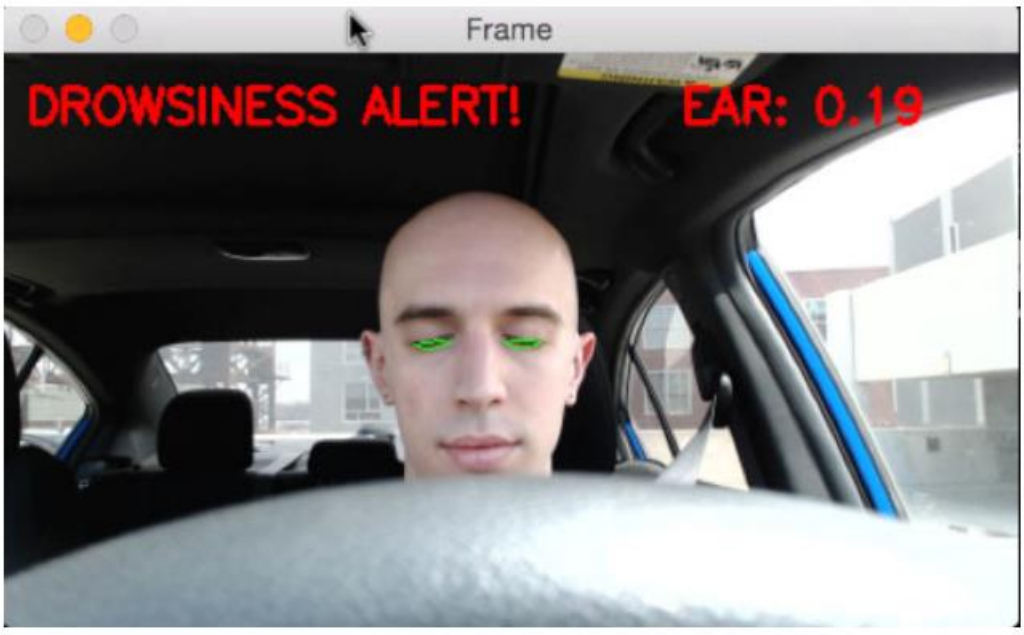


Figure 4 Drowsiness Detector (Pyimagesearch, 2017)

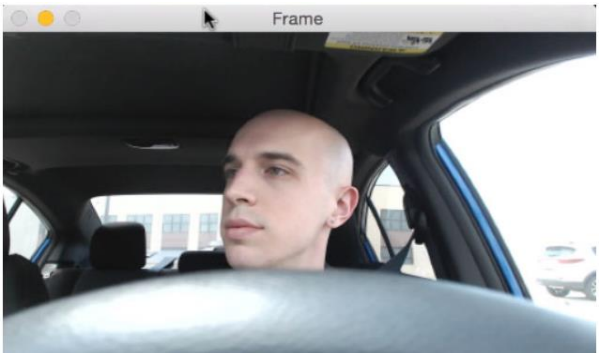
Pyimagesearch is a learning website and the owner of this website is Adrian Rosebrock. This website teaches computer vision, deep learning and OpenCV. This is one of the projects of Adrian, where he made a drowsiness alert system using OpenCV. My drowsiness feature is inspired by this project and most of the feature will be similar as the above project.

Features:

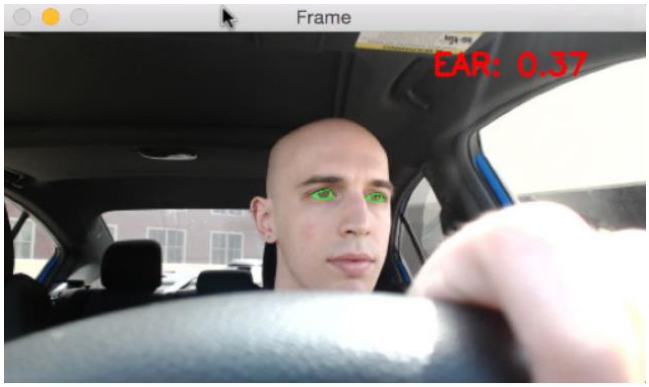
- Eye Tracking
- Alert Sound
- Facial Landmark Localization.

This project can be breakdown into following step:

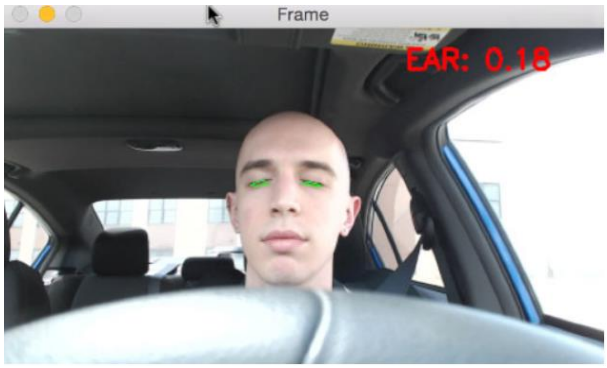
Step1: Looking for the faces in the input video stream.



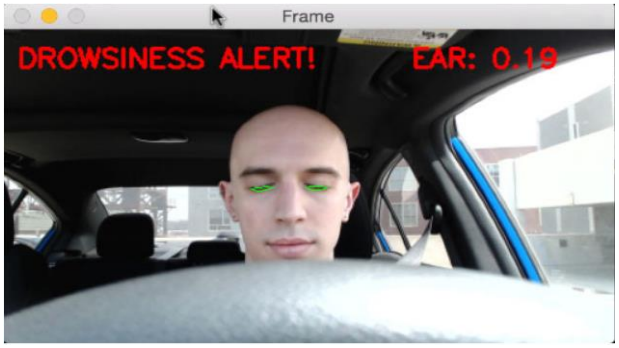
Step 2: Applying facial landmarks localization to extract the eye region from the face.



Step 3: Computing the eye aspect ratio to determine if the eyes are closed.



Step 4: Firing an alarm if the eyes have been closed for a sufficiently long enough time.



2.4 COMPARISIONS

Features	My Project	Jugnoo Taxi	Yelowsoft	AudenBerg Taxi Manager
Design	**Good**	**Excellent**	**Fair**	**Poor**
Dashboard	√	√	√	√
Live Tracking	X	√	√	X
Real Time Fleet Tracking	X	√	√	X
Manage Drivers	√	√	√	√
Manage Taxi	√	√	√	√
Secure Authentication	√	√	√	X
Income Generator	√	√	√	X

Table 1 Project Comparison

Comparing all the above projects with my project, I have found that my project will be lacking tracking system which is almost among all the projects. But in overall comparison, only my project will have feature to see all the drowsiness stats of the user.

CHAPTER 3: DEVELOPMENT

3.1 CONSIDERED METHODOLOGIES

- Waterfall Methodology

  The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach (The Economic Times, 2020). Basically, in waterfall methodology, all the requirements are gathered at first and then the tasks are divided into different phases sequentially. The documentation and testing happen at the end of each phase, which helps maintaining the quality of the project.

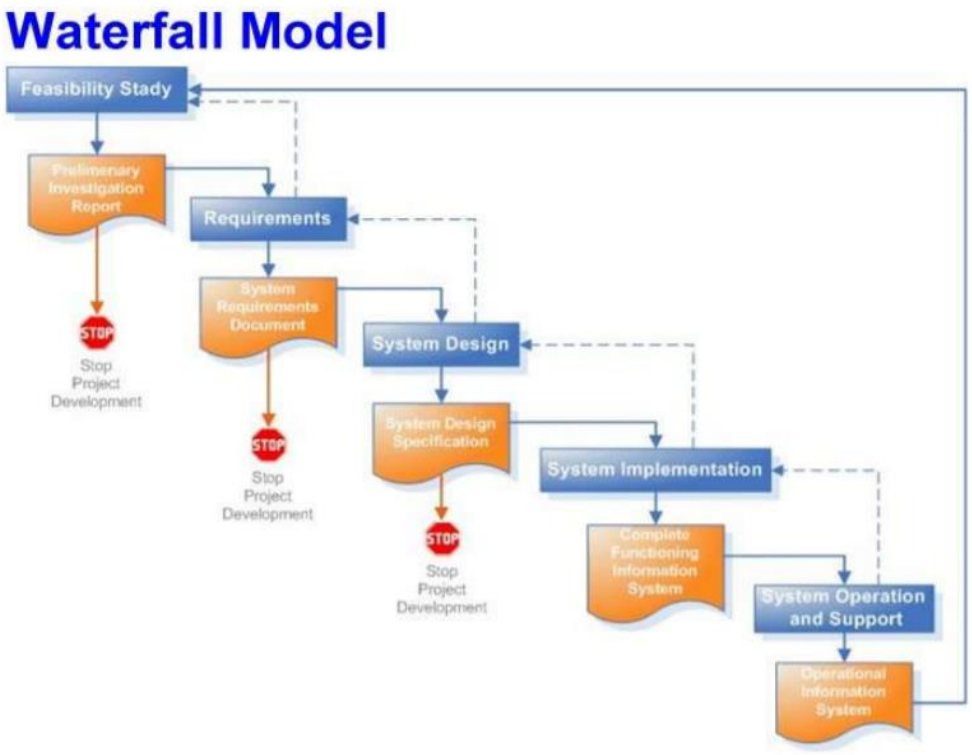


Figure 5 Waterfall Model (McCormick, 2012)

Pros:

- Requirement is clear before development starts.

- Each phase is completed in specified period after that it moves to next phase.
- As its linear model, it's easy to implement.
- The amount of resources required to implement this model are minimal.
- Each phase proper documentation is followed for the quality of the development.

Cons:

- The problems with one phase are never solved completely during that phase and in fact many problems regarding a phase arise after the phase is signed off, this result in badly structured system.
- If client want the requirement to be changed, it will not be implemented in the current development process (S.Balaji, 2012)

- **Prototype Methodology**

  Prototype Methodology is a software development model in which prototype is built, tested and is reworked until an acceptable prototype is achieved. Different phases of prototyping model are as follows:

  - Requirement gathering and analysis
  - Quick design
  - Build design
  - Initial user evaluation
  - Refining prototype
  - Implement Product and maintain (Guru 99, 2020)

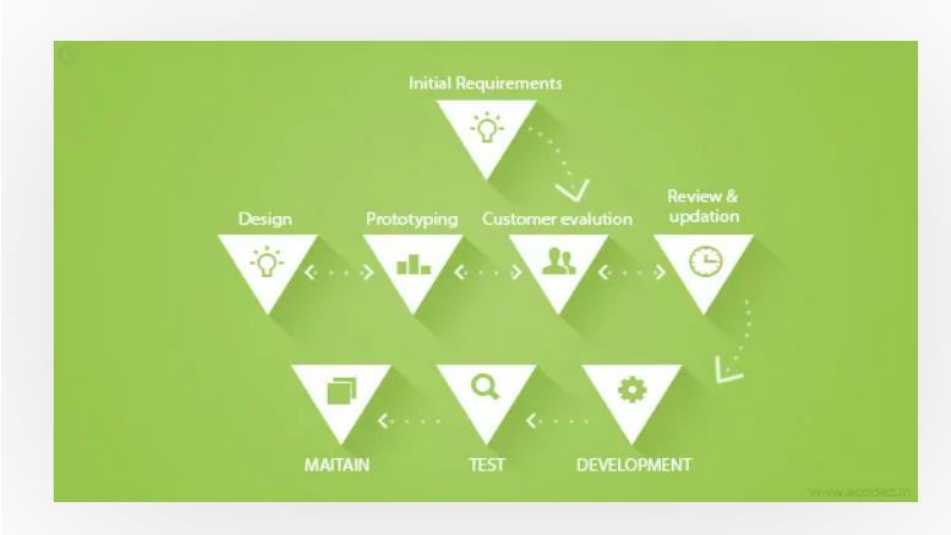


Figure 6 Prototype Methodology (K, 2020)

Pros:

- Gives clear idea about the functional process of the software.
- Reduces the risk of failure in a software functionality.
- Assists well in requirement gathering and the overall analysis

Cons:

- Chances of extension in management cost.
- Too many changes affect the workflow of the software.
- Excessive involvement of client can affect processing (K, 2020).

- **Iterative Methodology**

  Iterative model is one of the popular models adopted in SDLC (Software Development Life Cycle) that particularly focuses on small chunks of development and enhancing or evolving them to final software (educba, 2020). Different phases of iterative methodology are as follows:

  - Requirement Phase
  - Design Phase

- Development Phase
- Testing
- Review Phase

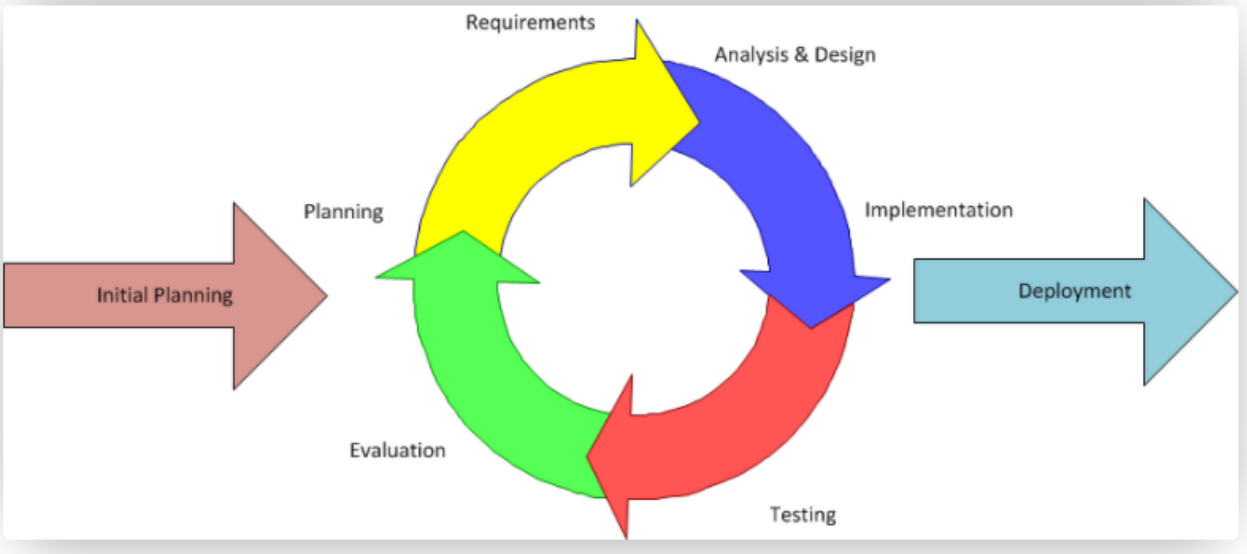


Figure 7 Iterative Methodology

Pros:

- Each stage has dedicated design phase which makes product more optimized.
- Parallel development can be planned which will save time.
- The requested changes can be easily implemented at each stage.

Cons:

- Not suitable for smaller projects.
- The progress of project is highly dependent upon the risk analysis phase.
- More management option is required.

3.2 SELECTED METHODOLOGY

- RUP Methodology (Rational Unified Process)

  The RUP methodology is an agile software development methodology which provides a disciplined approach to assigning tasks and responsibilities within a development organization (Kruchten, 2004). The goal of RUP methodology is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

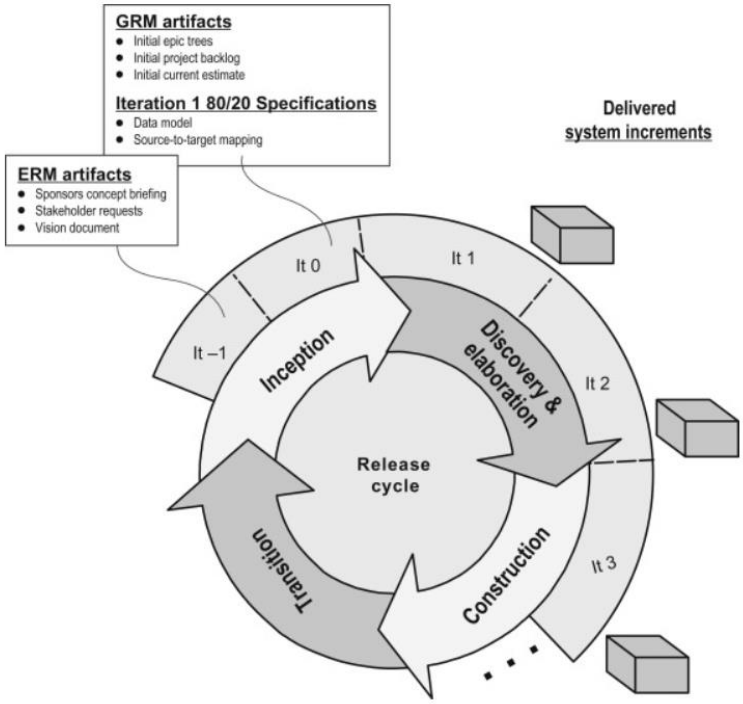


Figure 8 RUP Methodology (testbytes , 2019)

RUP lifecycle is divided into four distinct phases and they are as follows:

- Inception phase: This phase provides a general overall vision of the project. In this phase, the project is determined whether it is worth pursuing or not and gathering of resources is done.

- Elaboration phase: This phase contains analyzing the requirements and architecture of the system. It also analyzes the stability of projects and project costs.
- Construction phase: This phase contains design, development and testing of project.
- Transition phase: In this phase, product is finished, released and delivered to customers. Handling the bug also comes under this phase.

The following table shows the phase deliverables and success criteria of RUP methodology:

Phase	Deliverables	Success Criteria
Inception	• Vision Document • Initial Use Case • Initial Risk Assessment • Preliminary Project Prototypes	• Requirement Understanding • Credible cost and Schedule Estimates • Actual vs Planned Expenditures
Elaboration	• Revised use case • Management Plan • Phase Plan • Iteration Plan • Test Plan • Updated Risk Assessment • Software Architecture Description	• Baseline document vision • Assessing initial iterations during construction • Schedule estimates
Construction	• Individual Iteration plans	• Product stable • Ready for release

	• Release description document • Test case and results • Deployment plan • User Documentation	• Actual vs planned expenditures
Transition	• Final Product Release • Updated Product Documentation • Analysis of Project	• Customer Acceptance • User Satisfaction

Table 2 RUP Phase deliverables and success criteria

Advantages of RUP Methodology are as follows:

- This methodology allows to deal with frequent changing requirements which will occur during the making of the project.
- It emphasizes the need for accurate documentation.
- It forces integration to happen throughout the software development (Mrsic, 2017)
- It helps to identify the issues early in the process life cycle.

3.3 SURVEY RESULTS

3.3.1 PRE-SURVEY RESULTS

Before beginning my project, a pre survey was taken related to the project. So, below are the results of the survey:

3.3.1.1 Taxi mostly organized by

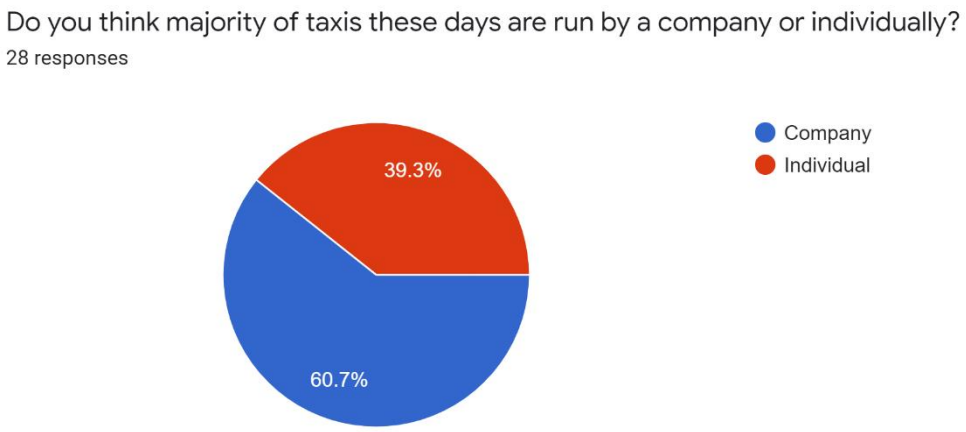


Figure 9 Majority of taxis

According to the general public, majority of taxis in Nepal are run by Company. Not only the public but I have also spoke to some drivers in my neighbor and according to them, most of the drivers fear to buy their own taxi and use the taxi of other organization. This concludes that this project should include some management for those taxi company.

3.3.1.2 Will taxi management system make business smooth?

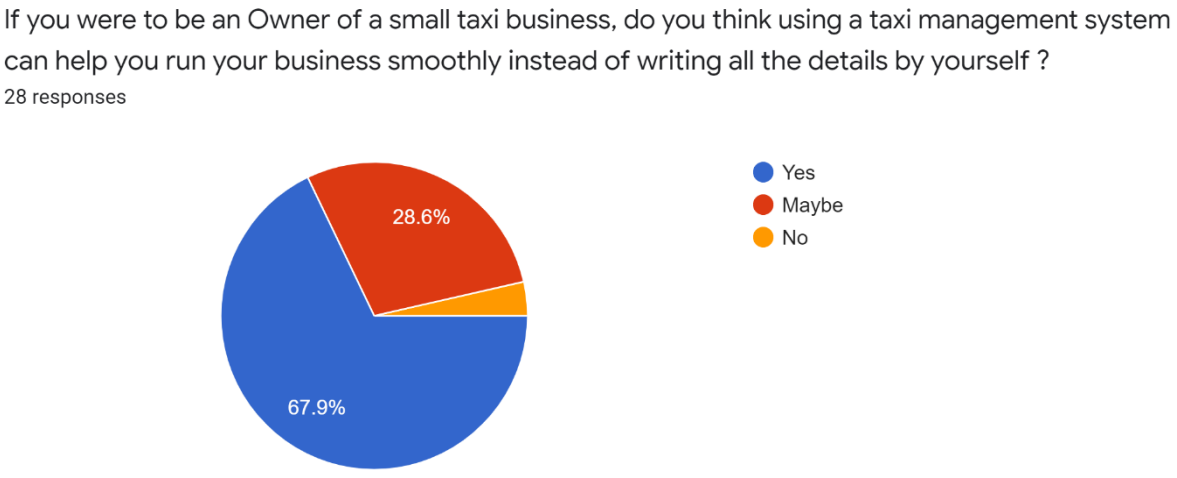


Figure 10 Taxi Management System for smooth business

From the above survey, it seems that people believe that taxi management system is needed to make business smoother.

3.3.1.3 Eye Tracking

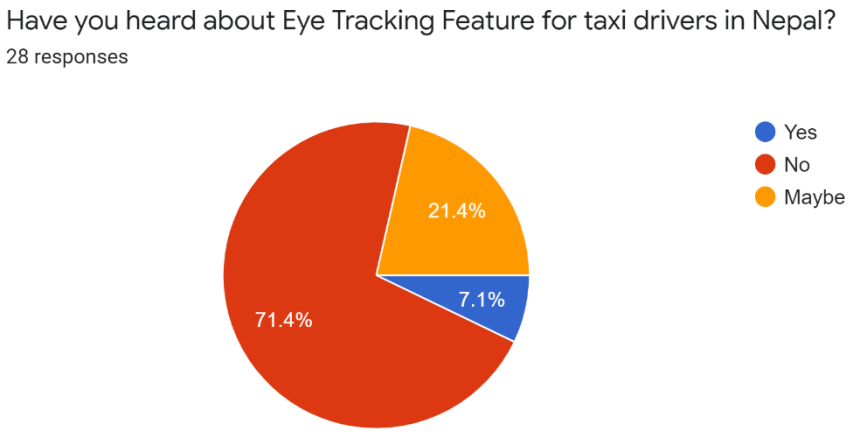


Figure 11 Eye Tracking Feature

From the above survey, out of 28 people 71.4% haven't heard about eye tracking feature which makes my project new to people.

3.3.1.4 Main cause of read accident

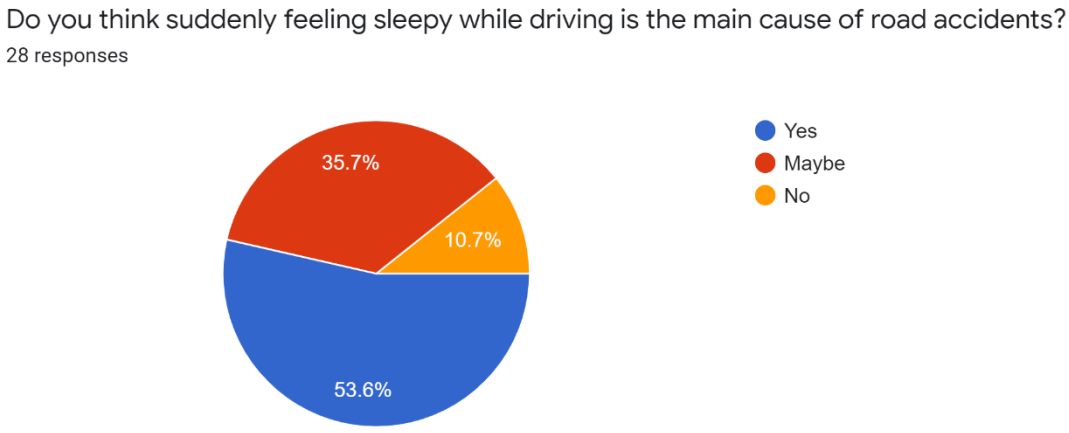


Figure 12 Main cause of read accidents

According to above survey, one of the main causes of road accident is feeling sleepy so my project should prevent people from accidents by making a eye tracking system.

3.3.1.5 Eye Tracking Feature Opinions

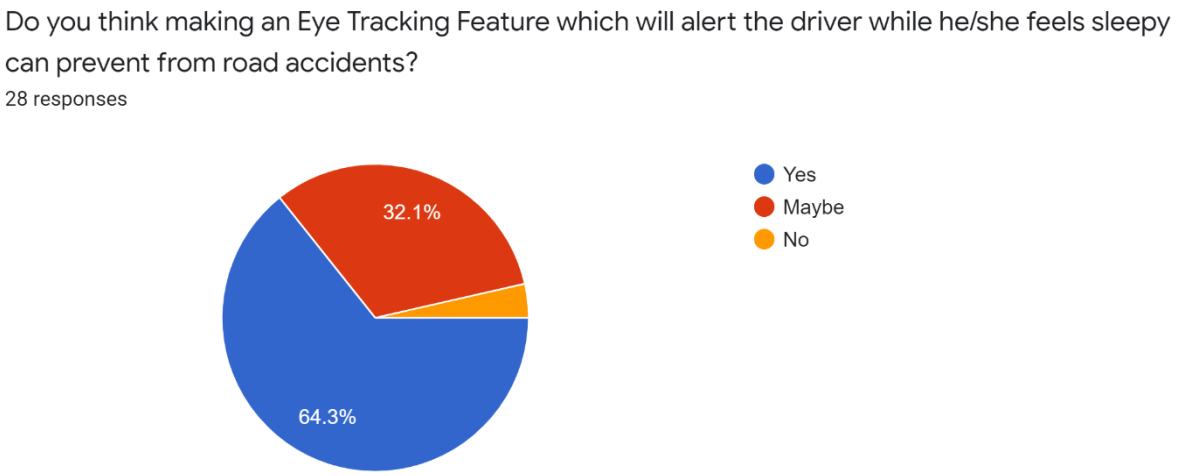


Figure 13 Eye Tracking Feature Opinions

Here, I asked about a proposed solution to prevent road accidents and 64.3% liked the idea of drowsiness detection system.

3.3.1.6 View Date Time Feature

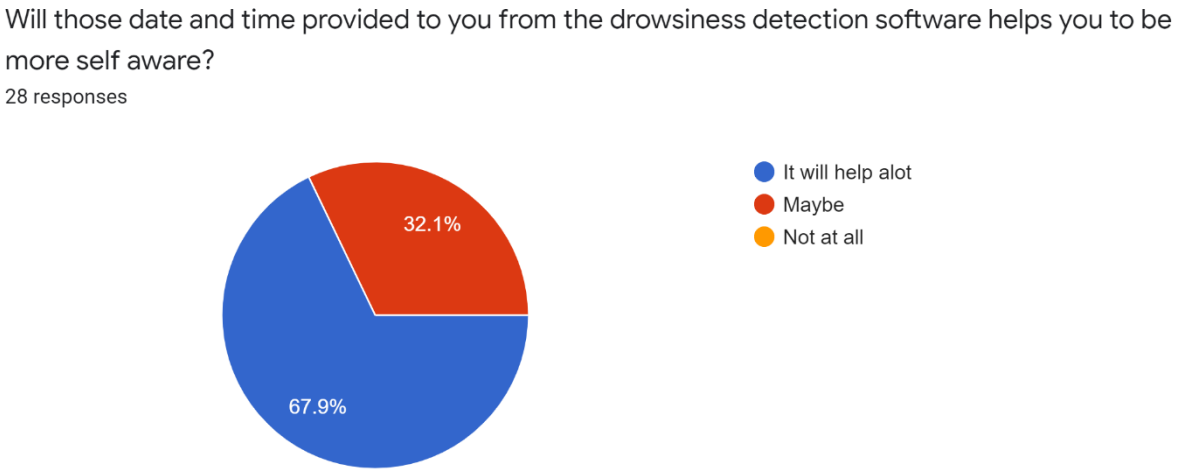


Figure 14 View Date Time Feature

From this survey I got a new feature to add on my project. People think that looking at the time of drowsiness time will help them a lot in maintaining their sleep and make people more self-aware.

3.3.2 POST-SURVEY RESULTS

After the completion of project, I took the post survey to know whether people understood my project or not and whether my project will solve the real-life problems of not.

3.3.2.1 Owner Dashboard

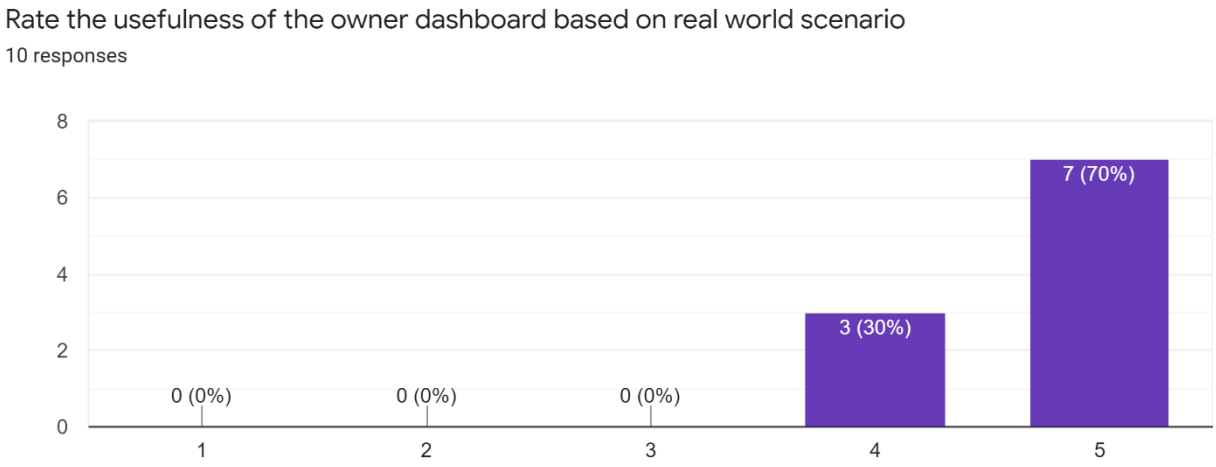


Figure 15 Owner Dashboard rating

The owner dashboard got good response and I am happy for it. Lots of future work can still be done.

3.3.2.2 Drowsiness Detection Software

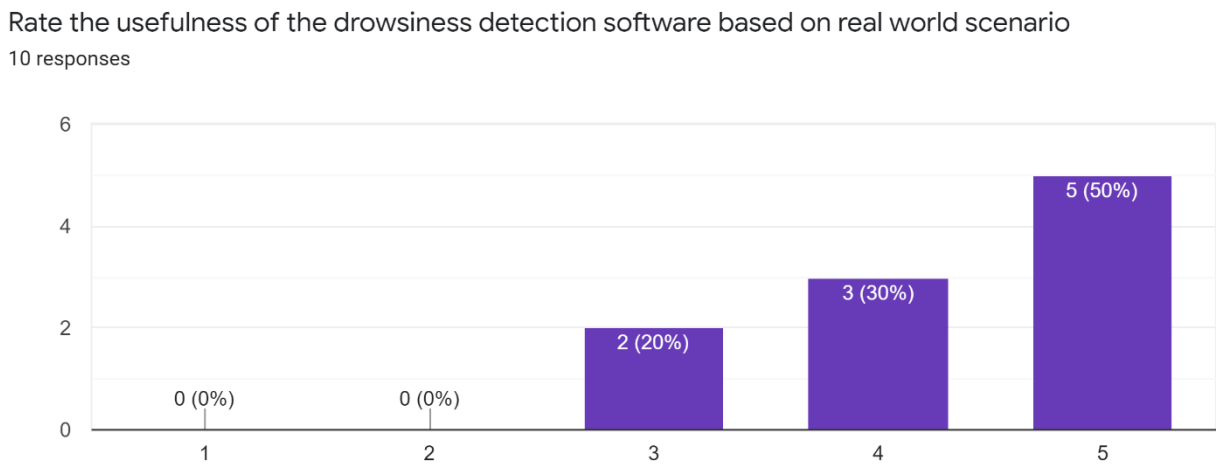


Figure 16 Drowsiness Detection Software rating

The software was completely new to many people including me. I got some positive feedback for the drowsiness software. This is just a prototype model and my intention were just to show the idea so more future work can be done.

3.3.2.3 Overall Project

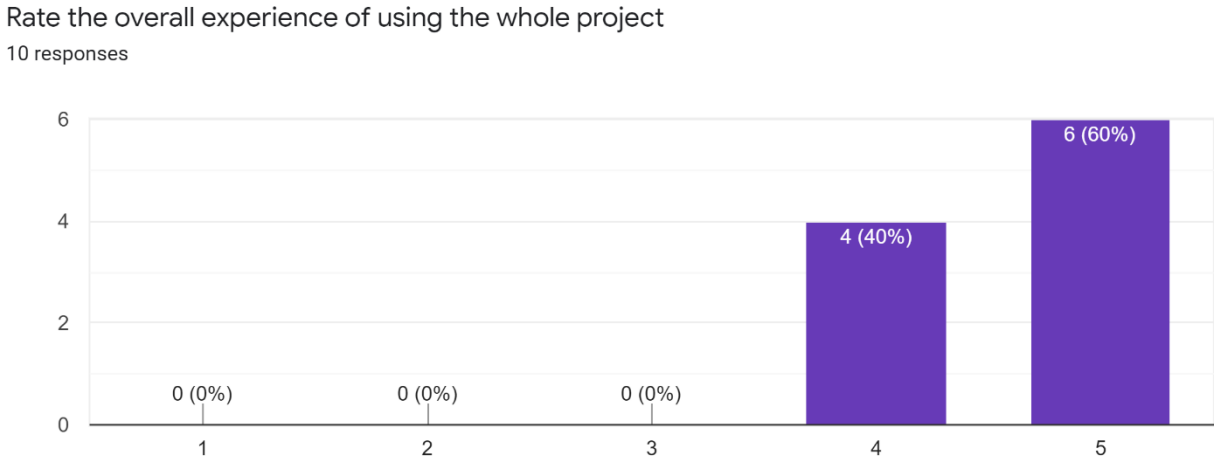


Figure 17 Overall Project feedback

The overall project got a very good response and majority of people liked the idea behind the project.

3.3.2.4 Mobile Application Rating

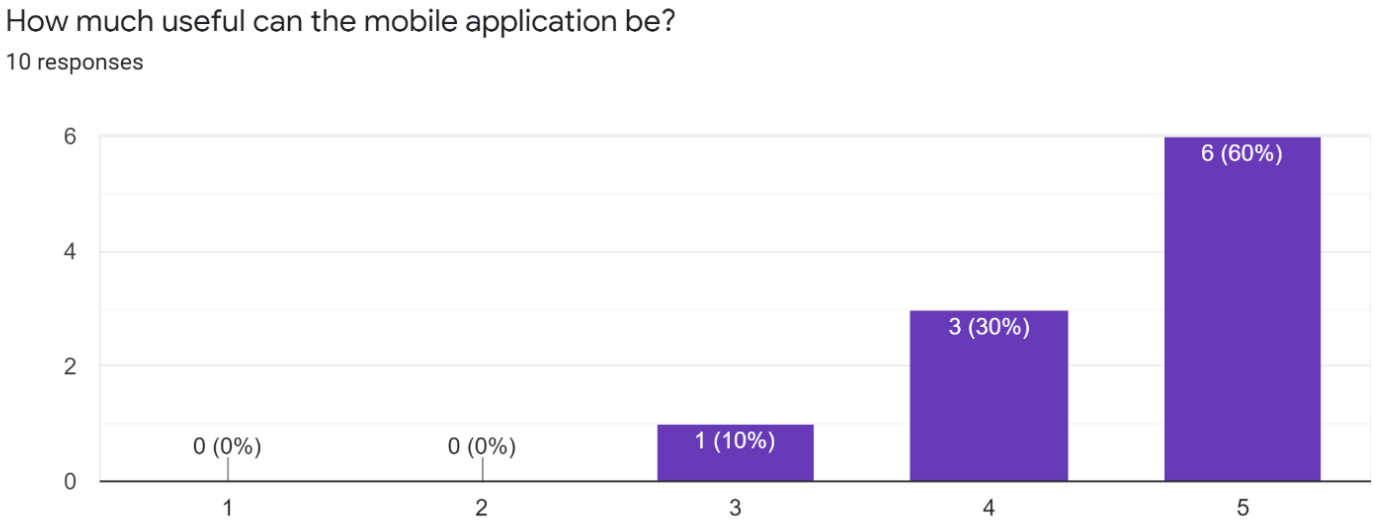


Figure 18 Overall Mobile Application rating

The mobile application was the last moment idea of mine. It wasn't included in the proposal. The mobile application is basically for the drivers. Mobile application also got good response.

3.4 REQUIREMENT ANALYSIS

This project is divided into 3 specific sections. First one is **web application** for the taxi owners, the second one is the **mobile application** and the third one is a **python script** for drowsiness detection. **(SRS Document is also written in the appendix section)**

3.4.1 Web Application

My web application is made by Django. It is not yet deployed in the live server. Things that are needed to run web application are as follows:

- Since, the web application is not deployed in live server it is run locally.
- No internet connection is required.
- The web application runs on any browser.

One of the main tools need to run my web application is Django along with other library. So, a requirement.txt file is created to make the web application run easily.

Figure 19 Requirements.txt

This file directly installs all the library and packages needed for the web application to run.

3.4.2 Mobile Application

My mobile application for now is only tested in android via android emulator. The mobile application is also not live and is run in emulator only. To run the mobile application, following requirements are needed:

- A mobile device or emulator is required which is connected to the IDE through a laptop or PC.
- Flutter must be installed in the laptop or PC.

- Django project should also run in the background to hit the API.

3.4.3 Drowsiness Detection Software

This is a python script which gives drowsiness alert and sends post request to Django URL. This is just a prototype model so this software is made under some limitation. To run this software, following requirements are needed:

- A laptop or desktop is required.
- Flutter must be installed in the device.
- Django project must also be run in the background because the drowsiness detection software sends post request to the Django URL.

3.5 DESIGN

System design is one of the crucial aspects which is compulsory while working under any type of methodologies. Since, I have selected RUP Methodology, all the diagrams below are made according to the methodology requirements. Besides the diagrams below, more designs are included in the Appendix section. **(Click Here)**

3.5.1 Entity Relationship Diagram

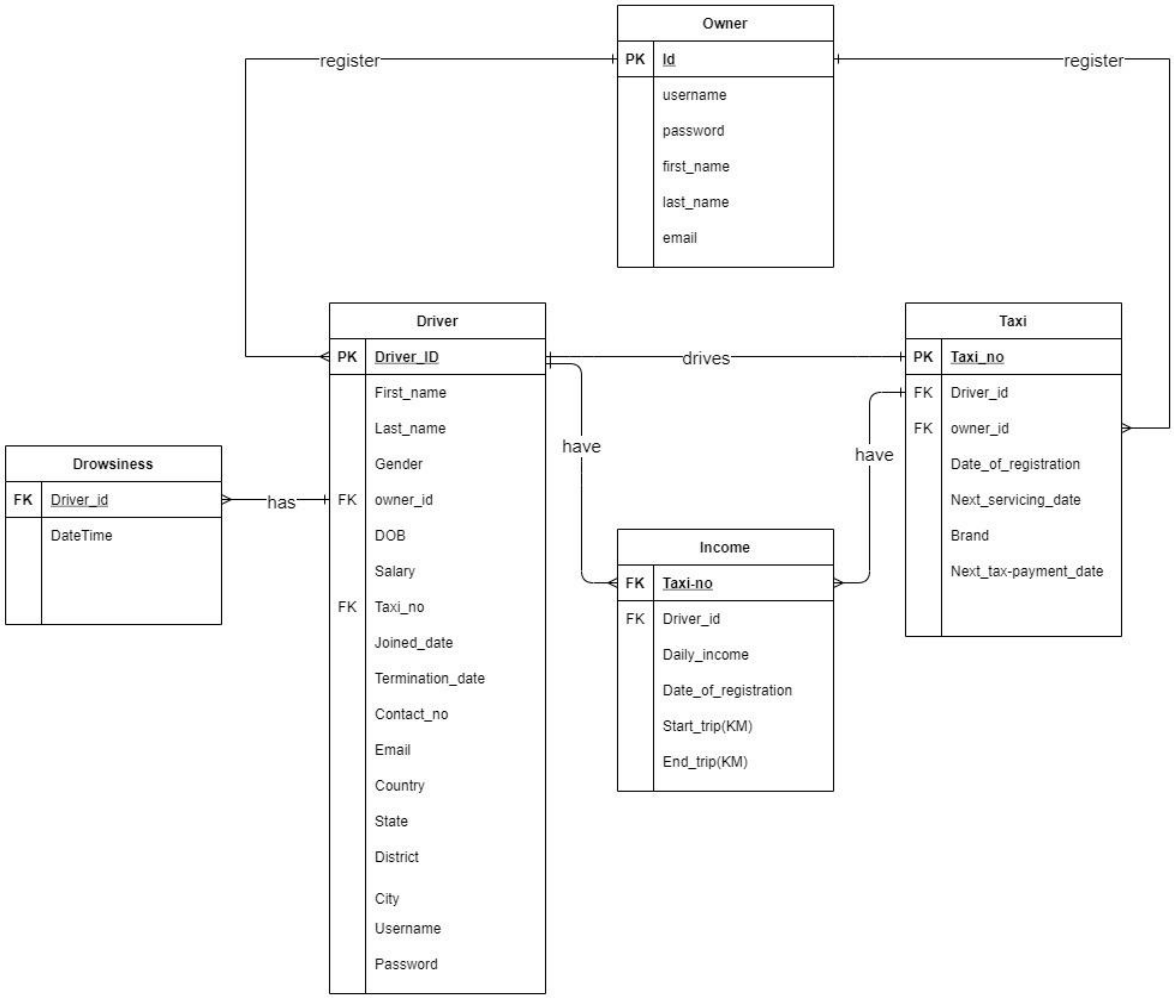


Figure 20 ERD Diagram

The above figure is the ERD diagram of my project. The Owner here basically plays role of admin where the owner can register taxi, driver, and income. Drowsiness table basically contains all the drowsiness data of the driver. There are total of 5 tables.

3.5.2 Use Case Diagram

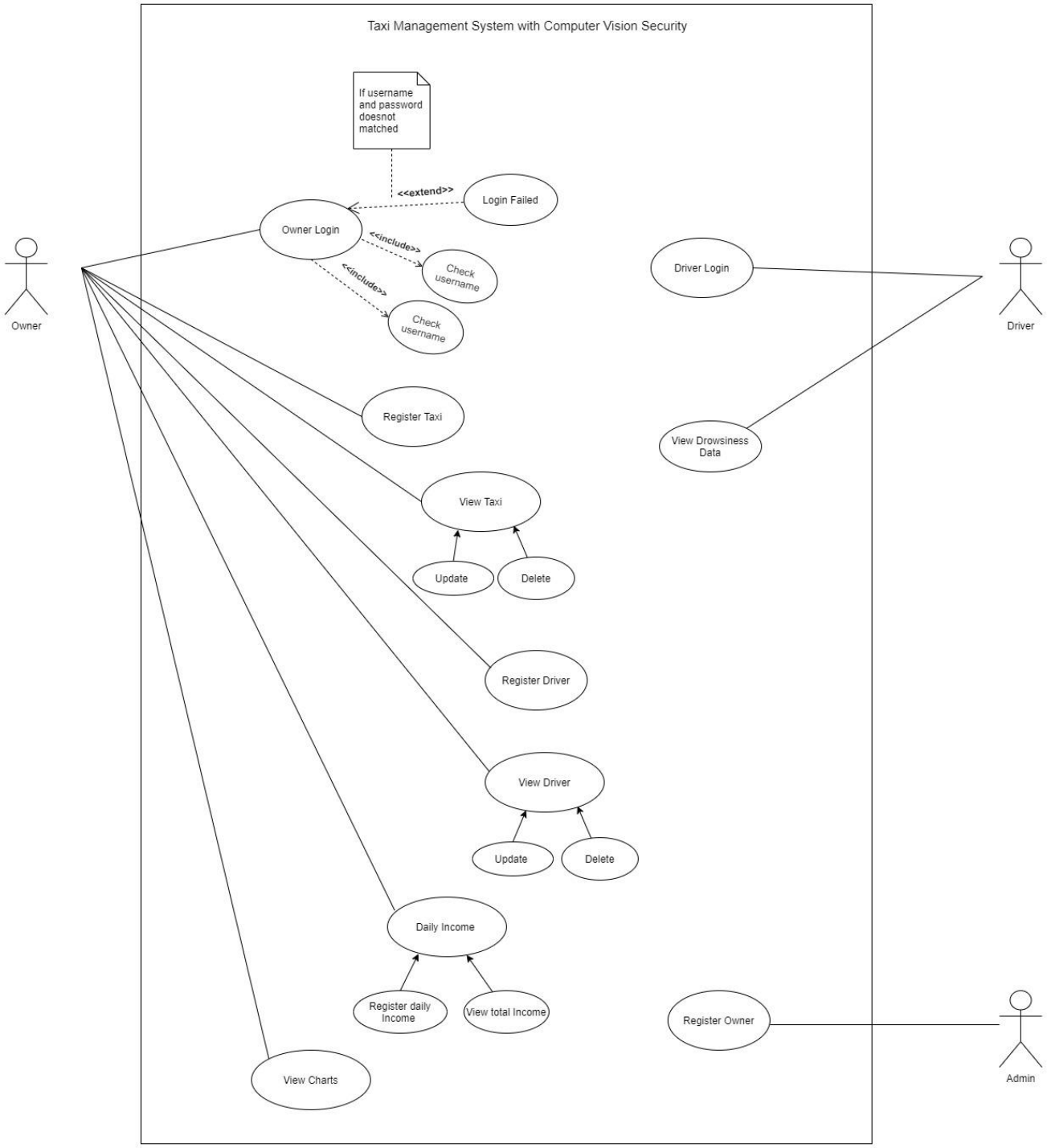


Figure 21 Use Case Diagram

3.5.3 Context Level Diagram

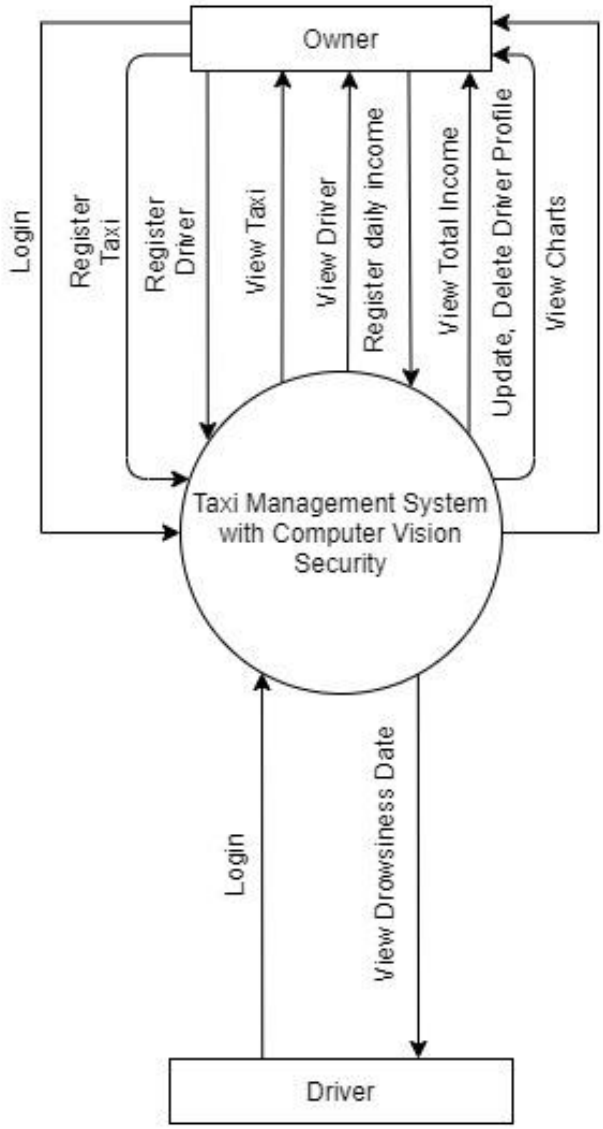


Figure 22 Context Level Diagram

The context level diagram for this project consists of two main external entities (actors); Owner and Driver. The Owner entity provides application with registration details of Driver, taxi, and income. The owner can view chart and do all sorts of CRUD operations. The Driver entity gives drowsiness details of the driver.

3.5.4 Data Flow Diagram

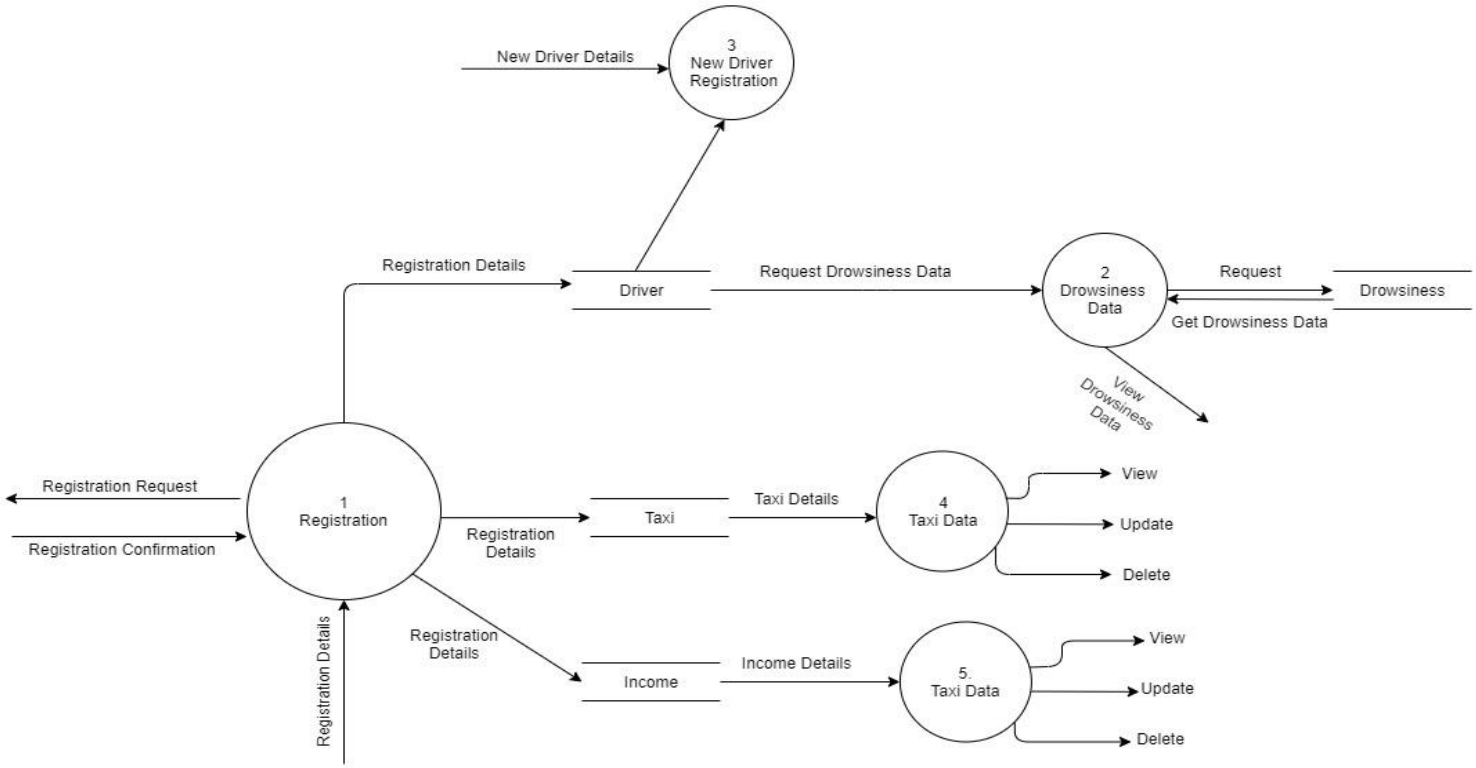


Figure 23 DFD Diagram

DFD (Data Flow Diagram) is a diagram used to graphically represent the flow of data in a business information system (visual-paradigm, 2021).

There are total of 5 processes in the above diagram. First, a registered owner should register Driver, Taxi, and Income to use all the features. After the registration process, driver can view drowsiness data. Also, owner can do CRUD operations with Taxi and Income.

3.5.5 Sequence Diagram

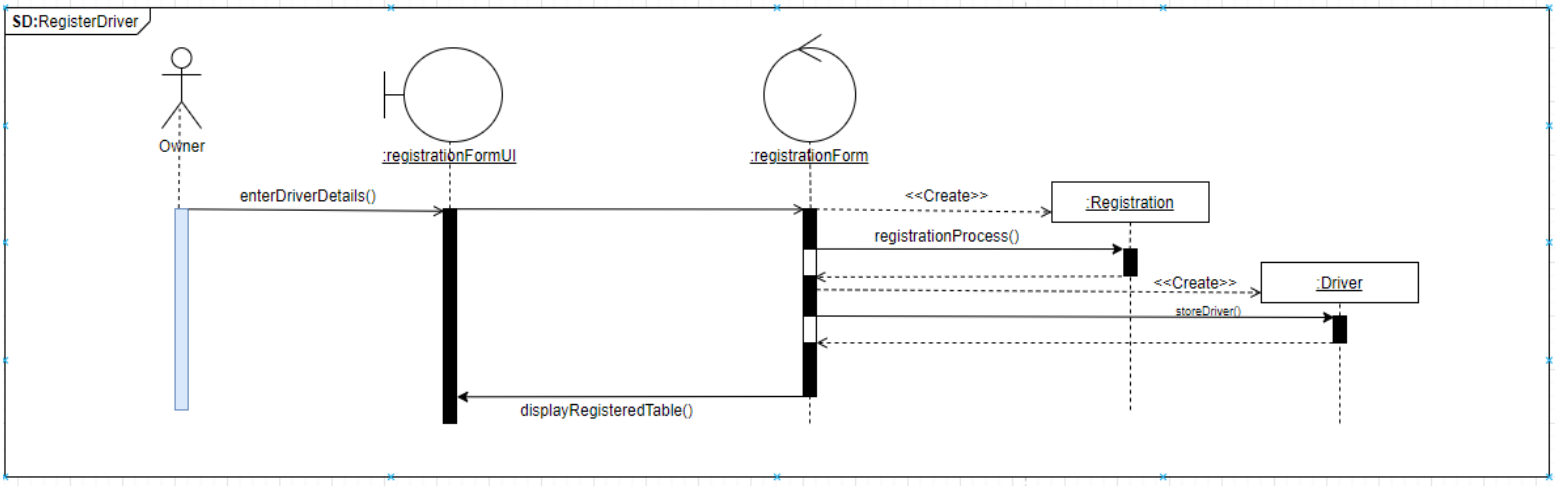


Figure 24 Register Driver

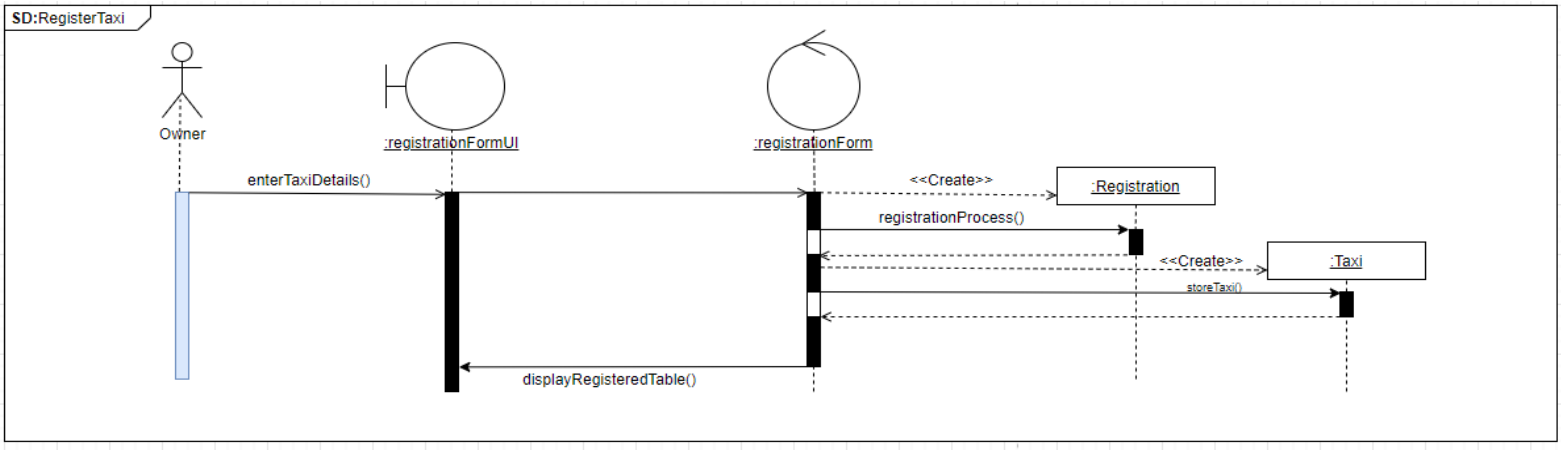


Figure 25 Register Taxi

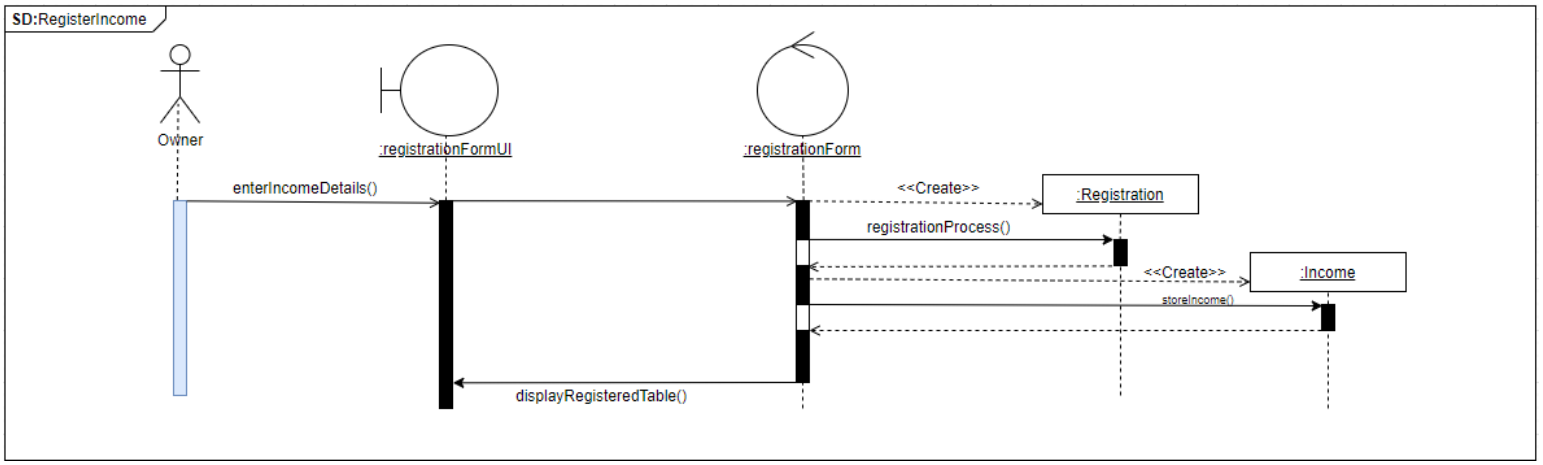


Figure 26 Register Income

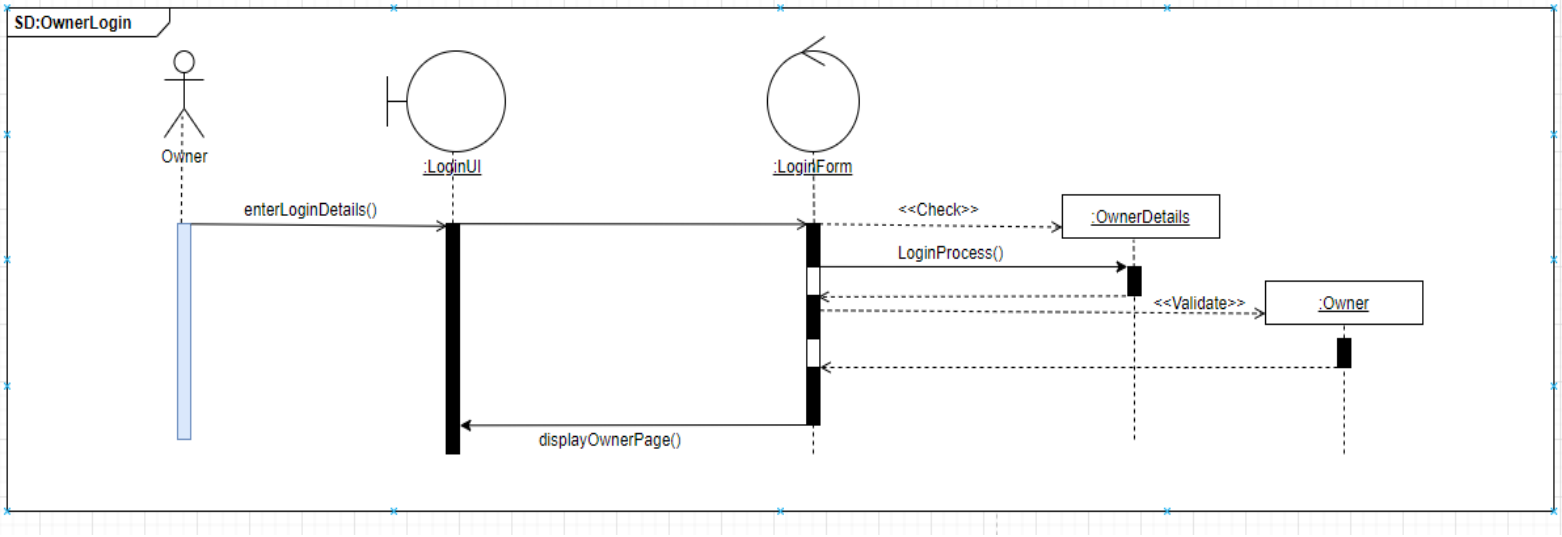


Figure 27 Owner Login

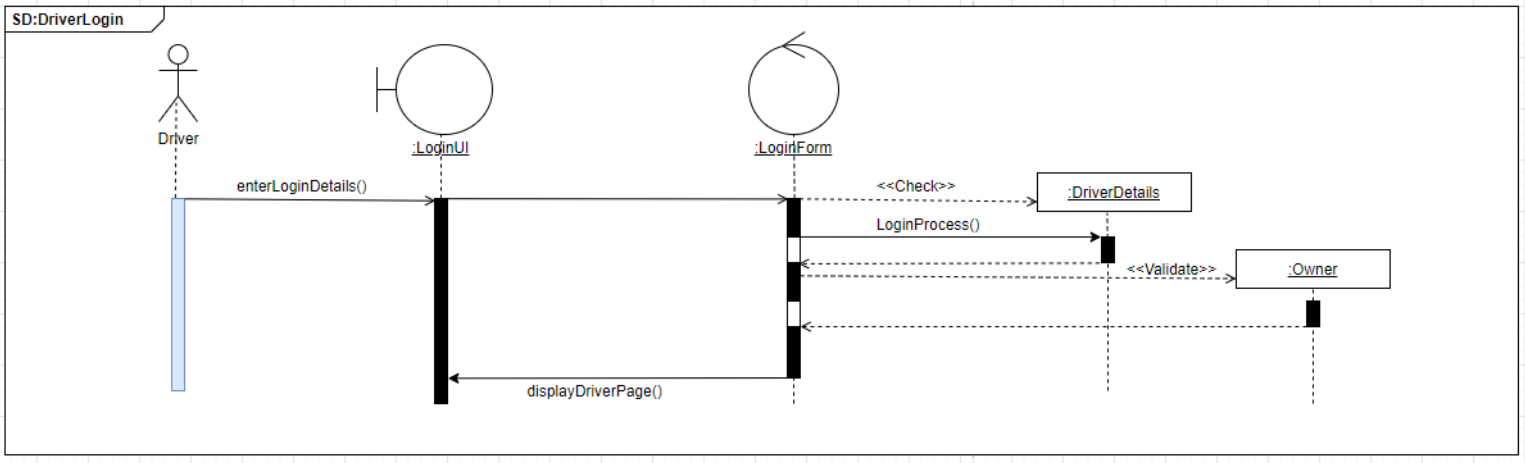


Figure 28 Driver Login

3.5.6 Communication Diagram

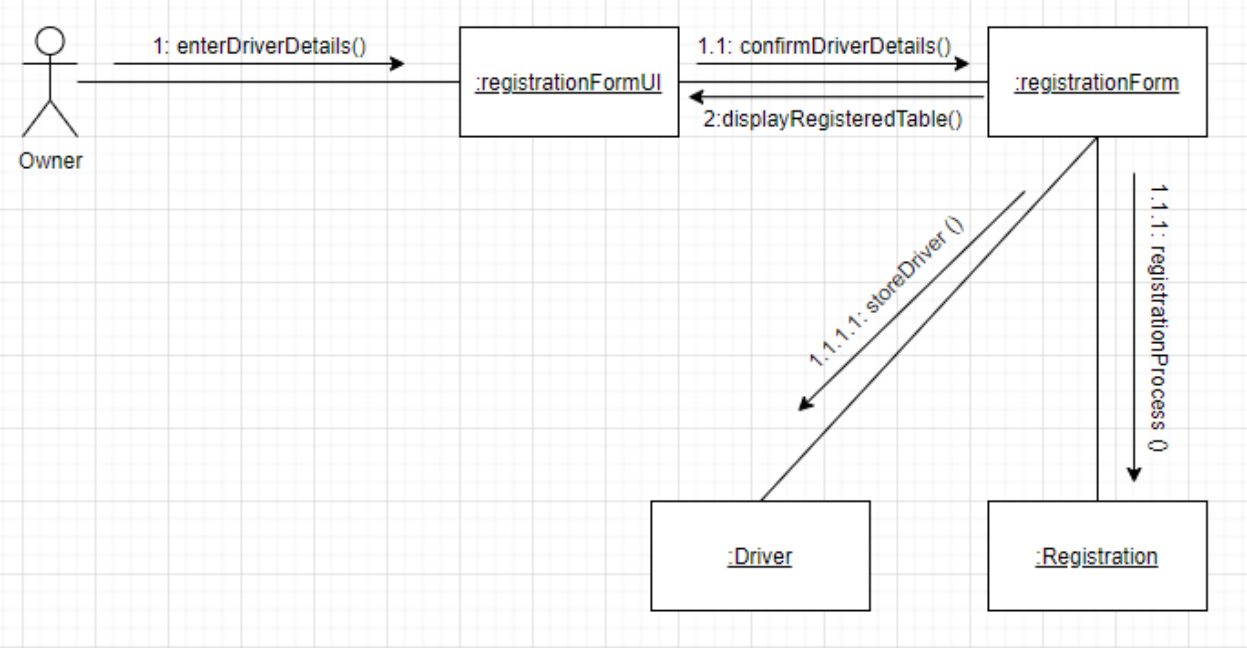


Figure 29 Driver Registration (Communication)

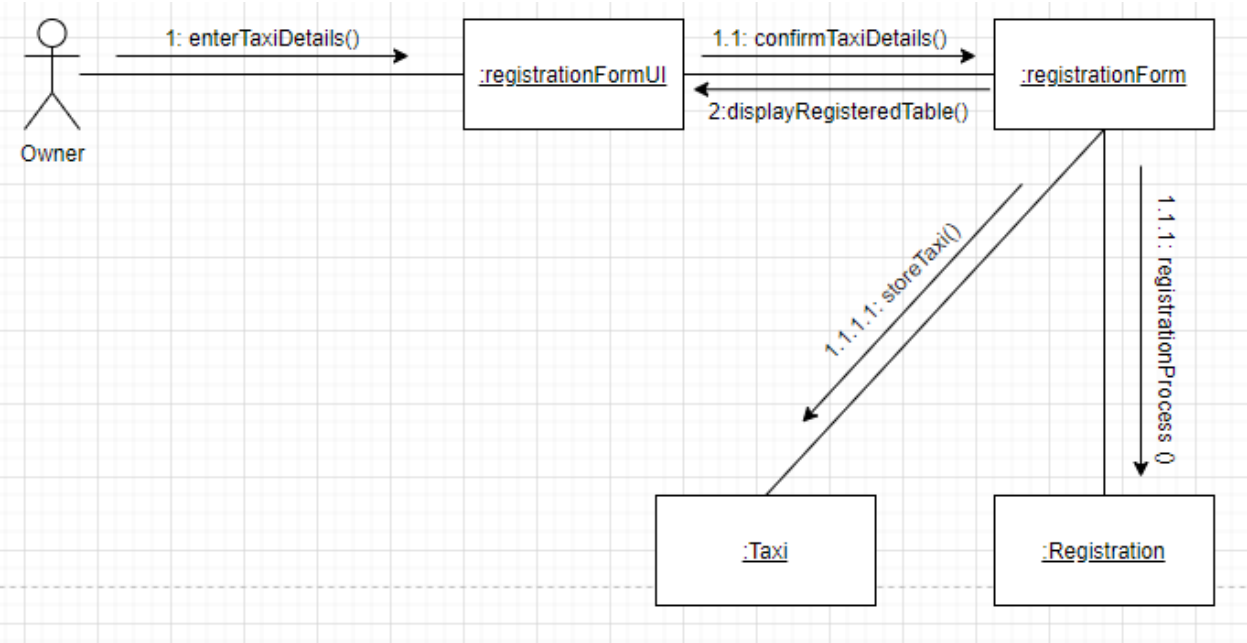


Figure 30 Taxi Registration (Communication)

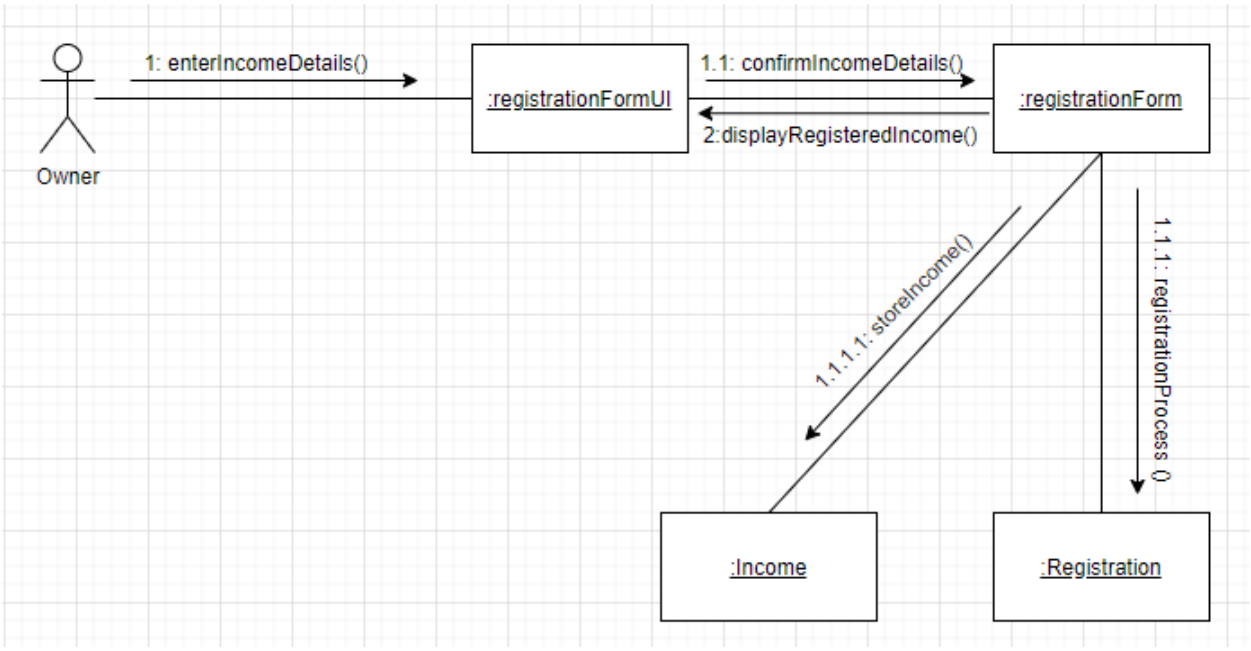


Figure 31 Income Registration (Communication)

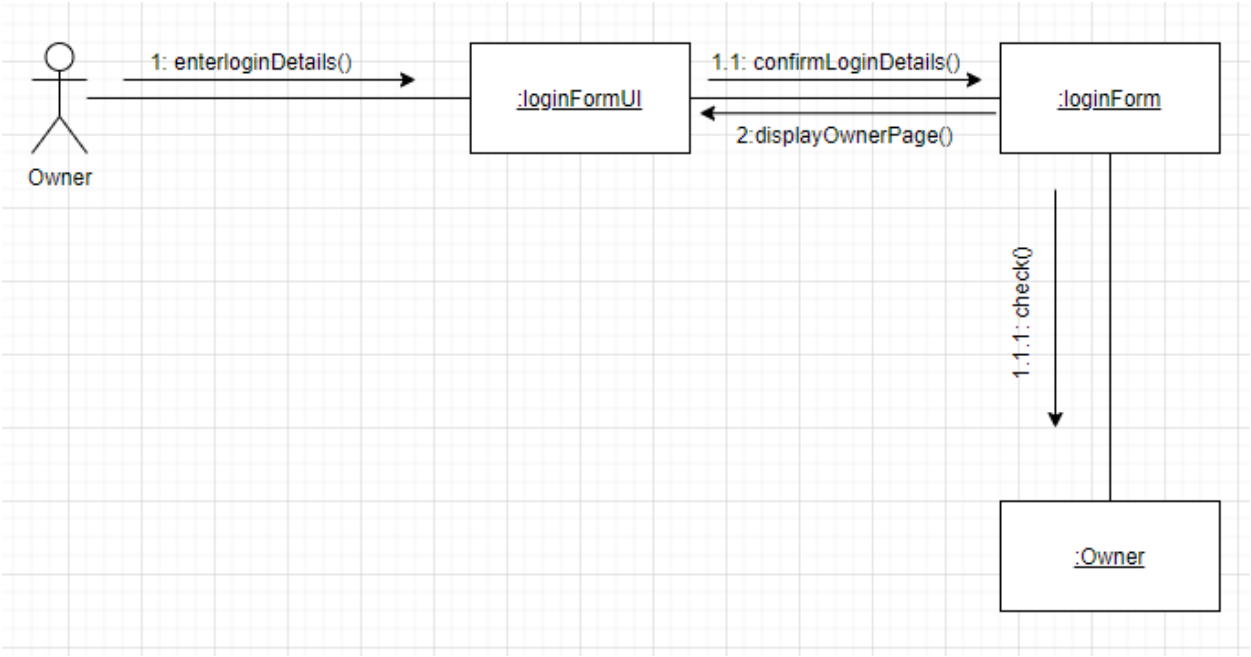


Figure 32 Owner Login (Communication)

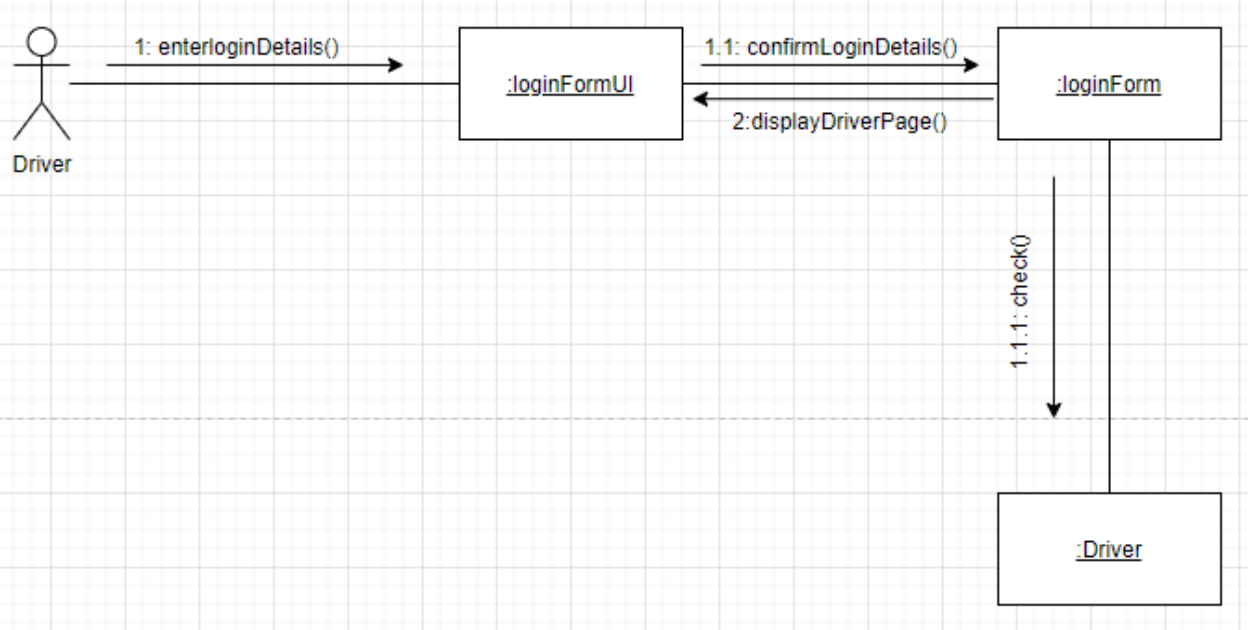


Figure 33 Driver Login (Communication)

3.7 IMPLEMENTATION

All the project related codes are here in Implementation portion, however some extra codes are written in Appendix section below: **(Click Here)**

3.7.1 Iteration 1: User Registration

Firstly, in this project there are 2 types of user one is Owner and the other is driver. Both have their own login page. All the iteration is based on Gantt chart **(Appendix Gantt Chart)**.

3.7.1.1 Database Construction

Taxi_Management_System > Taxi_Manager > models.py > Driver

```python
from django.db import models
from django.contrib.auth.models import User

class Owner(models.Model): #This is the owner table who will use the webapp
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True, )
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)

    def __str__(self):
        return '%s' % (self.first_name)
```

Figure 34 Owner (models.py)

```python
class Driver(models.Model): #All the driver are registered my owner
    GENDER = (
        ('Male', 'Male'),
        ('Female', 'Female'),
        ('Not to Specify', 'Not to Specify'),
    )
    taxi = models.ForeignKey(Taxi, on_delete=models.SET_NULL, null=True)
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True, )
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    gender = models.CharField(max_length=14, choices=GENDER)
    date_of_birth = models.DateField()
    salary = models.IntegerField()
    joined_date = models.DateField()
    termination_date = models.DateField()
    contact_no = models.BigIntegerField(unique=True, error_messages={'unique':"This phone number has already been registered."})
    email = models.EmailField(unique=True, error_messages={'unique':"This email has already been registered."})
    country = models.CharField(max_length=50)
    state = models.CharField(max_length=100)
    district = models.CharField(max_length=150)
    city = models.CharField(max_length=200, null=True)

    def __str__(self):
        return '%s %s' % (self.first_name, self.last_name)
```

Figure 35 Driver (models.py)

The above images are the database table created using Django. Driver has lots of attributes like first_name, last_name, contact_no and so on. In this project, there will be

only one owner but there can be multiple drivers. All the drivers are registered by the owner.

3.7.1.1 Backend

```
@login_required(login_url='/')
@allowed_owner(allowed_roles=['Owners'])
def driver_registration_view(request):
    if request.method == "POST":
        data = request.POST
        User.objects.create_user(username=data['username'],password=data['password1'],first_name=data['first_name'], last_name=data['last_name'])
        user = User.objects.all().last().id
        user = User.objects.get(id=user)
        new_driver = Driver(user = user,first_name=data['first_name'], last_name=data['last_name'],
                            gender=data['gender'], date_of_birth=data['date_of_birth'],
                            #profile_image=data['profile_image'],
                            salary=data['salary'],
                            taxi=Taxi.objects.get(id=int(data['taxi'])),
                            joined_date=data['joined_date'],
                            termination_date=data['termination_date'], contact_no=data['contact_no'],
                            email=data['email'], country=data['country'], state=data['state'],
                            district=data['district'], city=data['city']
                            )
        new_driver.save()
        group = Group.objects.get(name='Drivers')
        user.groups.add(group)
    user_form = UserCreationForm()
    form = DriverRegistrationForm()
    return render(request, 'Taxi_Manager/register_driver.html', {
        'user_form': user_form,
        'form': form,
        })
```

Figure 36 Register Driver (views.py)

```
class DriverList(ListView):
    model = Driver
    template_name = 'Taxi_Manager/view_driver.html'


class ViewDriverDetail(DetailView):
    model = Driver
    template_name = 'Taxi_Manager/view_driver_detail.html'


class DriverUpdateView(UpdateView):
    model = Driver
    fields = ('first_name', 'last_name',
              'gender', 'date_of_birth', 'salary',
              'joined_date', 'termination_date', 'taxi',
              'contact_no', 'email',
              'country', 'state', 'district', 'city',
              )
    template_name = 'Taxi_Manager/update_driver_detail.html'
    def get_success_url(self):
        return reverse('Taxi_Manager:view-driver')

class DriverDelete(DeleteView):
    model = Driver
    template_name = 'Taxi_Manager/delete.html'
    success_url = reverse_lazy('Taxi_Manager:view-driver')
```

Figure 37 Generic views for drivers

```
path('register/driver/', views.driver_registration_view, name='register-driver'),
path('view/driver/', views.DriverList.as_view(), name='view-driver'),
path('view/driver/<int:pk>',views.ViewDriverDetail.as_view(), name='driver-detail'),
path('view/driver/<int:pk>/update/',views.DriverUpdateView.as_view(), name='update-driver'),
path('view/driver/<int:pk>/delete/',views.DriverDelete.as_view(), name='delete-driver'),
```

Figure 38 driver URLs

The above code are used to create driver and store it as a authenticated user. Driver_registration_view register a new driver every time with username and password and creates an authenticated user from it which will be further used in login.

To do basic CRUD operations in Django, I have used default generic views like Updateview, delete view, listview and so on.

3.7.1.1 Frontend

Taxi_Management_System > Taxi_Manager > templates > Taxi_Manager > register_driver.html > div.container-fluid > div.card.mb-12 > div.card-body > form > div.card > div.card-body > div.row > div.col-md-6.mb-3

```
{% extends 'Taxi_Manager/base.html' %}

{% block title %}Driver Registration{% endblock %}

{% load widget_tweaks %}

{% block content %}
<!-- Begin Page Content -->
<div class="container-fluid">

</div>
<!-- /.container-fluid -->
<div class="container-fluid">
    <div class="card mb-12">

        <div class="card-header">
            <h3>
                <center>Driver Registration</center>
            </h3>
        </div>

        <div class="card-body">
            <form method="post" style="margin: 0 20px 0 20px;">
                {% csrf_token %}
                {{ form.non_field_errors }}

                <div class="card" style="margin-top: 20px">
                    <div class="card-header">
                        <h5>
                            <center>Personal Details</center>
                        </h5>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-md-6 mb-3">
                                {{ form.first_name.errors }}
                                <label>First Name:</label>
                                {% render_field form.first_name class="form-control" placeholder=form.first_name.label %}
                            </div>
```

Figure 39 register_driver.html (1)

```
                    <center>Personal Details</center>
                </h5>
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6 mb-3">
                        {{ form.first_name.errors }}
                        <label>First Name:</label>
                        {% render_field form.first_name class="form-control" placeholder=form.first_name.label %}
                    </div>

                    <div class="col-md-6 mb-3">
                        {{ form.last_name.errors }}
                        <label>Last Name:</label>
                        {% render_field form.last_name class="form-control" placeholder=form.last_name.label %}
                    </div>
                </div>

                <div class="row">
                    <div class="col-md-6">
                        {{ form.gender.errors }}
                        <label>{{form.gender.label}}:</label>
                        {{ form.gender|add_class:'form-control' }}
                    </div>
                    <div class="col-md-6">
                        <label>{{form.date_of_birth.label}}:</label>
                        {{ form.date_of_birth|add_class:'form-control' }}
                    </div>
                    <!--<div class="col-md-6">
                        <label>{{form.profile_image.label}}:</label>
                        {% render_field form.profile_image class="form-control" placeholder=form.profile_image.label %}
                    </div>-->
                </div>
            </div>
        </div>
        <div class="card" style="margin-top: 20px">
            <div class="card-header">
                <h5>
                    <center>Job Details</center>
                </h5>
```

Figure 40 register_driver.html (2)

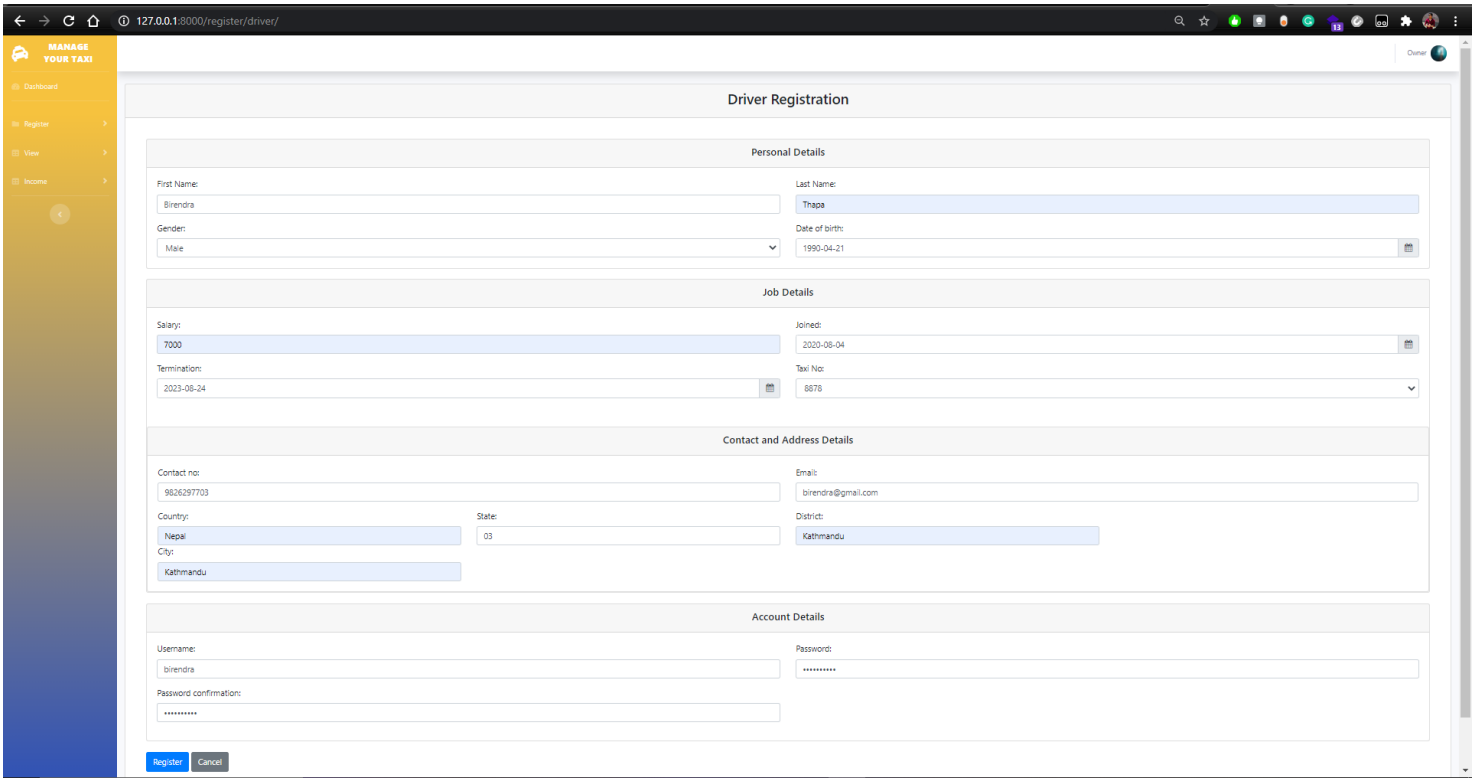
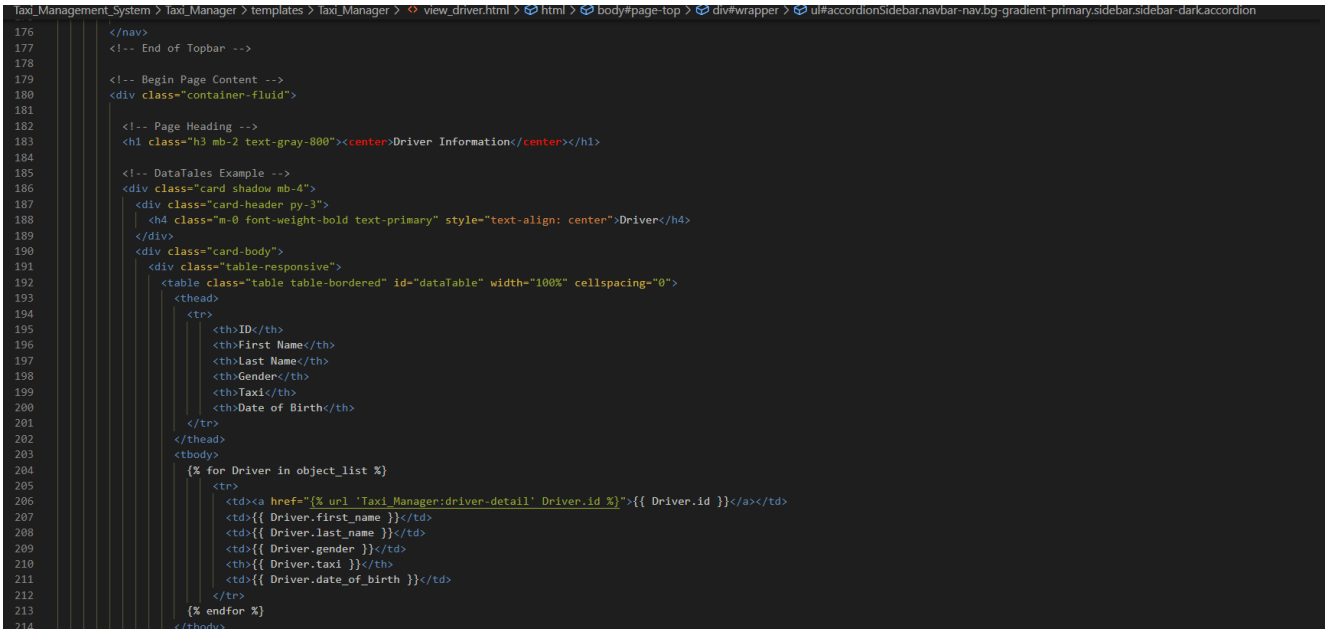


Figure 41 Driver Registration Form

This is the registration form for driver which is only assessed by the owners. Here no field should be left empty else validation is done to prevent from any kind of errors.



```
Taxi_Management_System > Taxi_Manager > templates > Taxi_Manager > view_driver.html > html > body#page-top > div#wrapper > ul#accordionSidebar.navbar-nav.bg-gradient-primary.sidebar.sidebar-dark.accordion
176         </nav>
177         <!-- End of Topbar -->
178
179         <!-- Begin Page Content -->
180         <div class="container-fluid">
181
182           <!-- Page Heading -->
183           <h1 class="h3 mb-2 text-gray-800"><center>Driver Information</center></h1>
184
185           <!-- DataTales Example -->
186           <div class="card shadow mb-4">
187             <div class="card-header py-3">
188               <h4 class="m-0 font-weight-bold text-primary" style="text-align: center">Driver</h4>
189             </div>
190             <div class="card-body">
191               <div class="table-responsive">
192                 <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
193                   <thead>
194                     <tr>
195                       <th>ID</th>
196                       <th>First Name</th>
197                       <th>Last Name</th>
198                       <th>Gender</th>
199                       <th>Taxi</th>
200                       <th>Date of Birth</th>
201                     </tr>
202                   </thead>
203                   <tbody>
204                     {% for Driver in object_list %}
205                       <tr>
206                         <td><a href="{% url 'Taxi_Manager:driver-detail' Driver.id %}">{{ Driver.id }}</a></td>
207                         <td>{{ Driver.first_name }}</td>
208                         <td>{{ Driver.last_name }}</td>
209                         <td>{{ Driver.gender }}</td>
210                         <th>{{ Driver.taxi }}</th>
211                         <td>{{ Driver.date_of_birth }}</td>
212                       </tr>
213                     {% endfor %}
214                   </tbody>
```

Figure 42 view_driver.html

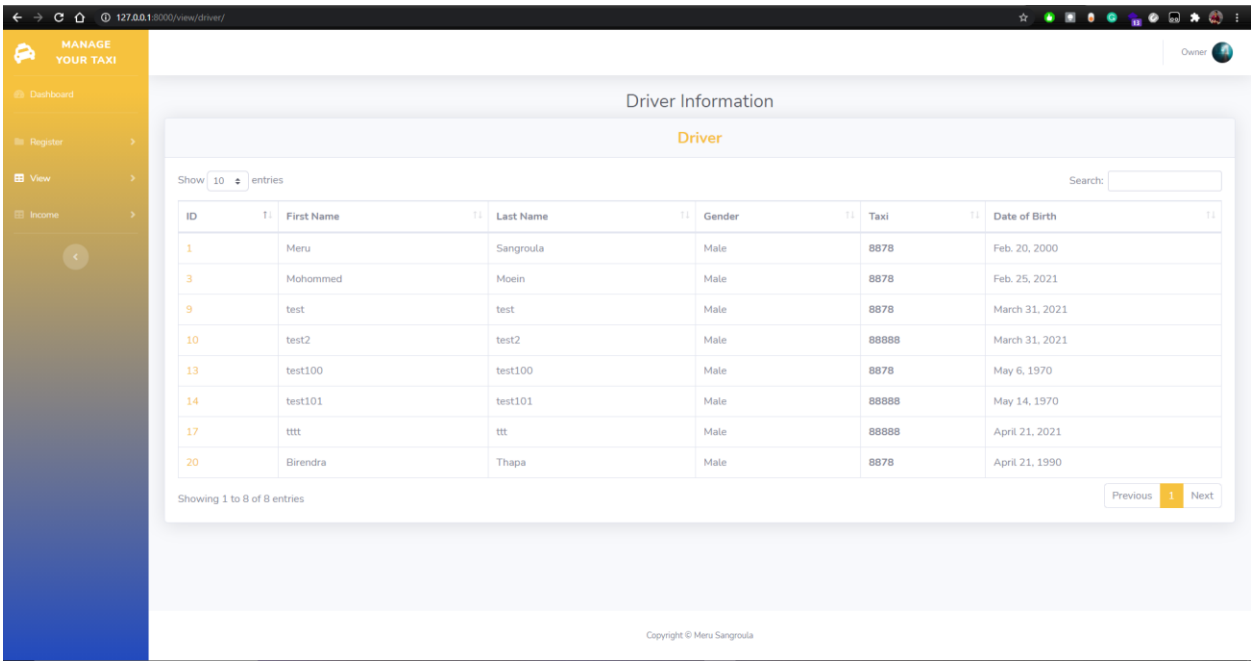
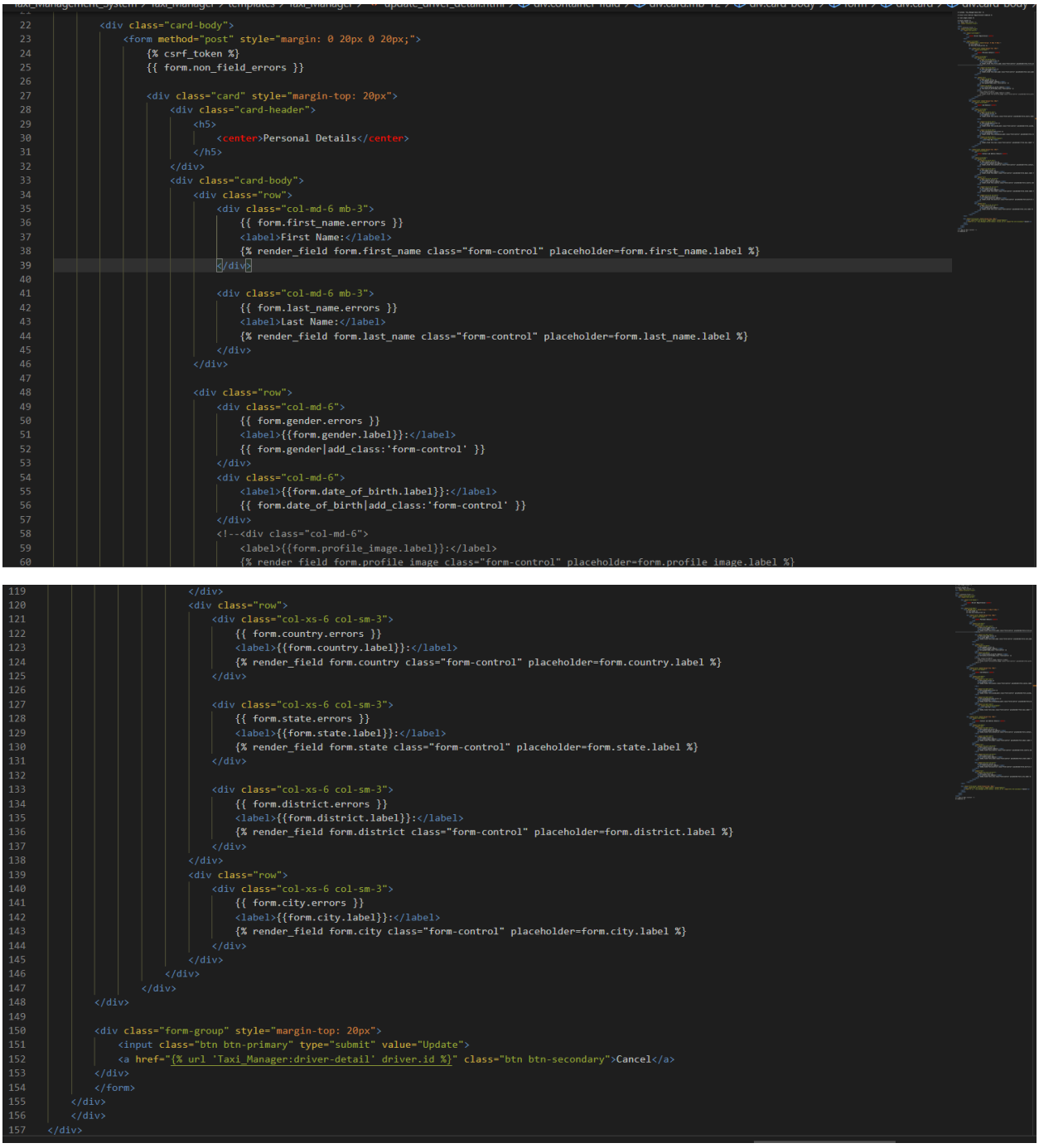


Figure 43 View Driver



```
<div class="card-body">
    <form method="post" style="margin: 0 20px 0 20px;">
        {% csrf_token %}
        {{ form.non_field_errors }}

        <div class="card" style="margin-top: 20px">
            <div class="card-header">
                <h5>
                    <center>Personal Details</center>
                </h5>
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6 mb-3">
                        {{ form.first_name.errors }}
                        <label>First Name:</label>
                        {% render_field form.first_name class="form-control" placeholder=form.first_name.label %}
                    </div>

                    <div class="col-md-6 mb-3">
                        {{ form.last_name.errors }}
                        <label>Last Name:</label>
                        {% render_field form.last_name class="form-control" placeholder=form.last_name.label %}
                    </div>
                </div>

                <div class="row">
                    <div class="col-md-6">
                        {{ form.gender.errors }}
                        <label>{{form.gender.label}}:</label>
                        {{ form.gender|add_class:'form-control' }}
                    </div>
                    <div class="col-md-6">
                        <label>{{form.date_of_birth.label}}:</label>
                        {{ form.date_of_birth|add_class:'form-control' }}
                    </div>
                    <!--<div class="col-md-6">
                        <label>{{form.profile_image.label}}:</label>
                        {% render_field form.profile_image class="form-control" placeholder=form.profile_image.label %}
```

```
                </div>
                <div class="row">
                    <div class="col-xs-6 col-sm-3">
                        {{ form.country.errors }}
                        <label>{{form.country.label}}:</label>
                        {% render_field form.country class="form-control" placeholder=form.country.label %}
                    </div>

                    <div class="col-xs-6 col-sm-3">
                        {{ form.state.errors }}
                        <label>{{form.state.label}}:</label>
                        {% render_field form.state class="form-control" placeholder=form.state.label %}
                    </div>

                    <div class="col-xs-6 col-sm-3">
                        {{ form.district.errors }}
                        <label>{{form.district.label}}:</label>
                        {% render_field form.district class="form-control" placeholder=form.district.label %}
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-6 col-sm-3">
                        {{ form.city.errors }}
                        <label>{{form.city.label}}:</label>
                        {% render_field form.city class="form-control" placeholder=form.city.label %}
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="form-group" style="margin-top: 20px">
        <input class="btn btn-primary" type="submit" value="Update">
        <a href="{% url 'Taxi_Manager:driver-detail' driver.id %}" class="btn btn-secondary">Cancel</a>
    </div>
    </form>
</div>
</div>
</div>
```

Figure 44 update_driver_detail.html

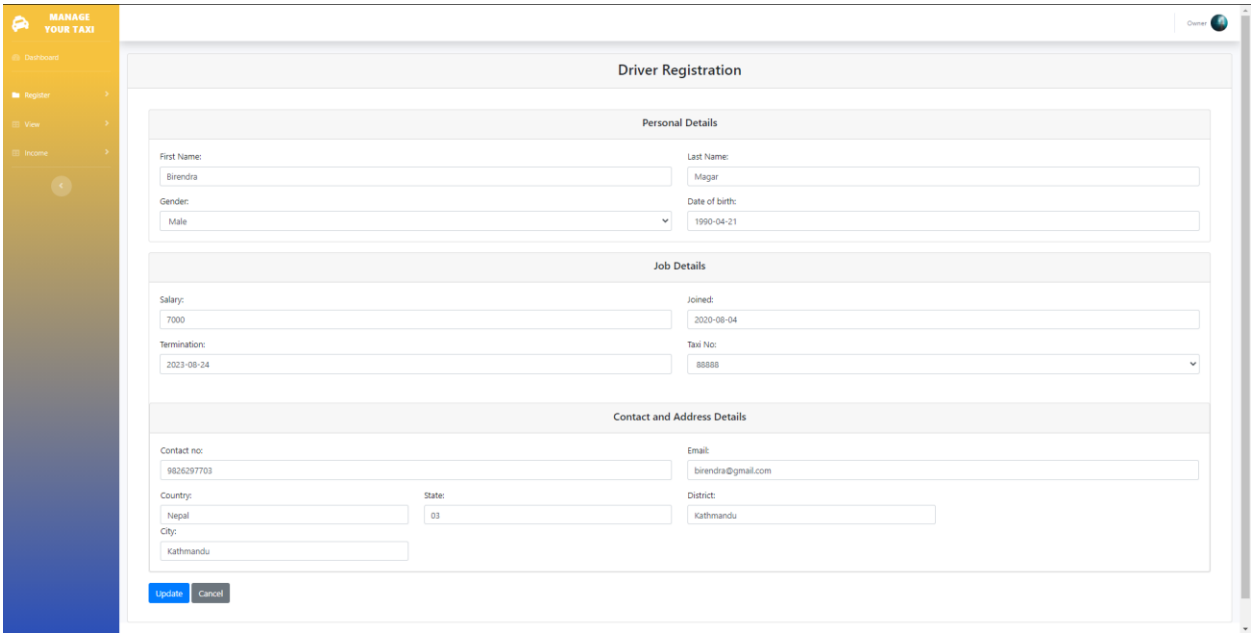


Figure 45 Update Driver Page

```
{% extends 'Taxi_Manager/base.html' %}

{% block title %}Delete{% endblock %}

{% block content %}
    <div style="margin-top: 50px">
        <h2>Deletion Confirmation</h2>

        <form method="post">{% csrf_token %}
            <p>Are you sure you want to delete "{{ object }}"?</p>
            <input type="submit" value="Confirm">
        </form>
    </div>

{% endblock %}
```

Figure 46 delete.html

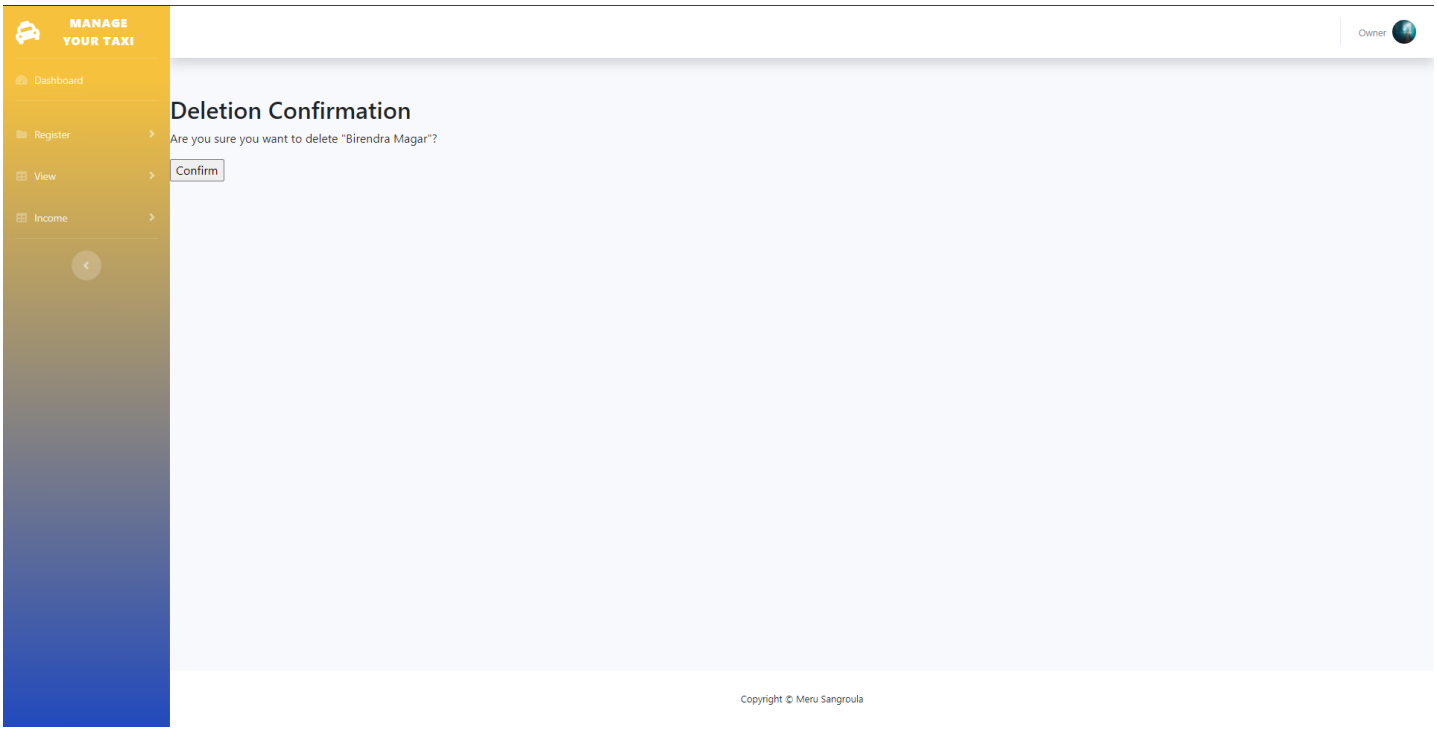


Figure 47 Delete Page

These are the code and frontend design of my web app. These designs do CRUD operation for the driver.

3.7.2 Iteration 2: Taxi Registration

All the CRUD operation of taxi is done under this iteration. Taxi has a foreign key relationship with driver.

3.7.2.1 Database Construction

```
class Taxi(models.Model): #All the taxi are registered my owner
    taxi_no = models.CharField(max_length=10)
    registration_date = models.DateField()
    next_servicing_date = models.DateField()
    brand = models.CharField(max_length=50)
    next_tax_payment_date = models.DateField()

    def __str__(self):
        return '%s' % (self.taxi_no)
```

Figure 48 Taxi (models.py)

The taxi models contains taxi_no, registration_date, brand and so on.

3.7.2.2 Backend

```
@login_required(login_url='/')
@allowed_owner(allowed_roles=['Owners'])
def taxi_registration_view(request):
    if request.method == "POST":
        form = TaxiRegistrationForm(request.POST)
        if form.is_valid():
            data = form.cleaned_data
            new_taxi = Taxi(taxi_no=data['taxi_no'],registration_date=data['registration_date'],
                            next_servicing_date=data['next_servicing_date'],
                            brand=data['brand'], next_tax_payment_date=data['next_tax_payment_date']
                            )
            new_taxi.save()
            return redirect('Taxi_Manager:view-taxi')
    else:
        form = TaxiRegistrationForm()
    return render(request,'Taxi_Manager/register_taxi.html',{'form': form})
```

Figure 49 taxi_registration_view(views.py)

```
class TaxiList(ListView):
    model = Taxi
    template_name = 'Taxi_Manager/view_taxi.html'

class TaxiDetailView(DetailView):
    model = Taxi
    template_name = 'Taxi_Manager/view_taxi_detail.html'

class TaxiUpdateView(UpdateView):
    model = Taxi
    fields = ('taxi_no', 'registration_date',
              'next_servicing_date', 'brand',
              'next_tax_payment_date',
              )
    template_name = 'Taxi_Manager/update_taxi_detail.html'
    def get_success_url(self):
        return reverse('Taxi_Manager:view-taxi')

class TaxiDelete(DeleteView):
    model = Taxi
    template_name = 'Taxi_Manager/delete.html'
    success_url = reverse_lazy('Taxi_Manager:view-taxi')
```

Figure 50 taxi registration (generic views)

```
path('register/taxi/', views.taxi_registration_view, name='register-taxi'),
path('view/taxi/', views.TaxiList.as_view(), name='view-taxi'),
path('view/taxi/<int:pk>/',views.TaxiDetailView.as_view(), name='taxi-detail'),
path('view/taxi/<int:pk>/update/',views.TaxiUpdateView.as_view(), name='update-taxi'),
path('view/taxi/<int:pk>/delete/',views.TaxiDelete.as_view(), name='delete-taxi'),
```

Figure 51 taxi URLs

The above taxi_registration_view is the view to register new taxi by the owner. Only the owner can do all the CRUD operations related to taxi registration. Generic views are used in taxi registration as well.

3.7.2.3 Frontend

Taxi_Management_System > Taxi_Manager > templates > Taxi_Manager > register_taxi.html > div.container-fluid > div.card.mb-12 > div.card-body > form > div.card > div.card-body > div

```
{% extends 'Taxi_Manager/base.html' %}

{% block title %}Taxi Registration{% endblock %}

{% load widget_tweaks %}

{% block content %}
<!-- Begin Page Content -->
<!-- /.container-fluid -->
<div class="container-fluid">
    <div class="card mb-12">

        <div class="card-header">
            <h3>
                <center>Taxi Registration</center>
            </h3>
        </div>

        <div class="card-body">
            <form method="post" style="margin: 0 20px 0 20px;">
                {% csrf_token %}
                {{ form.non_field_errors }}

                <div class="card" style="margin-top: 20px">
                    <div class="card-header"><h5><center>Taxi Details</center></h5></div>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-md-6 mb-3">
                                {{ form.taxi_no.errors }}
                                <label>Taxi no:</label>
                                {% render_field form.taxi_no class="form-control" placeholder=form.taxi_no.label %}
                            </div>
                            <div class="col-md-6 mb-3">
                                {{ form.brand.errors }}
                                <label>Brand:</label>
                                {% render_field form.brand class="form-control" placeholder=form.brand.label %}
                            </div>
                            <div class="col-md-6 mb-3">
                                {{ form.registration_date.errors }}
```

```
                        {% render_field form.taxi_no class="form-control" placeholder=form.taxi_no.label %}
                    </div>
                    <div class="col-md-6 mb-3">
                        {{ form.brand.errors }}
                        <label>Brand:</label>
                        {% render_field form.brand class="form-control" placeholder=form.brand.label %}
                    </div>
                    <div class="col-md-6 mb-3">
                        {{ form.registration_date.errors }}
                        <label>Date of Registration:</label>
                        {% render_field form.registration_date class="form-control" placeholder=form.registration_date.label %}
                    </div>
                    <div class="col-md-6 mb-3">
                        {{ form.next_servicing_date.errors }}
                        <label>Next Servicing Date:</label>
                        {% render_field form.next_servicing_date class="form-control" placeholder=form.next_servicing_date.label %}
                    </div>
                    <div class="col-md-6 mb-3">
                        {{ form.next_tax_payment_date.errors }}
                        <label>Next Tax Payment Date:</label>
                        {% render_field form.next_tax_payment_date class="form-control" placeholder=form.next_tax_payment_date.label %}
                    </div>

                </div>
            </div>
        </div>
        <div class="form-group" style="margin-top: 20px">
            <input class="btn btn-primary" type="submit" value="Register">
            <a href="{% url 'Taxi_Manager:home' %}" class="btn btn-secondary">Cancel</a>
        </div>
    </form>
</div>
</div>
</div>
<!-- End of Main Content -->
{% endblock %}
```

Figure 52 register_taxi.html

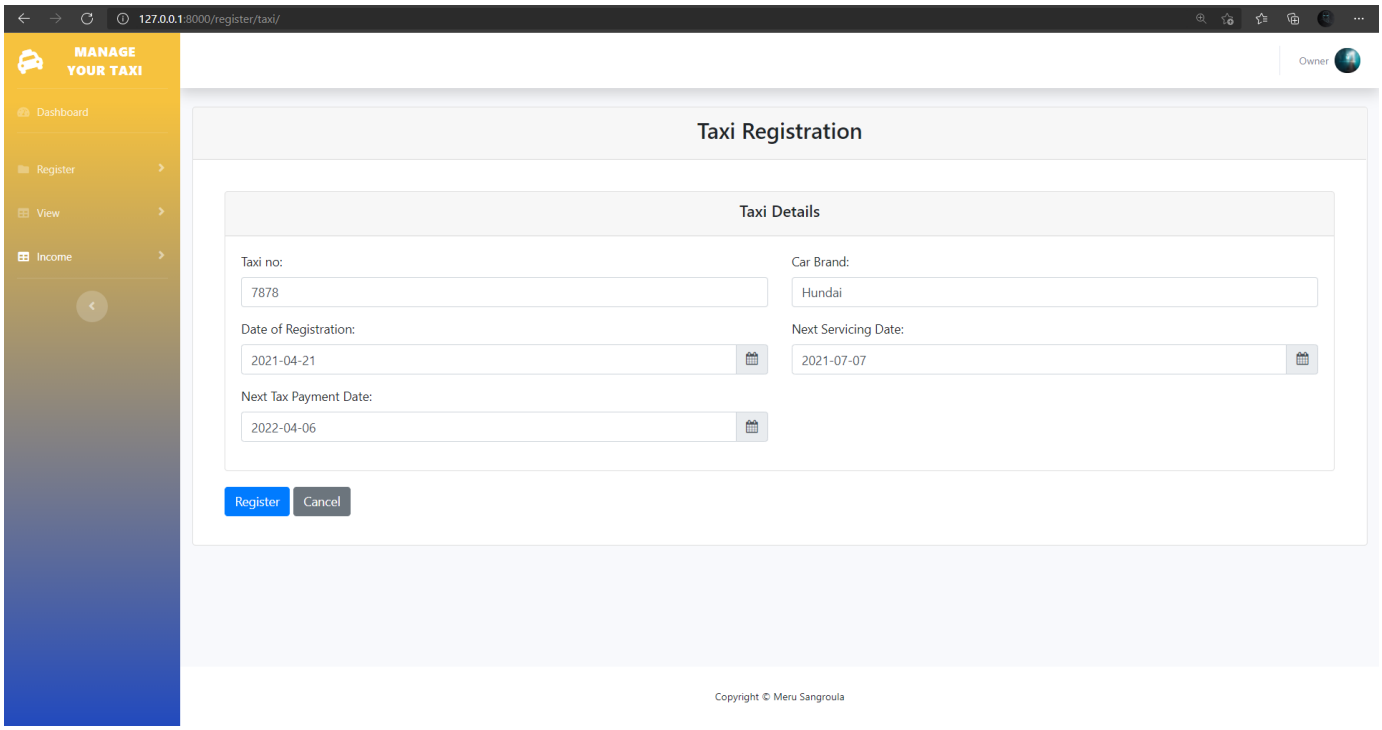


Figure 53 Taxi registration page

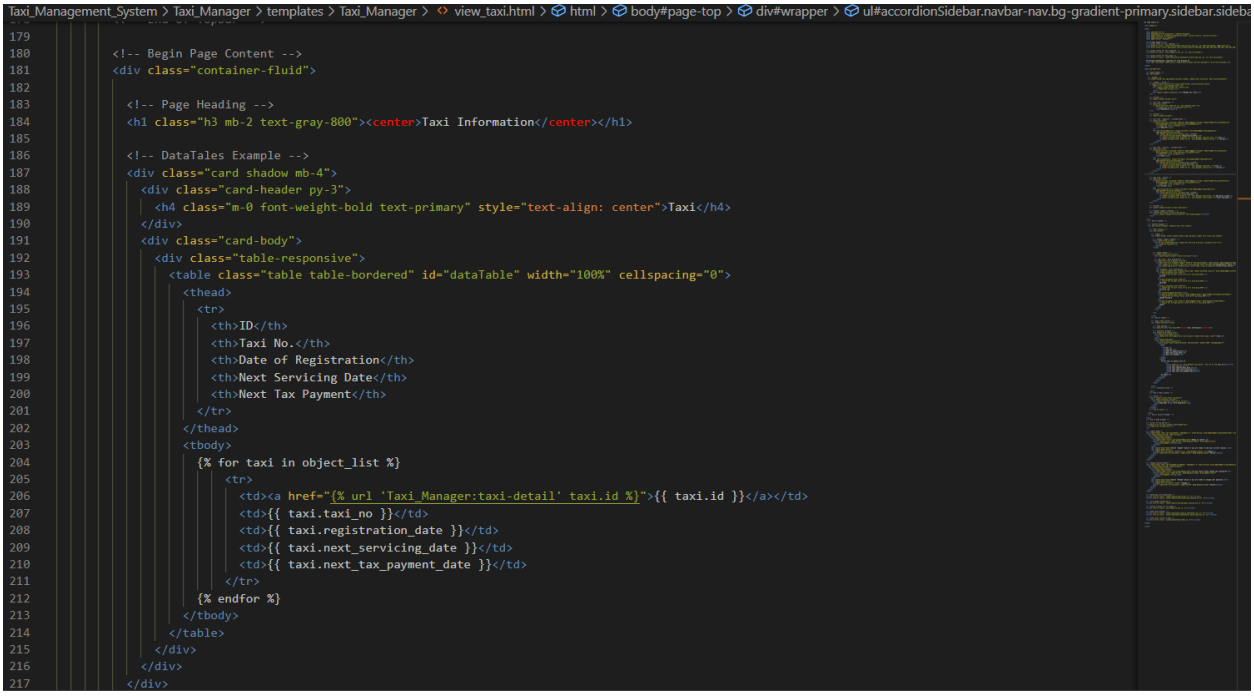


Figure 54 view_taxi.html

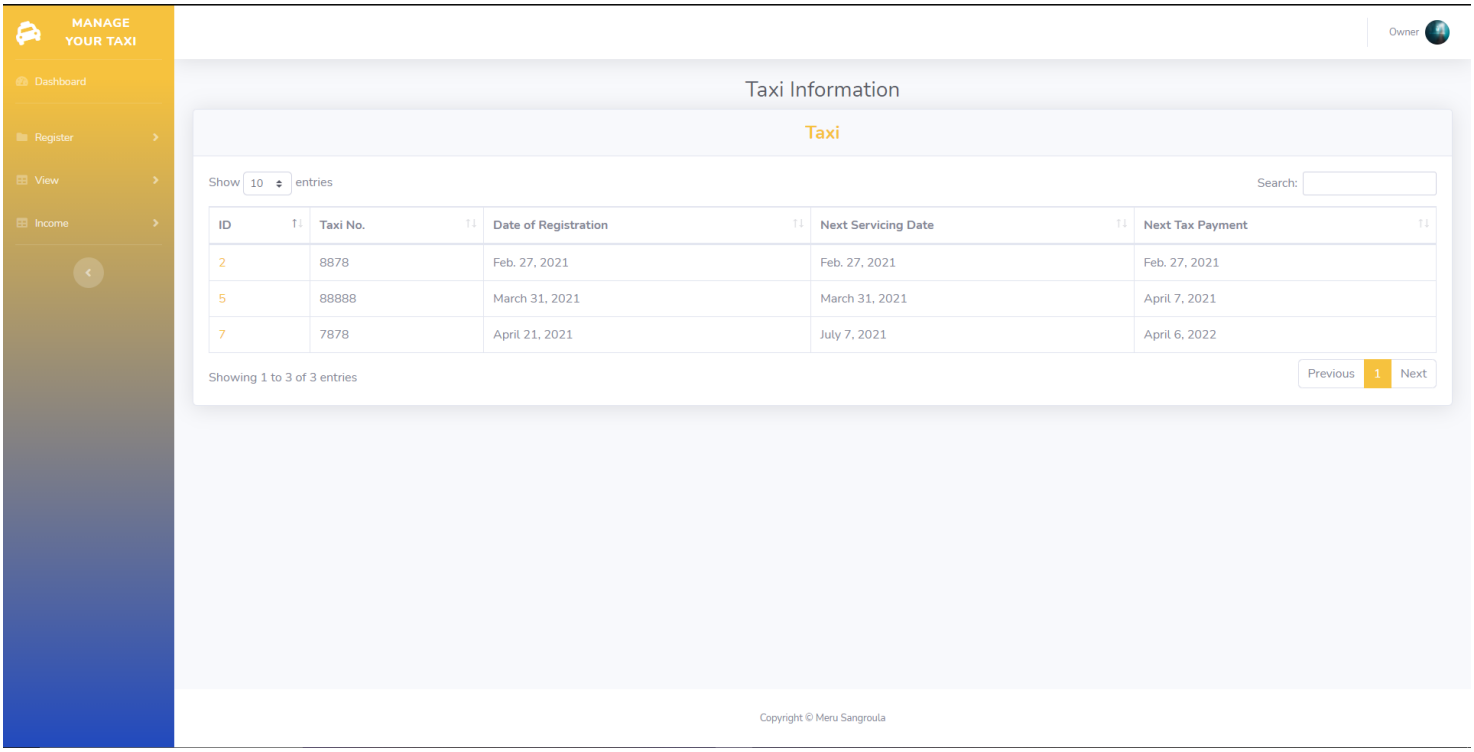


Figure 55 View Taxi page

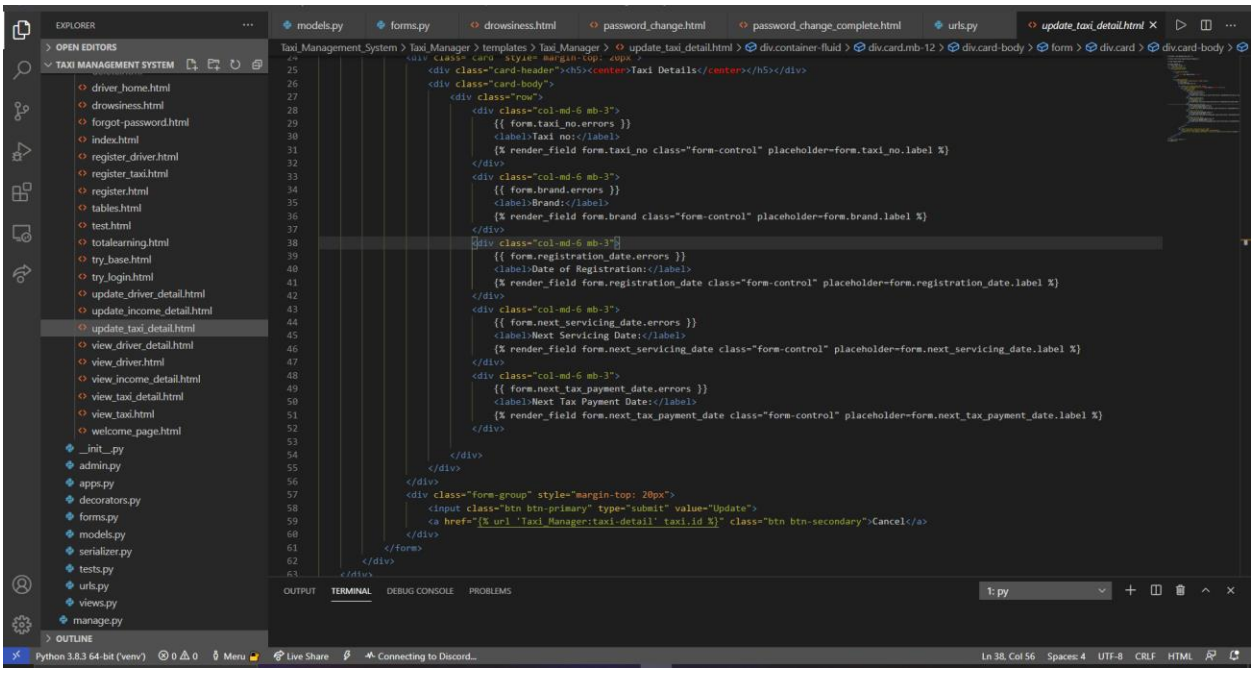


Figure 56 update_taxi_detail.html

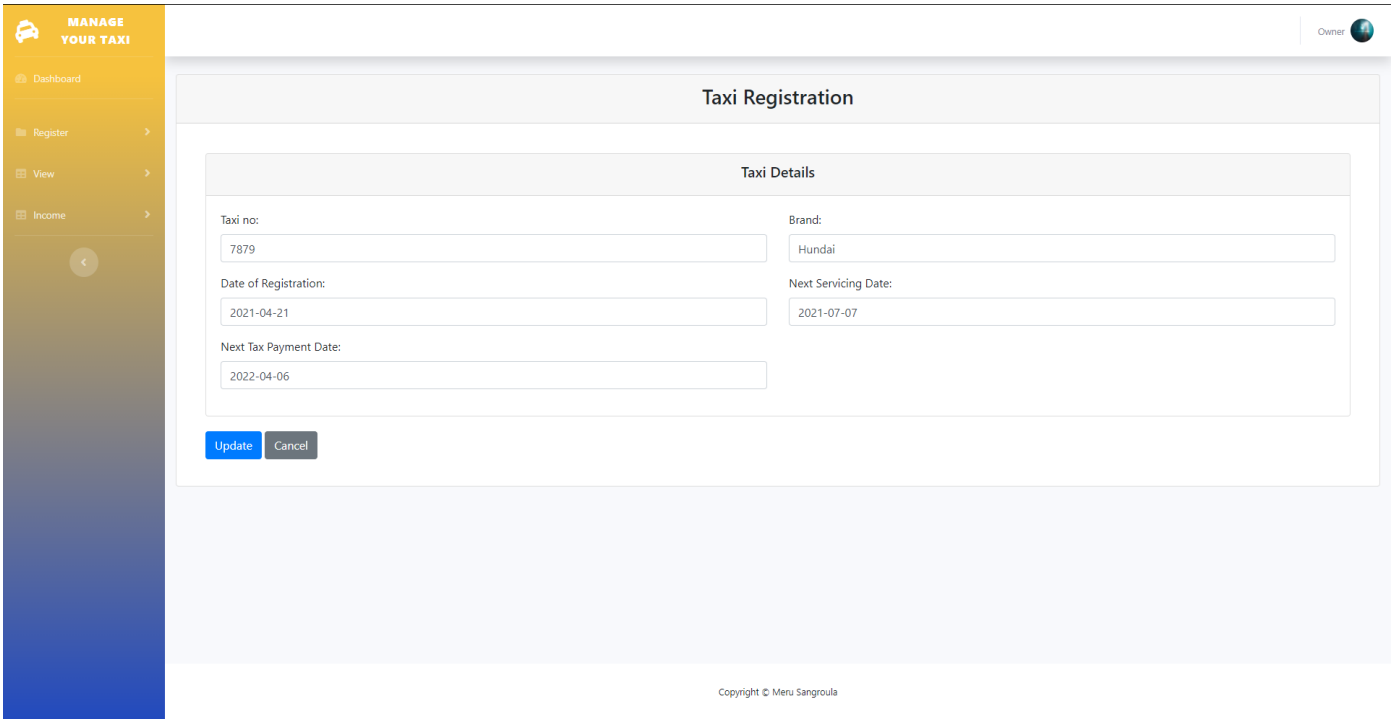


Figure 57 Update taxi page

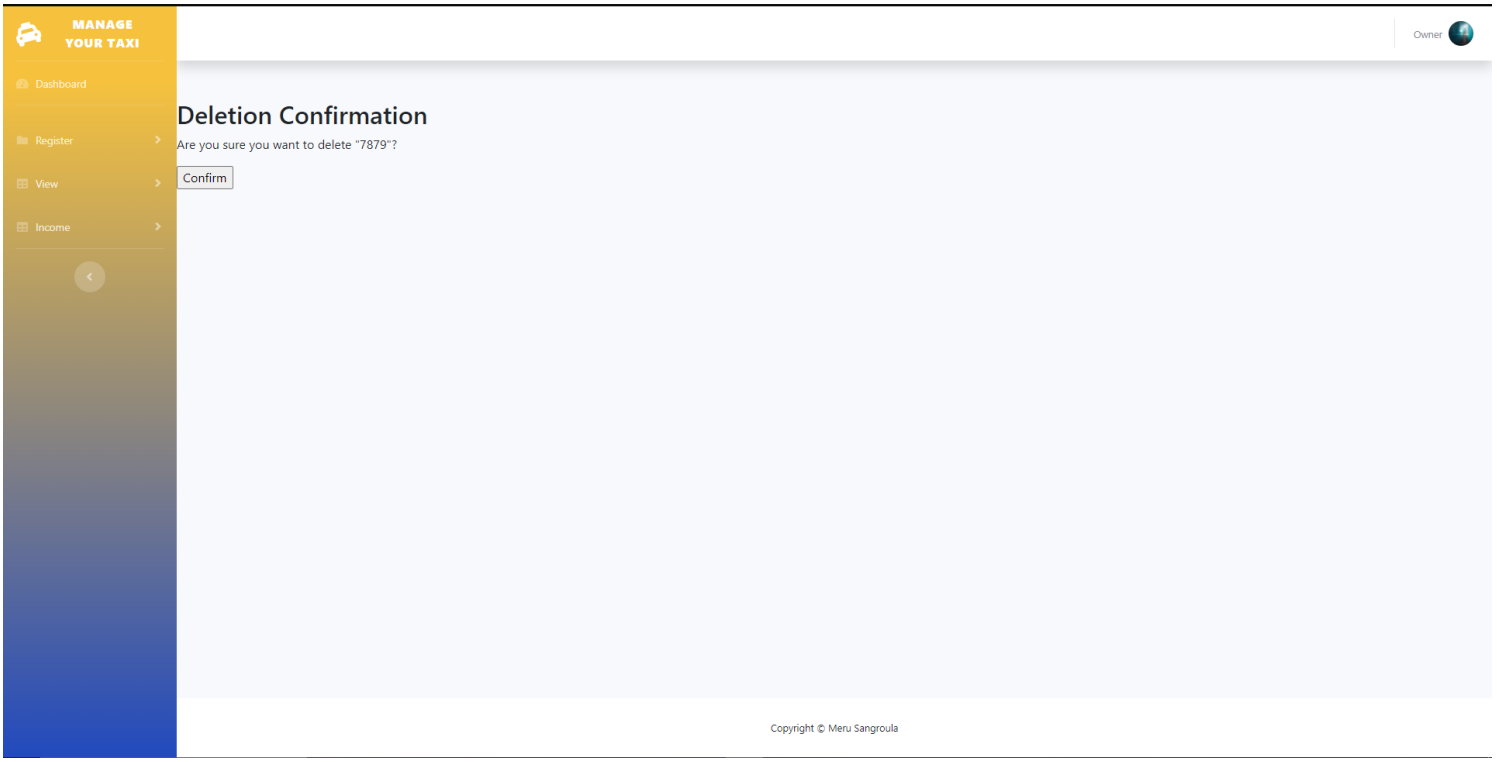


Figure 58 Delete Taxi Page

Template is used for the frontend and all the above pages are for the CRUD operations.

3.7.3 Iteration 3: Drowsiness Software

3.7.3.1 Codes

drowsiness.py 1 | api.http | csv_read.py | apifetch.py

drowsiness.py > ...

```python
#python drowniness_yawn.py --webcam webcam_index

from operator import index, is_not
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
from playsound import playsound
import time
import dlib
import cv2
import os
from datetime import datetime
import pandas
import requests

# def alarm(msg):
#     global alarm_status

#     while alarm_status:
#         print('call')
#         s = 'espeak "'+msg+'"'
#         os.system(s)

test_time = None

times=[]

df=pandas.DataFrame(columns=["Drowsiness"])

def sound_alarm():
    playsound.playsound("D:\3rd Year\FYP\Drowsiness Detection\alarm.wav")



def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
```

Figure 59 drowsiness.py (1)

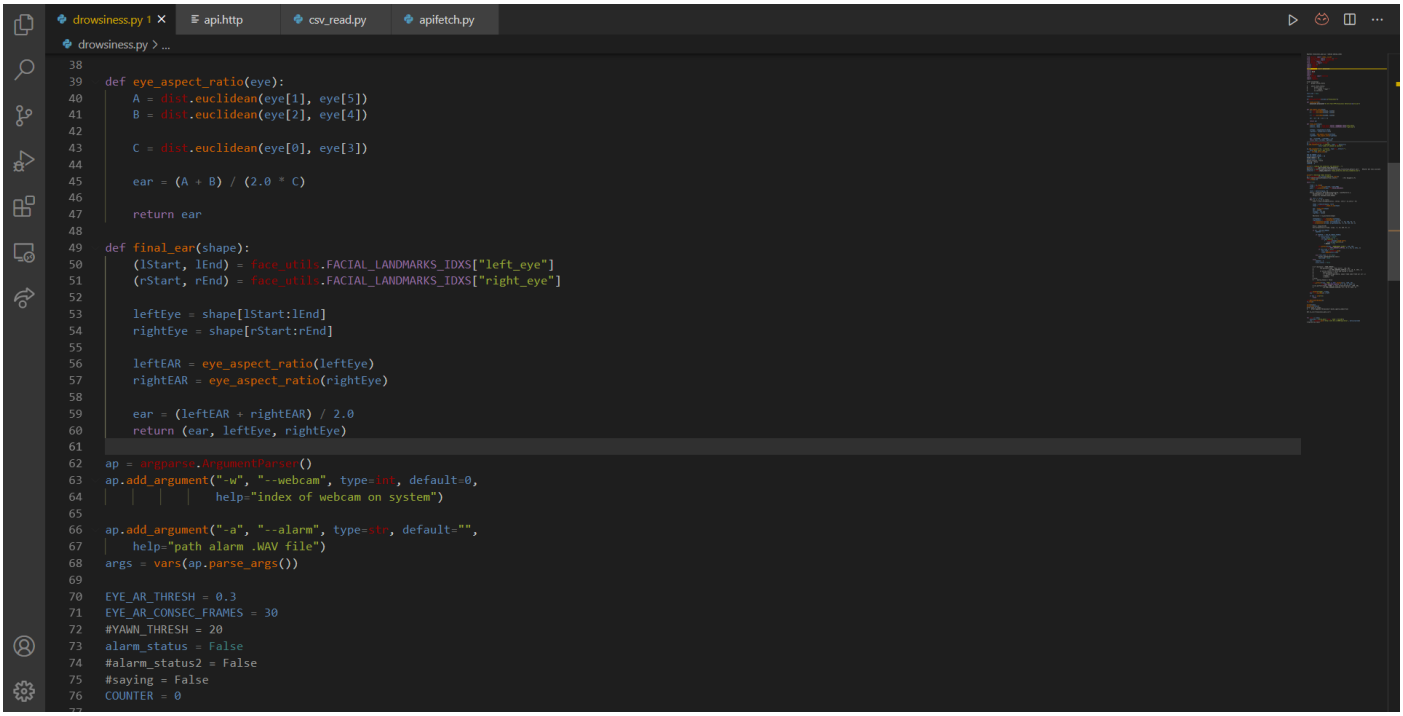


Figure 60 drowsiness.py (2)

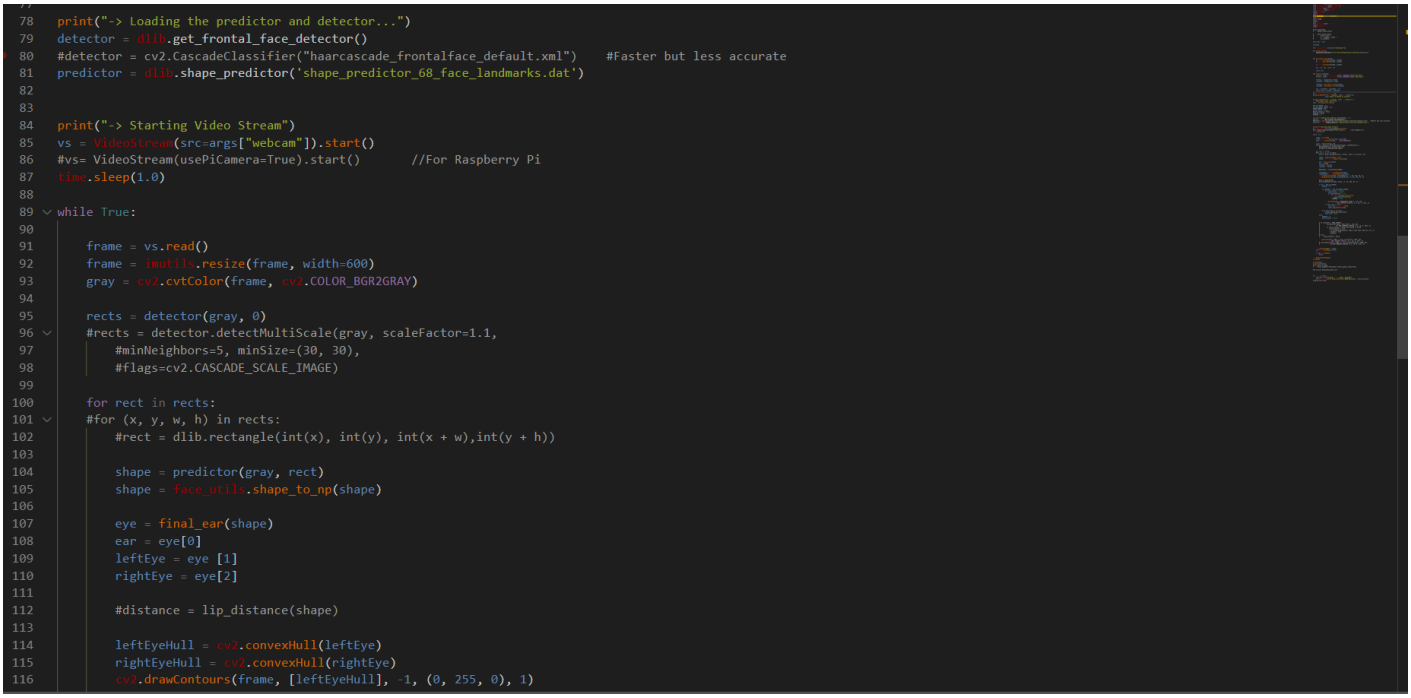


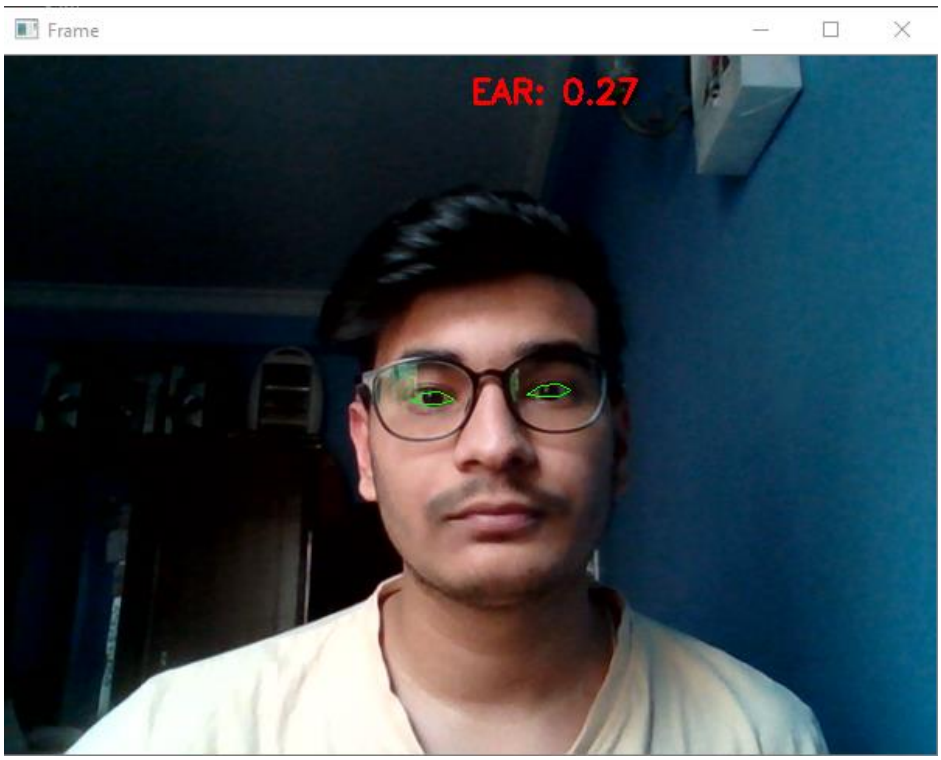
Figure 61 drowsiness.py (3)

```
                    alarm_status = True
                    if args["alarm"] != "":
                        t = Thread(target=sound_alarm,
                            args=(args["alarm"],))
                        t.deamon = True

                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                if test_time is None:
                    test_time = datetime.now()
                    times.append(test_time)

            elif test_time is not None:
                #times.append(datetime.now())
                test_time = None
        else:
            COUNTER = 0
            alarm_status = False


        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)



    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()

print(times)
print(len(times))

for time in times:
    playload = {'drow_data':time, 'name':'test100'}
    res = requests.post('http://127.0.0.1:8000/api/drow/', data=playload)
```

Figure 62 drowsiness.py (4)

For drowsiness software, Open CV is used. Open CV is a computer vision library. It makes the detection process much easier cause it has already got trained AI. From Line no. 164 API is being created and it a post request is send to the Django local host URL.

3.7.3.2 Implementation



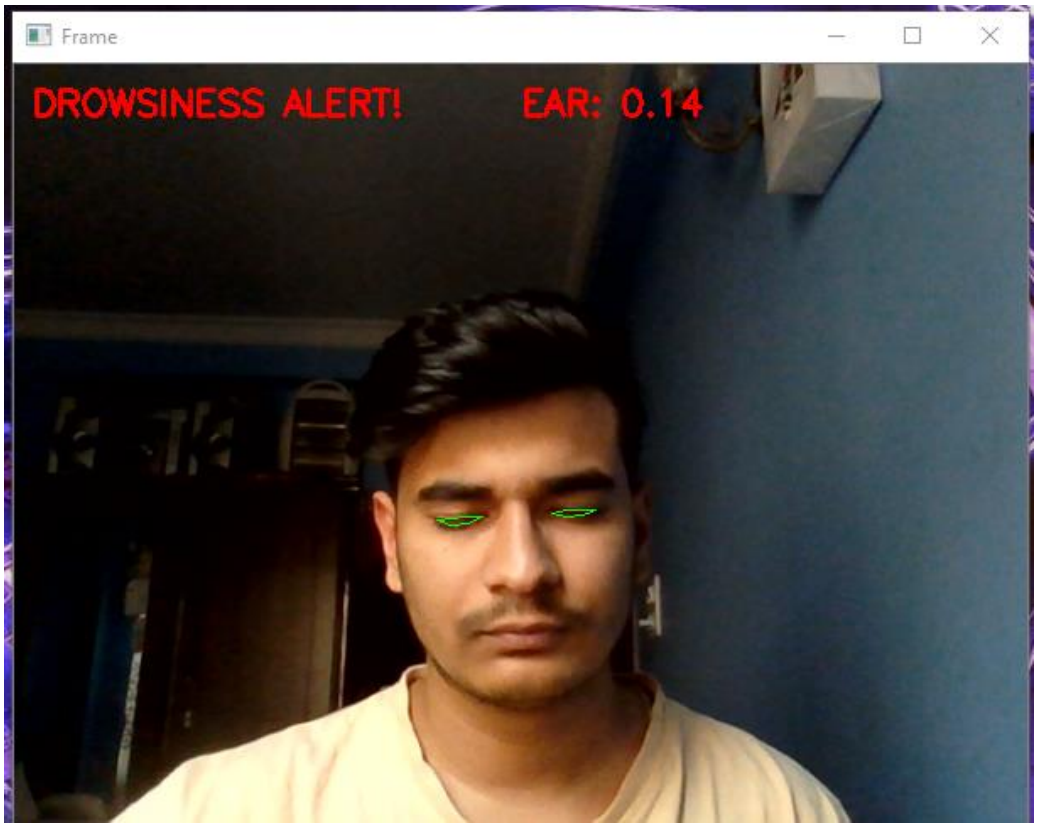


Figure 63 Drowsiness Detection Software

Here, in the implementation part, the same code above is run in the terminal and the above figures are the example of the script.

3.7.4 Iteration 4: Income

3.7.4.1 Database Connection

```
class Income(models.Model): #This class stores all the daily income produced by individual drivers
    taxi = models.ForeignKey(Taxi, on_delete=models.CASCADE)
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE)
    daily_income = models.IntegerField()
    registration_date = models.DateField()
    start_trip = models.FloatField()
    end_trip = models.FloatField()

    def __str__(self):
        return '%s' % (self.driver)
```

Figure 64 Income (models.py)

3.7.4.2 Backend

```
192  @login_required(login_url='/')
193  def add_income_view(request):
194      if request.method == "POST":
195          form = DailyIncomeForm(request.POST)
196          if form.is_valid():
197              data = form.cleaned_data
198              new_income = Income(taxi=data['taxi'], driver=data['driver'],
199                                  daily_income=data['daily_income'],
200                                  registration_date=data['registration_date'],
201                                  start_trip=data['start_trip'], end_trip=data['end_trip']
202                                  )
203              new_income.save()
204              return redirect('Taxi_Manager:view-income')
205      else:
206          form = DailyIncomeForm()
207      return render(request,'Taxi_Manager/add_daily_income.html',{'form': form})
208
209  class IncomeList(ListView):
210      model = Income
211      template_name = 'Taxi_Manager/totalearning.html'
212      # def total(request):
213      #     total_price = Income.objects.aggregate(Sum('daily_income'))
214      #     return render(request, 'Taxi_Manager/totalearning.html',{'total':total_price})
215
216  @login_required(login_url='/')
217  @allowed_owner(allowed_roles=['Owners'])
218  def total_price(request):
219      incomedata = Income.objects.all()
220      total_price = Income.objects.aggregate(Sum('daily_income'))
221      context = {'incomedata':incomedata,'total':total_price}
222      return render(request,'Taxi_Manager/totalearning.html',context)
223
224
```

Figure 65 add_income_view(views.py)

```python
class IncomeDetailView(DetailView):
    model = Income
    template_name = 'Taxi_Manager/view_income_detail.html'

class IncomeUpdateView(UpdateView):
    model = Income
    fields = ('taxi', 'driver',
              'daily_income', 'registration_date',
              'start_trip','end_trip'
              )
    template_name = 'Taxi_Manager/update_income_detail.html'
    def get_success_url(self):
        return reverse('Taxi_Manager:view-income')

class IncomeDelete(DeleteView):
    model = Income
    template_name = 'Taxi_Manager/delete.html'
    success_url = reverse_lazy('Taxi_Manager:view-income')
```

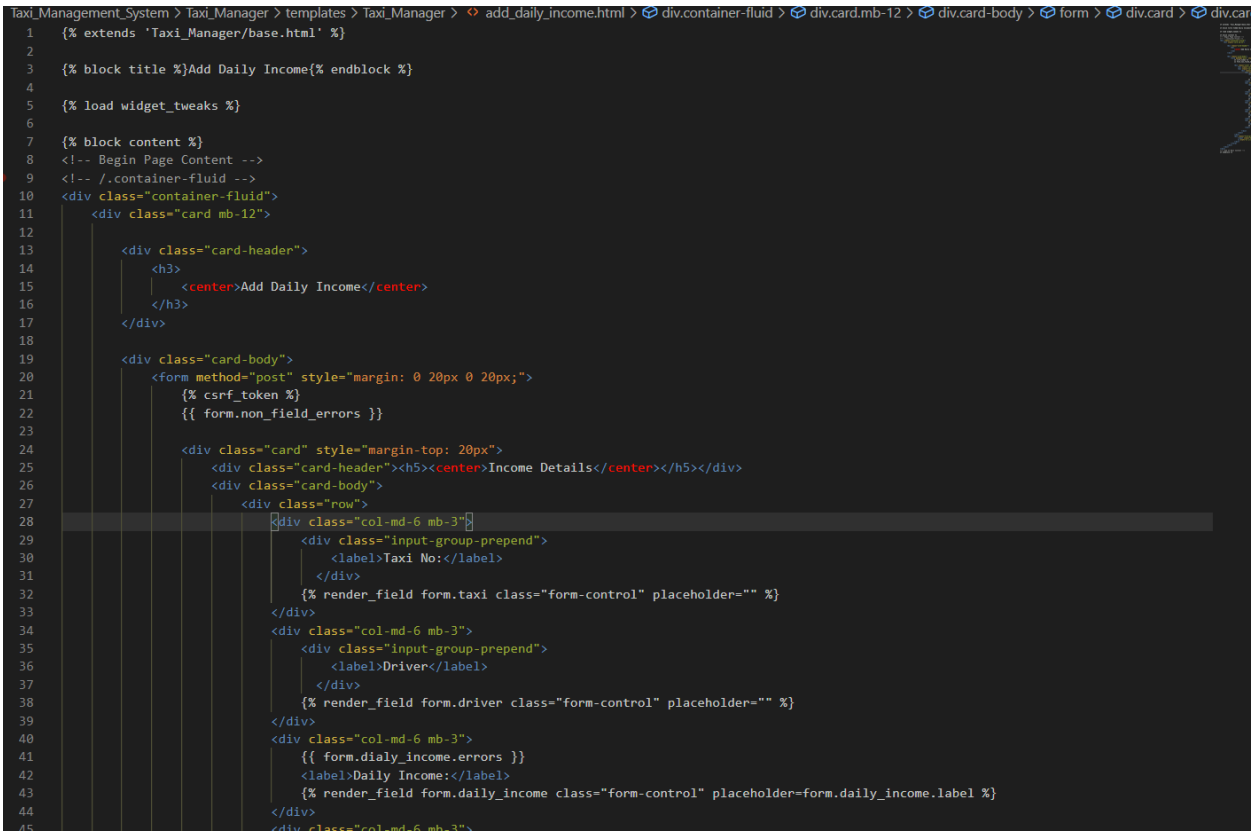
Figure 66 Income generic views

```python
path('add/income/', views.add_income_view, name='add-income'),
path('view/income', views.total_price, name='view-income'),
path('view/income/<int:pk>',views.IncomeDetailView.as_view(), name='income-detail'),
path('view/income/<int:pk>/update/',views.IncomeUpdateView.as_view(), name='update-income'),
path('view/income/<int:pk>/delete/',views.IncomeDelete.as_view(), name='delete-income'),
```

Figure 67 Income URLs

In the views.py above, we can see many @login_required which basically are the decorators in Django. This decorator will can allow the user to directly enter into the registration form page. In income also, default Django generic views are made which makes the CRUD process easier.

3.7.4.3 Frontend



```
                        <div class="col-md-6 mb-3">
                            <div class="input-group-prepend">
                                <label>Taxi No:</label>
                             </div>
                            {% render_field form.taxi class="form-control" placeholder="" %}
                        </div>
                        <div class="col-md-6 mb-3">
                            <div class="input-group-prepend">
                                <label>Driver</label>
                             </div>
                            {% render_field form.driver class="form-control" placeholder="" %}
                        </div>
                        <div class="col-md-6 mb-3">
                            {{ form.dialy_income.errors }}
                            <label>Daily Income:</label>
                            {% render_field form.daily_income class="form-control" placeholder=form.daily_income.label %}
                        </div>
                        <div class="col-md-6 mb-3">
                            {{ form.registration_date.errors }}
                            <label>Date of Registration:</label>
                            {% render_field form.registration_date class="form-control" placeholder=form.registration_date.label %}
                        </div>
                        <div class="col-md-6 mb-3">
                            {{ form.start_trip.errors }}
                            <label>Start Trip(KM):</label>
                            {% render_field form.start_trip class="form-control" placeholder=form.start_trip.label %}
                        </div>
                        <div class="col-md-6 mb-3">
                            {{ form.end_trip.errors }}
                            <label>End Trip(KM):</label>
                            {% render_field form.end_trip class="form-control" placeholder=form.end_trip.label %}
                        </div>

                    </div>
                </div>
            </div>
            <div class="form-group" style="margin-top: 20px">
                <input class="btn btn-primary" type="submit" value="Register">
                <a href="{% url 'Taxi_Manager:home' %}" class="btn btn-secondary">Cancel</a>
            </div>
        </form>
    </div>
</div>
</div>
```

Live Share | Connecting to Discord... | Ln 28, Col 56 | Spaces: 4 | UTF-8 | CRLF | HTML

Figure 68 add_daily_income.html

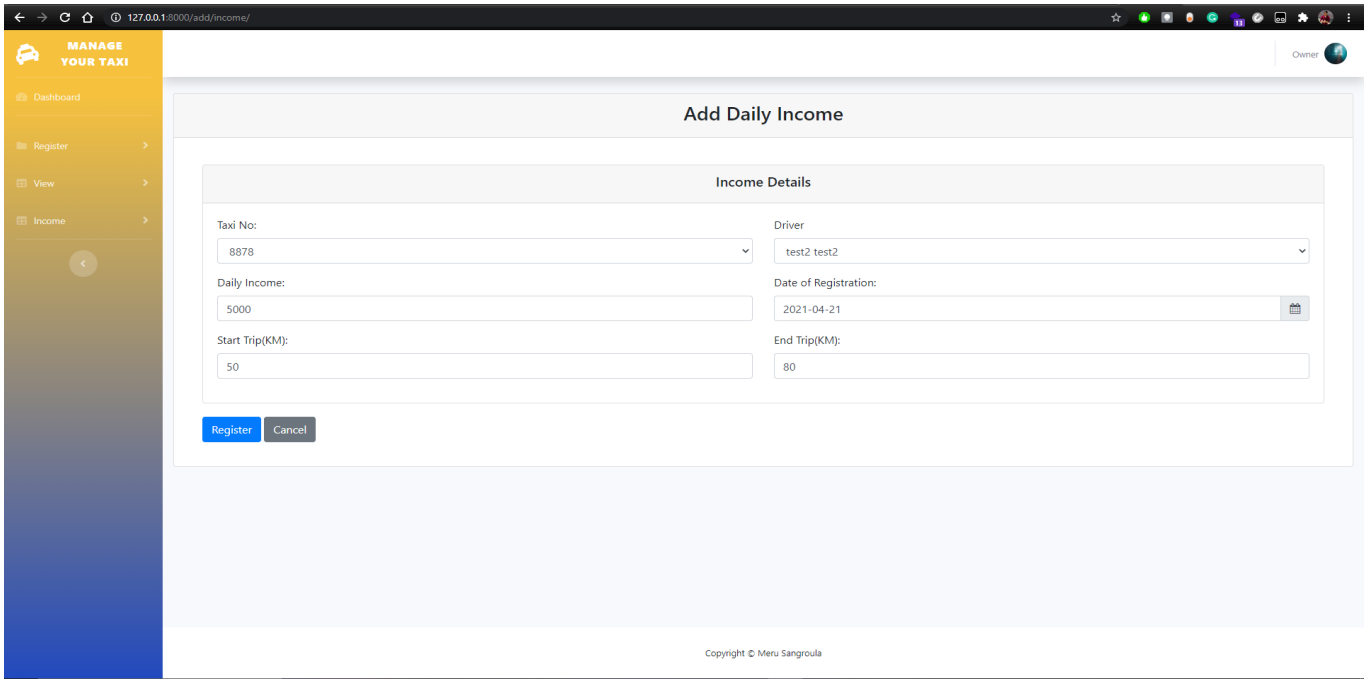
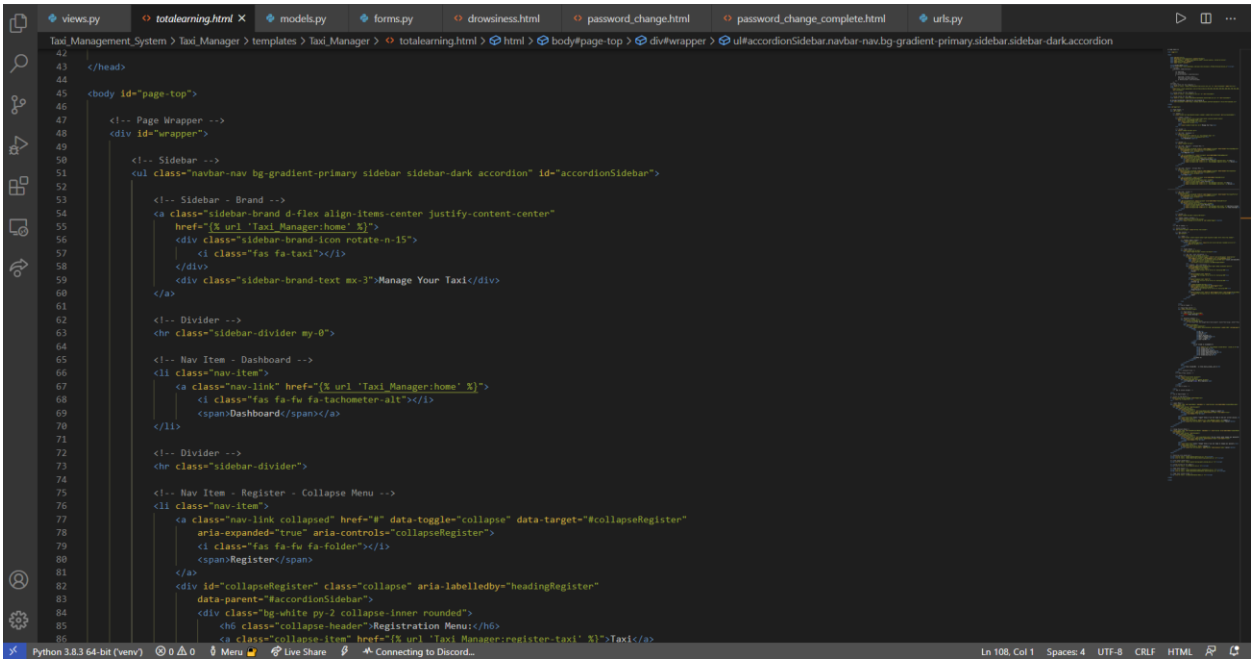
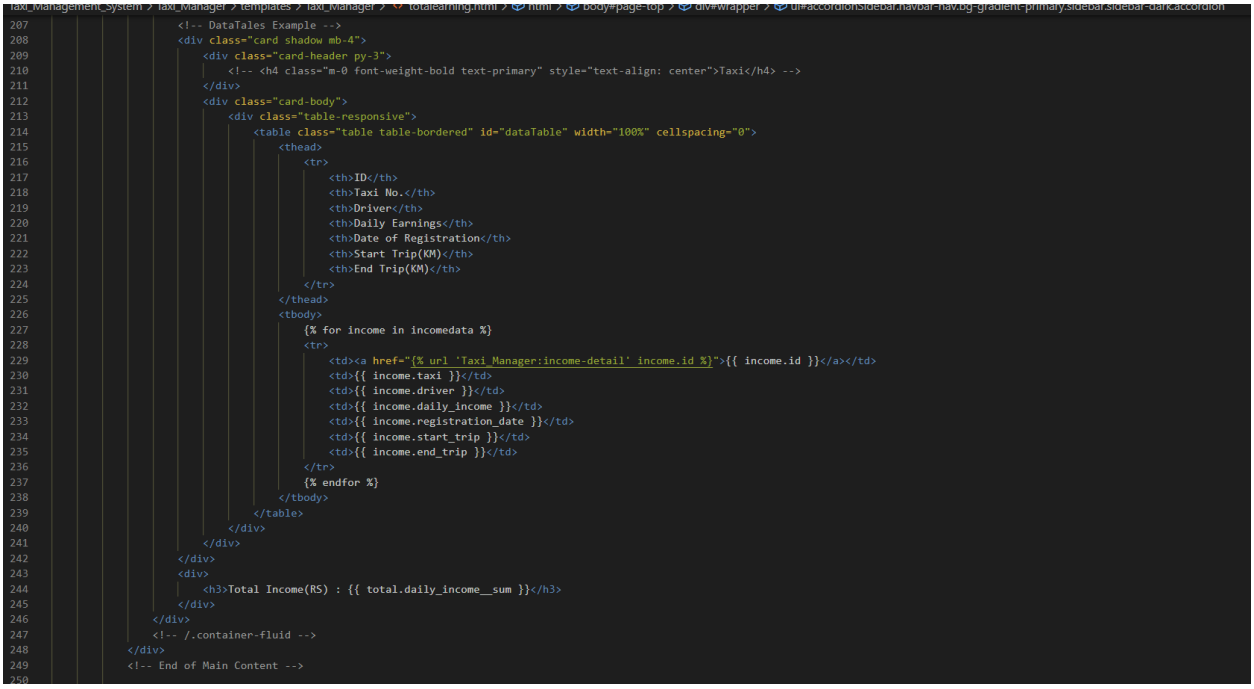


Figure 69 Add Daily Income Page



```
</head>

<body id="page-top">

    <!-- Page Wrapper -->
    <div id="wrapper">

        <!-- Sidebar -->
        <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar">

            <!-- Sidebar - Brand -->
            <a class="sidebar-brand d-flex align-items-center justify-content-center"
                href="{% url 'Taxi_Manager:home' %}">
                <div class="sidebar-brand-icon rotate-n-15">
                    <i class="fas fa-taxi"></i>
                </div>
                <div class="sidebar-brand-text mx-3">Manage Your Taxi</div>
            </a>

            <!-- Divider -->
            <hr class="sidebar-divider my-0">

            <!-- Nav Item - Dashboard -->
            <li class="nav-item">
                <a class="nav-link" href="{% url 'Taxi_Manager:home' %}">
                    <i class="fas fa-fw fa-tachometer-alt"></i>
                    <span>Dashboard</span></a>
            </li>

            <!-- Divider -->
            <hr class="sidebar-divider">

            <!-- Nav Item - Register - Collapse Menu -->
            <li class="nav-item">
                <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseRegister"
                    aria-expanded="true" aria-controls="collapseRegister">
                    <i class="fas fa-fw fa-folder"></i>
                    <span>Register</span>
                </a>
                <div id="collapseRegister" class="collapse" aria-labelledby="headingRegister"
                    data-parent="#accordionSidebar">
                    <div class="bg-white py-2 collapse-inner rounded">
                        <h6 class="collapse-header">Registration Menu:</h6>
                        <a class="collapse-item" href="{% url 'Taxi_Manager:register-taxi' %}">Taxi</a>
```

Figure 70 Totalearning.html (1)



```
                <!-- DataTales Example -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <!-- <h4 class="m-0 font-weight-bold text-primary" style="text-align: center">Taxi</h4> -->
                    </div>
                    <div class="card-body">
                        <div class="table-responsive">
                            <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
                                <thead>
                                    <tr>
                                        <th>ID</th>
                                        <th>Taxi No.</th>
                                        <th>Driver</th>
                                        <th>Daily Earnings</th>
                                        <th>Date of Registration</th>
                                        <th>Start Trip(KM)</th>
                                        <th>End Trip(KM)</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    {% for income in incomedata %}
                                    <tr>
                                        <td><a href="{% url 'Taxi_Manager:income-detail' income.id %}">{{ income.id }}</a></td>
                                        <td>{{ income.taxi }}</td>
                                        <td>{{ income.driver }}</td>
                                        <td>{{ income.daily_income }}</td>
                                        <td>{{ income.registration_date }}</td>
                                        <td>{{ income.start_trip }}</td>
                                        <td>{{ income.end_trip }}</td>
                                    </tr>
                                    {% endfor %}
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
                <div>
                    <h3>Total Income(RS) : {{ total.daily_income__sum }}</h3>
                </div>
            </div>
            <!-- /.container-fluid -->
        </div>
        <!-- End of Main Content -->
```

Figure 71 Totalearning.html (2)

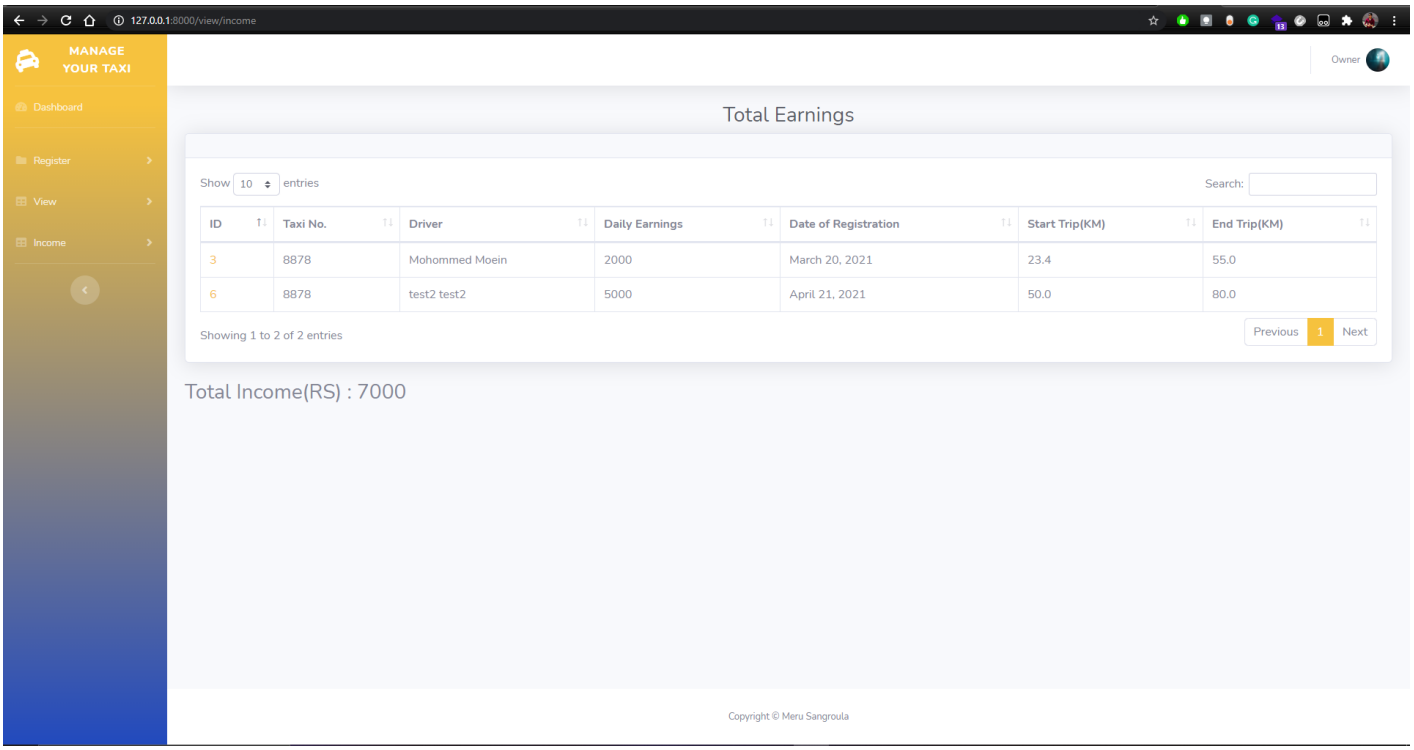


Figure 72 Total Earnings Page

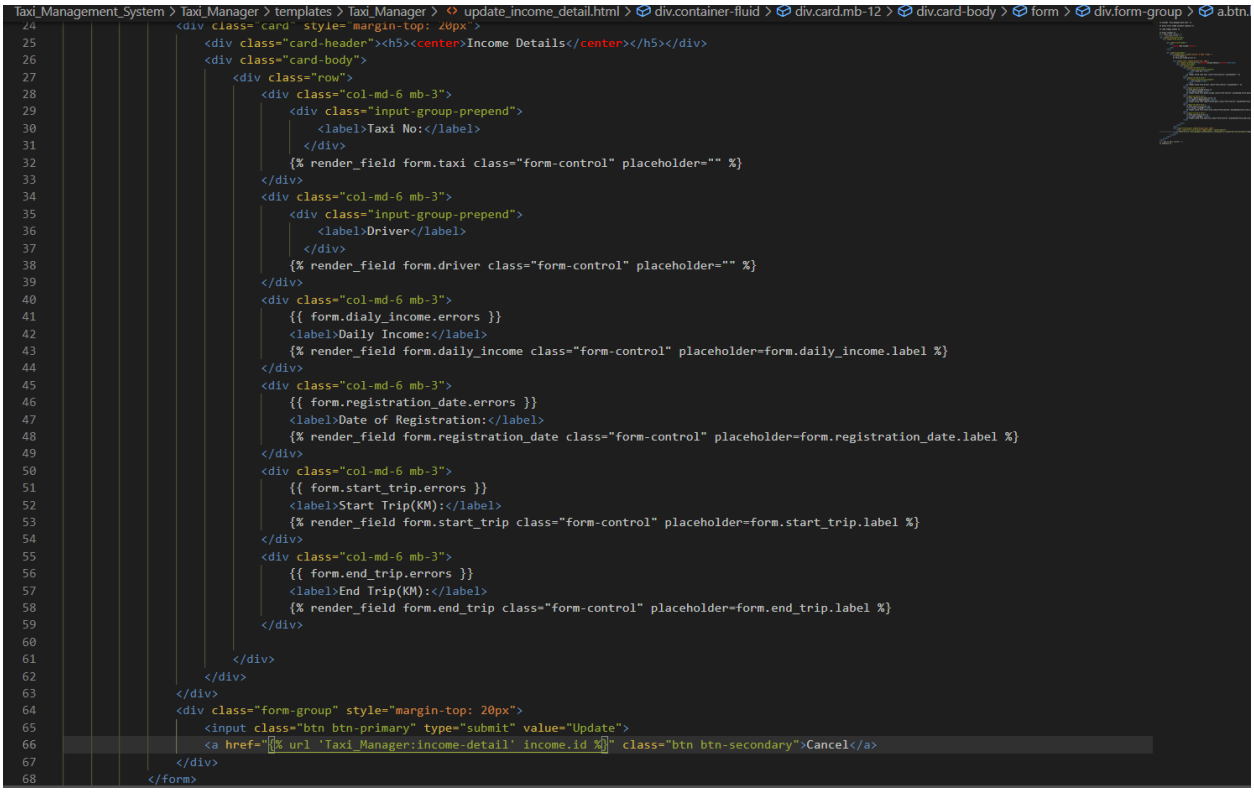


Figure 73 update_income.html

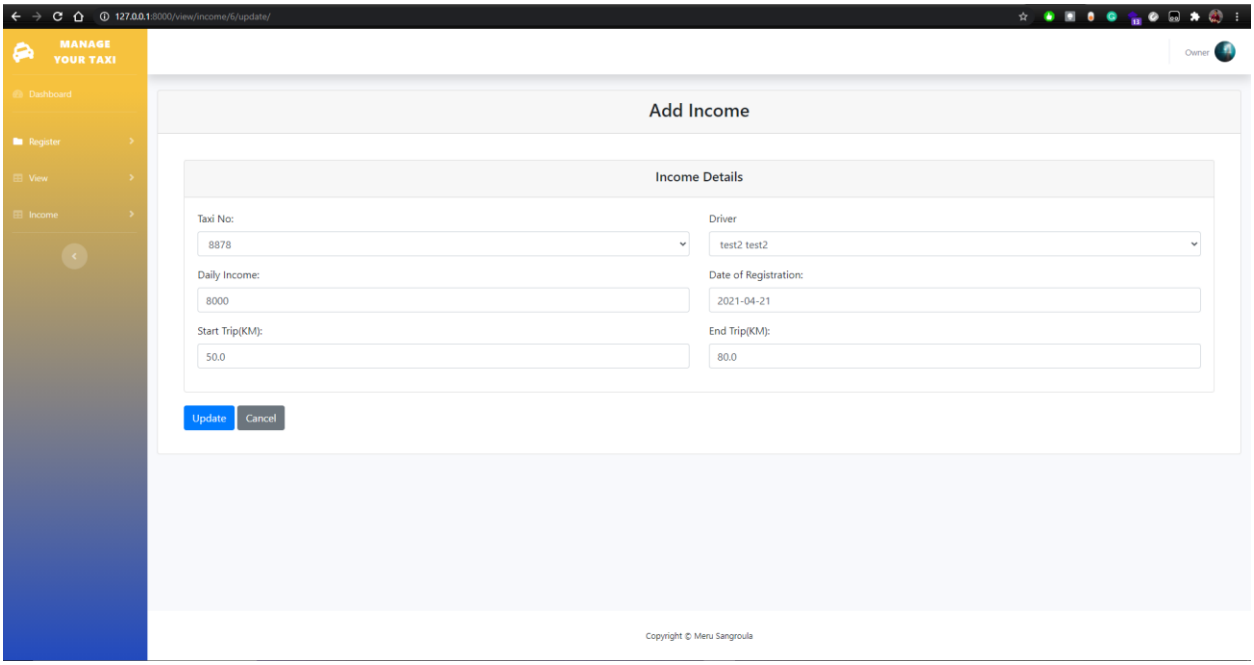


Figure 74 Update Income Page

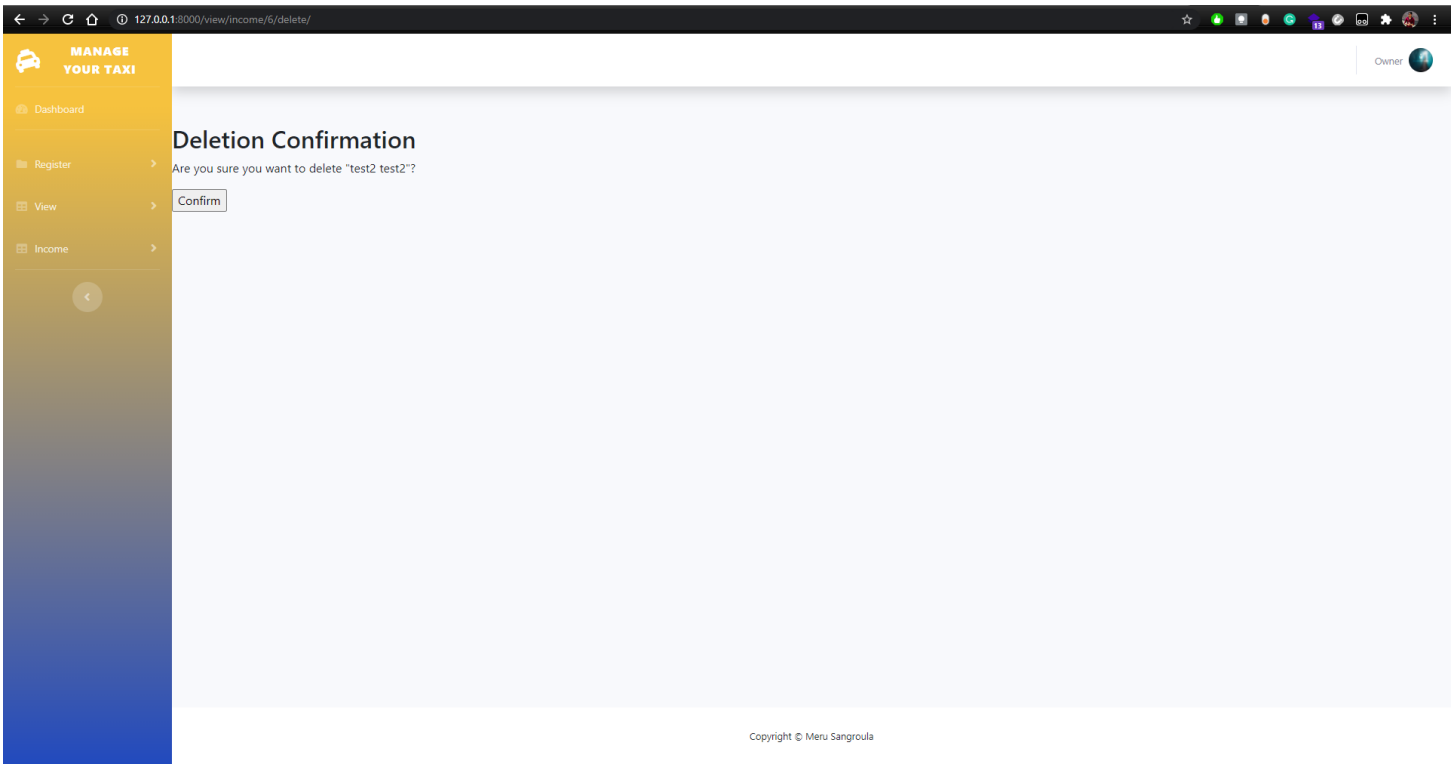


Figure 75 Delete Page

The above figure shows all the CRUD pages available inside Income page. The add income page is only for the Owner but not for the driver. The idea behind this was the driver will enter all the details of his/her day in the taxi like the start and end km, today's income and owner can keep the record out of it.

3.7.5 Iteration 5: Login Authentication

3.7.5.1 Backend

```
from django.contrib.auth.models import Group, User
from django.db.models import Sum, Count, query
from django.contrib.auth import authenticate, login, logout, decorators
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .decorators import all_login, allowed_owner, unauthenticated_user
from django.http import HttpResponse, request
from django.contrib import auth
import datetime
from django.contrib.auth import views as auth_views


# API imports
from rest_framework.views import APIView
from .serializer import DriverSerializer, DrowsinessSerializer
from rest_framework.response import Response
from rest_framework import status, generics
from rest_framework.authtoken.views import ObtainAuthToken
from rest_framework.authtoken.models import Token
from rest_framework.permissions import IsAuthenticated


@unauthenticated_user
def loginPage(request):
    if request.method == 'POST':
        username=request.POST.get('username')
        password=request.POST.get('password')

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('Taxi_Manager:home')

        else:
            messages.error(request, "Username OR password incorrect")

    return render(request, 'registration/login.html')

def logoutUser(request):
    logout(request)
    return redirect('Taxi_Manager:Login')
```

Figure 76 Login View (view.py)

```
app_name = 'Taxi_Manager'
urlpatterns = [
    path('', views.loginPage, name='Login'),
    path('home/', views.home_page, name='home'),
    path('logout/', views.logoutUser, name='Logout'),
    path('welcome/', views.welcome, name='welcome'),
```

Figure 77 Login URLs

The loginPage function is only accessed by the authenticat6ed user and this function basically checks the username and password provided by the anonymous user. When clicked on logout, the logout function is called, and the user is logout. Once logout, the user again cannot simply go back, instead the user again has to go through the login process.

3.7.5.2 Frontend

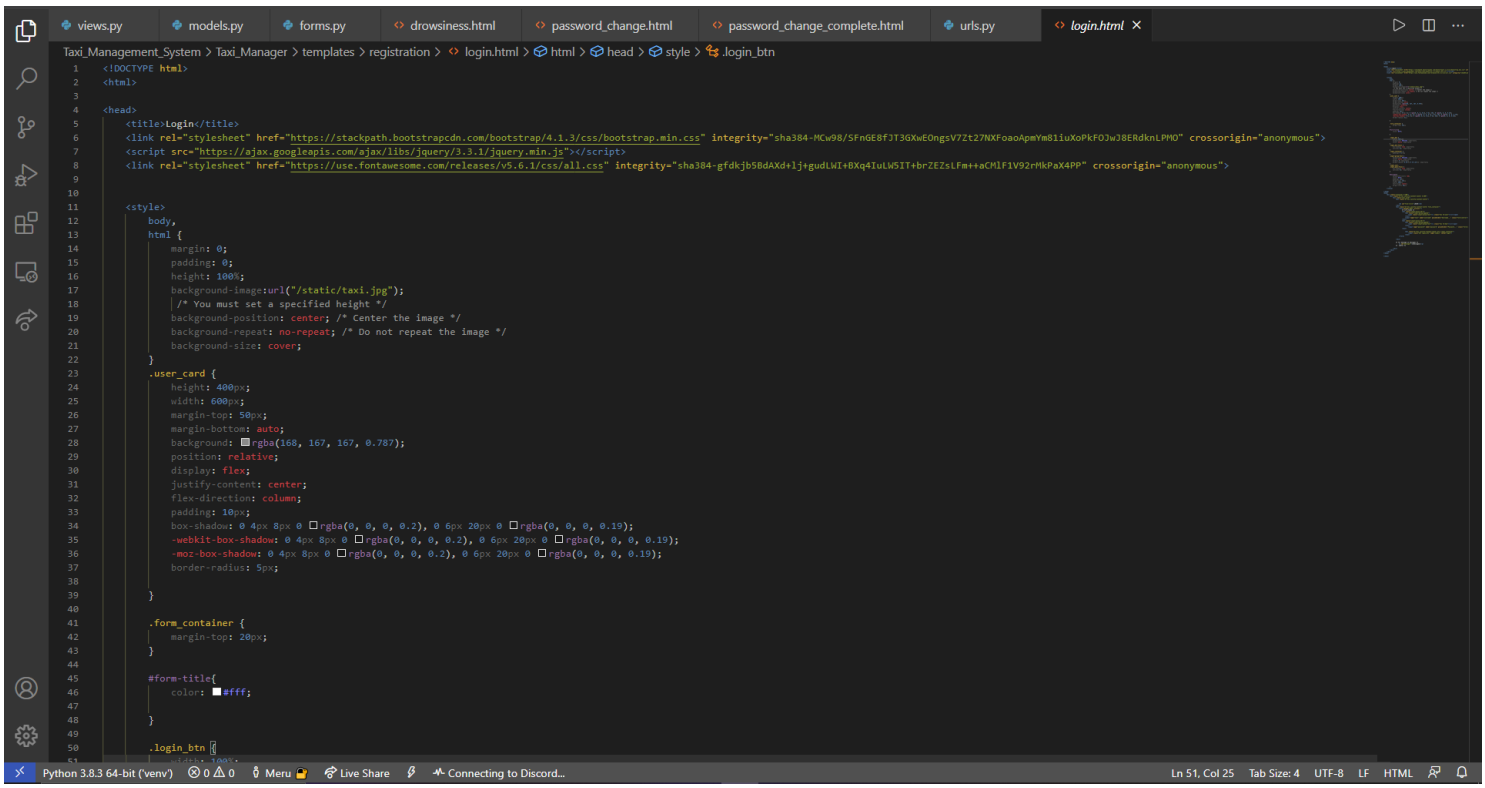


Figure 78 Login.html (1)

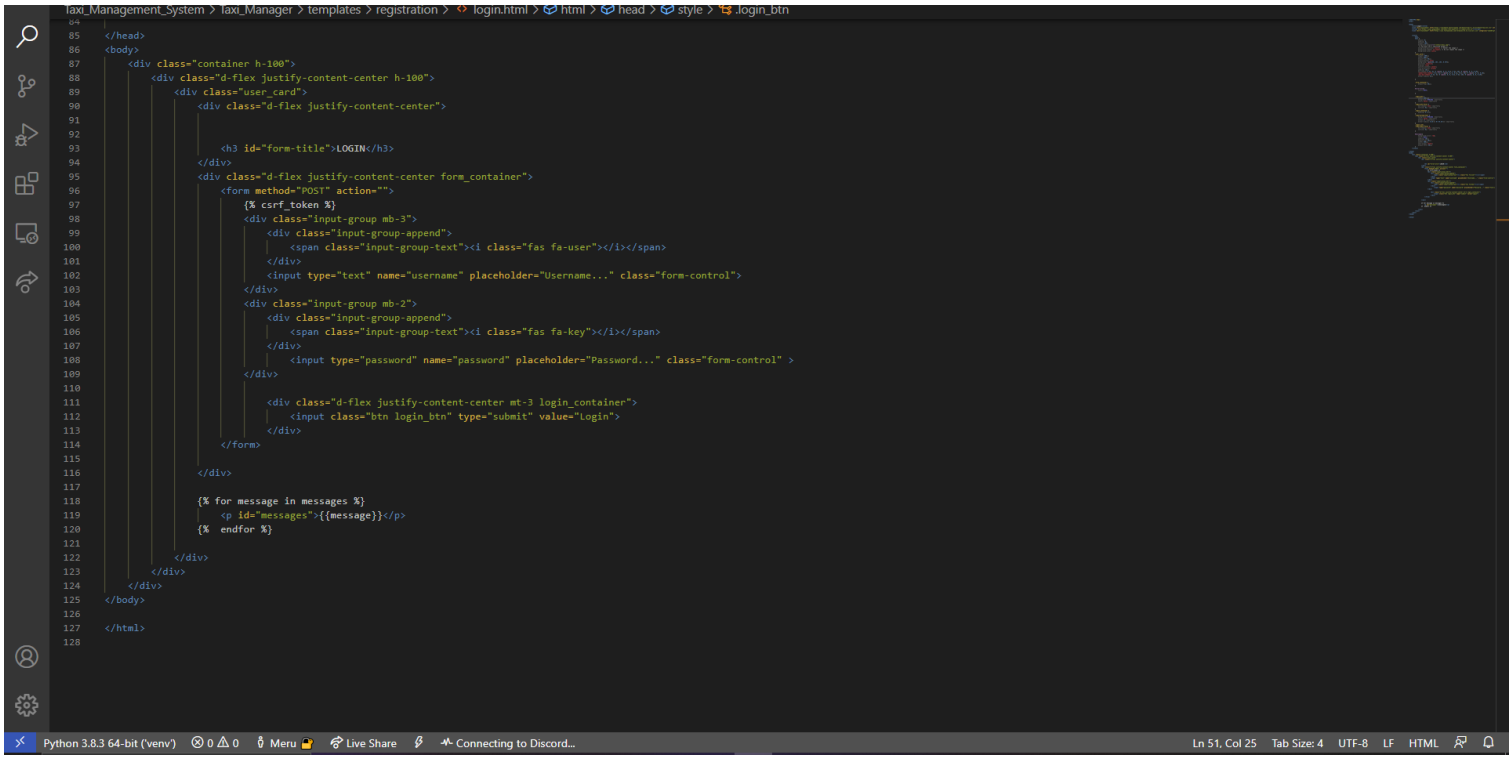


Figure 79 Login.html (2)

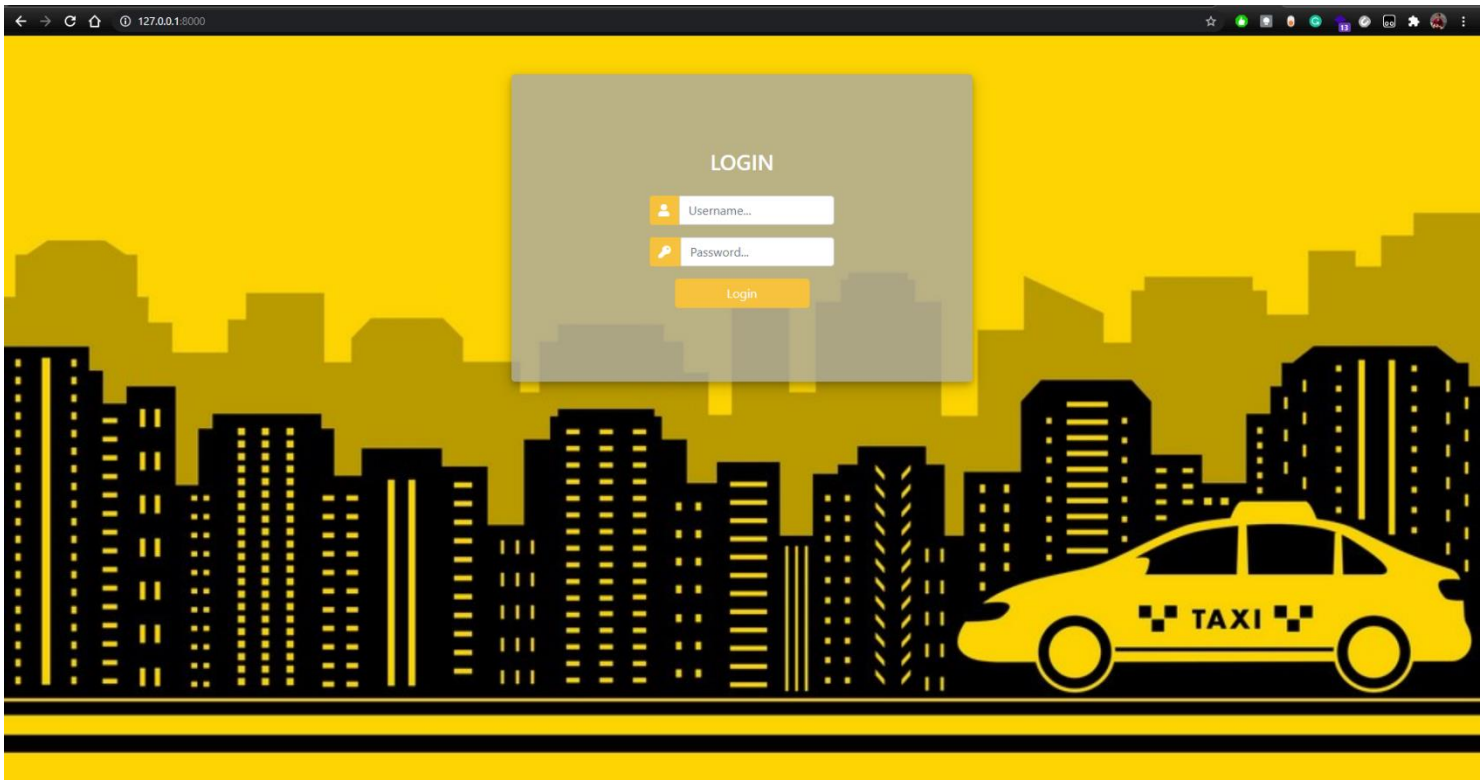


Figure 80 Login Page (1)

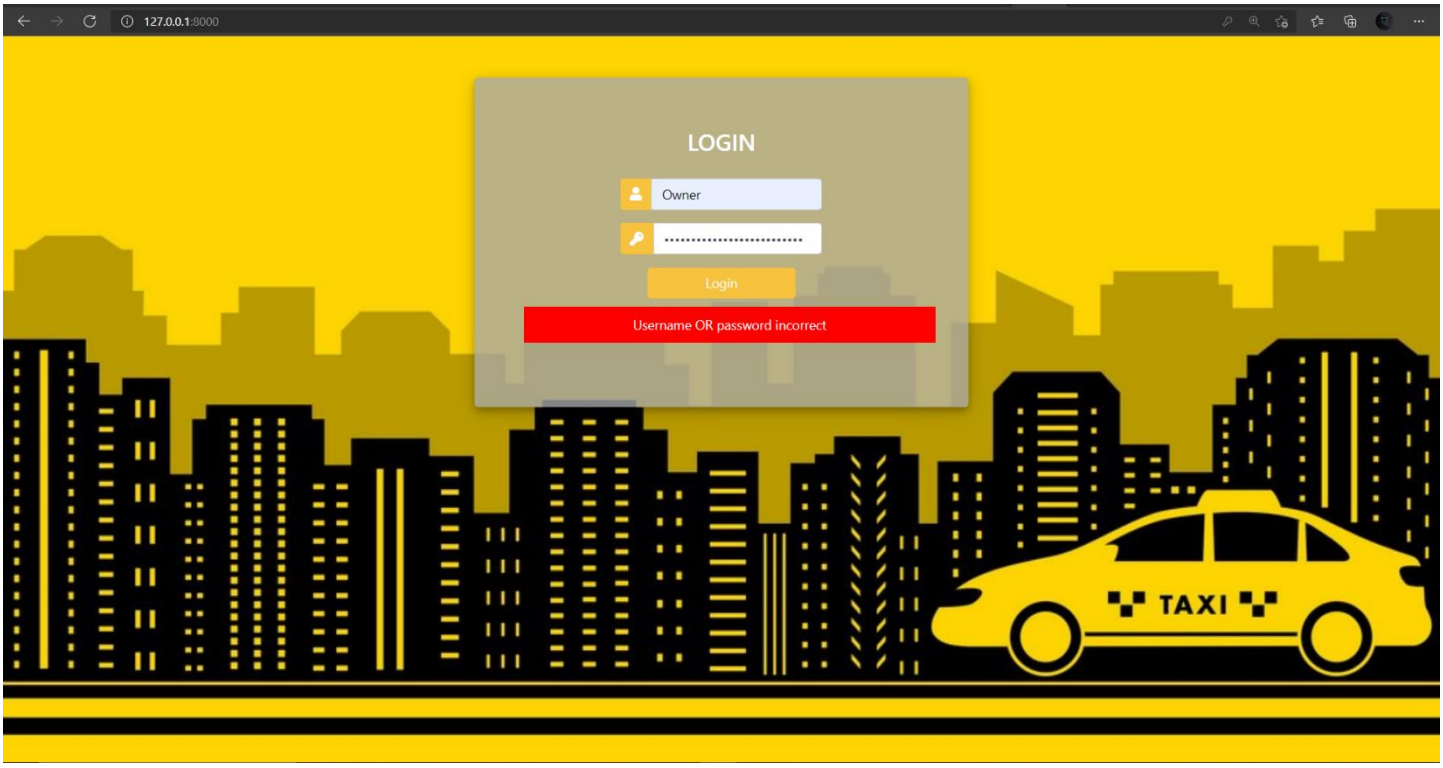


Figure 81 Login Page (2)

Above figure are the login page. Here we can see in login Page (2) that an error message is shown once the login is not authenticated and the login page renders again.

3.7.6 Iteration 6: Drowsiness API

3.7.6.1 Code

```
cv2.destroyAllWindows()
vs.stop()

print(times)
print(len(times))

for time in times:
    playload = {'drow_data':time, 'name':'test100'}
    res = requests.post('http://127.0.0.1:8000/api/drow/', data=playload)
# #print(res.text)
```

Figure 82 Sending POST request

3.7.6.2 Implementation

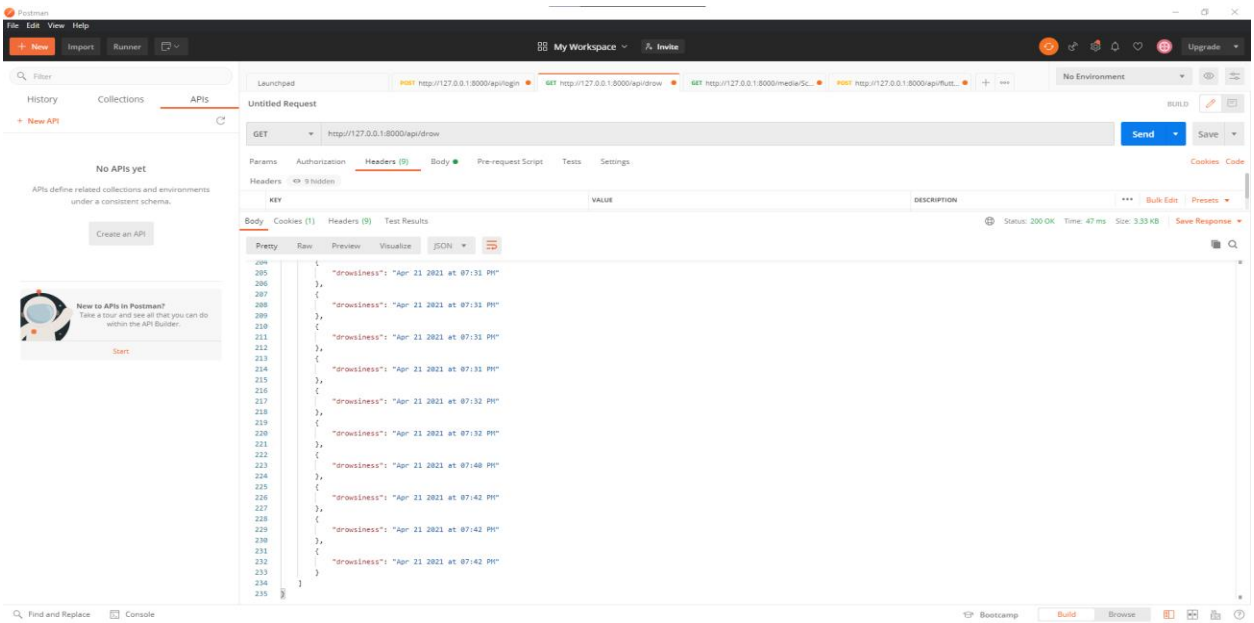


Figure 83 Postman API

In the API, once the post request is sent from the drowsiness script, the given Django URL is called and here we can see in figure 67 that the data is sent in the API URL.

3.7.7 Iteration 7: Charts

3.7.7.1 Iteration 7: Backend

```
def driver_age_chart(request):
    queryset = Driver.objects.all()
    labels = ['20-30', '30-40', '40-50', '50-60']
    a = 0
    b = 0
    c = 0
    d = 0

    for obj in queryset:
        difference = datetime.datetime.now().date() - obj.date_of_birth
        years = round(difference.days/365)

        if 20 < years <= 30:
            a = a + 1
        if 30 < years <= 40:
            b = b + 1
        if 40 < years <= 50:
            c = c + 1
        if 50 < years <= 60:
            d = d + 1

    data = [a, b, c, d,]

    return JsonResponse(data={
        'labels': labels,
        'data': data,
    })
```

Figure 84 Age Chart (views.py)

```python
def driver_gender_chart(request):
    queryset = Driver.objects.all()
    labels = ['Male', 'Female', 'Transgender']
    m = 0
    f = 0
    t = 0

    for driver in queryset:
        if driver.gender == 'Male':
            m = m + 1
        if driver.gender == 'Female':
            f = f + 1
        if driver.gender == 'Transgender':
            t = t + 1

    data = [m, f, t]

    return JsonResponse(data={
        'labels': labels,
        'data': data,
    })
```

Figure 85 Gender Chart (views.py)

```python
path('charts/patient/', DriverChartView.as_view(), name='driver-chart'),

path('ajax/driver_age_chart/', views.driver_age_chart, name='driver-age-chart'),
path('ajax/driver_gender_chart/', views.driver_gender_chart, name='driver-gender-chart'),
```

Figure 86 Chart URLs

3.7.7.2 Frontend

```
    <div class="container-fluid" style="width: 1350px; margin-left: -1px;">
        <!-- Page Heading -->
        <div class="row">

            <div class="col-xl-7">
                <!-- Bar Chart -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <h4 class="m-0 font-weight-bold text-primary" align="center">Driver Age</h4>
                    </div>
                    <div class="card-body">
                        <div class="chart-area", style="height: 400px; width: 700px;">
                            <canvas id="driver_age_chart"></canvas>
                        </div>
                    </div>
                </div>
            </div>

            <div class="col-xl-5">
                <div class="card shadow mb-3", style="width: 700px; height: 500px;">
                    <div class="card-header py-3">
                        <h4 class="m-0 font-weight-bold text-primary" align="center">Gender Chart</h4>
                    </div>
                    <div class="card-body">
                        <div class="chart-area">
                            <canvas id="driver_gender_chart" width="122px" height="70px"></canvas>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <!-- End of Main Content -->
<!--  AJAX + Charts-->
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>
```

Figure 87 index.html (1)

```
<script>
  var endpoint = '{% url "Taxi_Manager:driver-gender-chart" %}';

  $.ajax({
    url: endpoint,
    success: function (data) {
      var ctx = document.getElementById('driver_gender_chart').getContext('2d');
      var myChart = new Chart(ctx, {
        type: 'pie',
        data: {
          labels: data.labels,
          datasets: [{
            label: '',
            data: data.data,
            backgroundColor: [
                'rgba(54, 162, 235, 0.2)',
                'rgba(255, 99, 132, 0.2)',
                'rgba(186, 85, 211, 0.2)',

            ],
            borderColor: [
                'rgba(54, 162, 235, 1)',
                'rgba(255, 99, 132, 1)',
                'rgba(186, 85, 211, 1)',
            ],
            borderWidth: 1
          }]
        },
        options: {
          responsive: true,
          legend: {
            position: 'right',
            display: true,
          },
          title: {
            display: true,
            text: '',
            fontSize: 30,
            fontFamily: 'Helvetica',
          },
        }
      });
```

Figure 88 index.html (2)

```
$.ajax({
  url: endpoint,
  success: function (data) {
    var ctx = document.getElementById('driver_age_chart').getContext('2d');
    var myChart = new Chart(ctx, {
      type: 'bar',
      data: {
        labels: data.labels,
        datasets: [{
          label: '',
          data: data.data,
          backgroundColor: [
              'rgba(255, 99, 132, 0.2)',
              'rgba(54, 162, 235, 0.2)',
              'rgba(255, 206, 86, 0.2)',
              'rgba(75, 192, 192, 0.2)',
              'rgba(153, 102, 255, 0.2)',
              'rgba(255, 159, 64, 0.2)',
              'rgba(153,204,204,0.2)',
              'rgba(255,0,0,0.2)',
          ],
          borderColor: [
              'rgba(255, 99, 132, 1)',
              'rgba(54, 162, 235, 1)',
              'rgba(255, 206, 86, 1)',
              'rgba(75, 192, 192, 1)',
              'rgba(153, 102, 255, 1)',
              'rgba(255, 159, 64, 1)',
              'rgba(255,0,0,0.2)',
          ],
          borderWidth: 3,
          // fill: false,
          // lineTension: 0,
          // pointBackgroundColor: 'rgba(255, 99, 132, 1)',
        }]
      },
      options: {
        responsive: true,
        legend: {
          position: 'right',
          display: false,
        },
```

Figure 89 index.html (3)

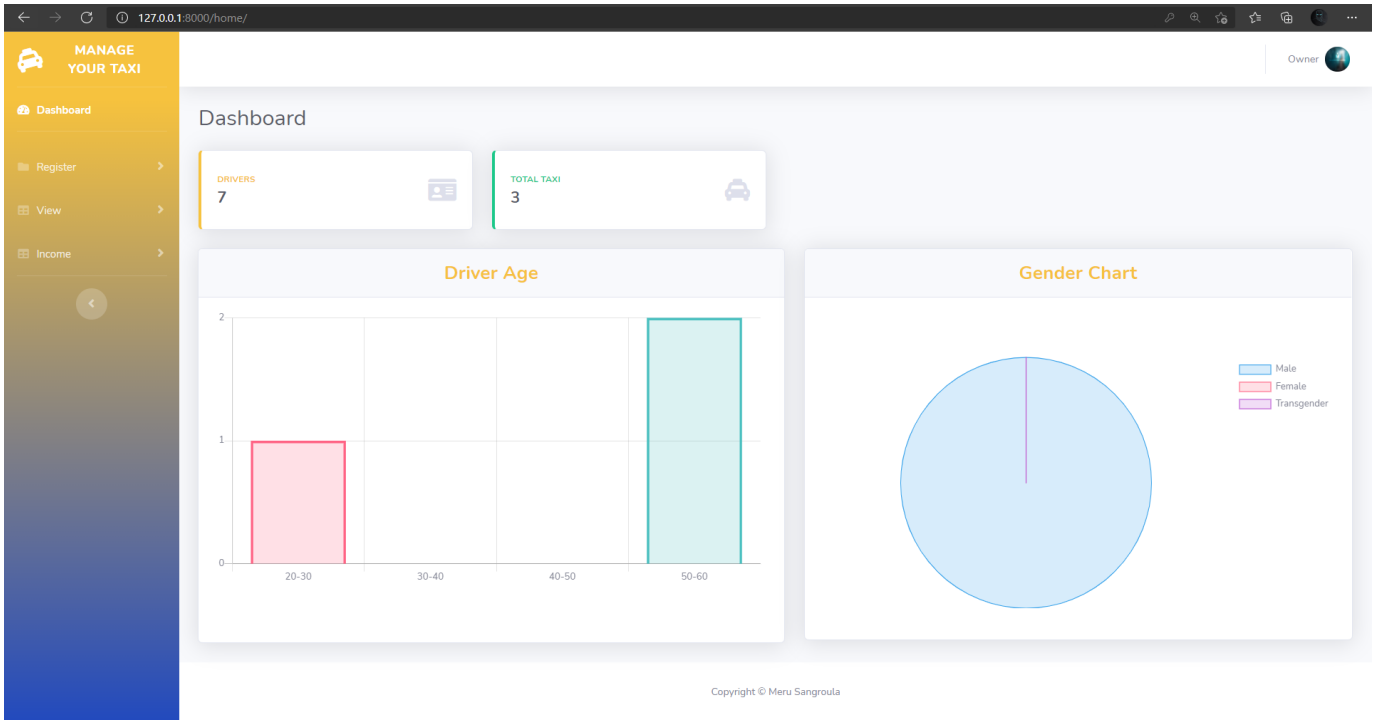


Figure 90 Chart

3.7.8 Iteration 8: Mobile Application

3.7.8.1 Backend

models.dart

lib > models > models.dart > Loginn

```dart
class Loginn {
  List<String> records = [];
  // Loginn({this.username, this.password});
  Loginn.fromJson(Map<String, dynamic> drowsiness) {
    //print(drowsiness['serilizer']);
    int size = drowsiness['serilizer'].length;

    for (int i = 0; i <= size - 1; i++) {
      records.add(drowsiness['serilizer'][i]['drowsiness']);
    }
  }
}
```

Figure 91 models.dart

verify.dart

lib > Screens > Login > verify.dart > ...

```dart
//import 'package:flutter/cupertino.dart';
import 'dart:convert';

import 'package:drowsydriver/models/models.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
// import 'dart:convert';
import 'package:flutter_session/flutter_session.dart';
//import 'dart:convert';

// ignore: missing_return
Future<Loginn> verify(
    BuildContext context, String username, String password) async {
  var response = await http.post(
    'http://10.0.2.2:8000/api/login',
    body: {'username': username, 'password': password},
  );
  print(response.body);
  if (response.statusCode == 200) {
    Loginn drow = Loginn.fromJson(json.decode(response.body));
    // print(response.body);
    return drow;
  }
  return showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text('Error'),
          content: Text('Invalid username or password'),
          actions: <Widget>[
            FlatButton(
              onPressed: () {
                Navigator.pop(context);
              },
              child: Text('Okay'),
            ), // FlatButton
          ], // <Widget>[]
        ); // AlertDialog
      });
}
```

Figure 92 verify.dart

In models.dart, the code is written for json data. Since the api data is in the list, this file takes the data from the list. Verify.dart basically gets the api response and does validation accordingly.

3.7.8.2 Frontend

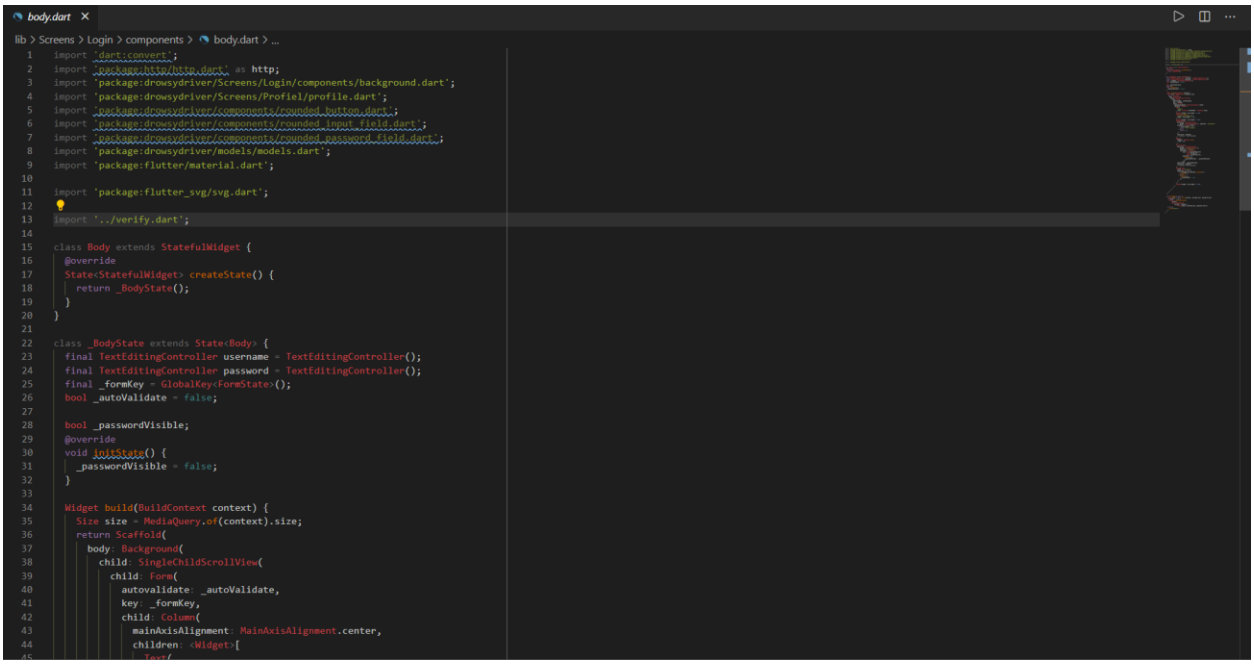


Figure 93 body.dart (1)

```
Widget build(BuildContext context) {
  Size size = MediaQuery.of(context).size;
  return Scaffold(
    body: Background(
      child: SingleChildScrollView(
        child: Form(
          autovalidate: _autoValidate,
          key: _formKey,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text(
                "LOGIN",
                style: TextStyle(fontWeight: FontWeight.bold),
              ), // Text
              SizedBox(height: size.height * 0.03),
              SvgPicture.asset(
                "assets/icons/login.svg",
                height: size.height * 0.35,
              ), // SvgPicture.asset
              SizedBox(height: size.height * 0.03),
              TextFormField(
                decoration: InputDecoration(
                    border: OutlineInputBorder(), labelText: "Username"), // InputDecoration
                validator: (String value) {
                  if (value.length == 0) {
                    return 'Text is empty';
                  } else {
                    return null;
                  }
                },
                controller: username,
                // hintText: "Your Username",
              ), // TextFormField
              Padding(
                padding: EdgeInsets.only(
                  top: 20.0,
                ), // EdgeInsets.only
              ), // Padding
              TextFormField(
                decoration: InputDecoration(
```

Figure 94 body.dart (2)

```
                padding: EdgeInsets.only(
                  top: 20.0,
                ), // EdgeInsets.only
              ), // Padding
              TextFormField(
                decoration: InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: "Password",
                    suffixIcon: IconButton(
                        icon: Icon(_passwordVisible
                            ? Icons.visibility
                            : Icons.visibility_off), // Icon
                        onPressed: () {
                          setState(() {
                            _passwordVisible = !_passwordVisible;
                          });
                        })), // IconButton // InputDecoration
                obscureText: !_passwordVisible,
                controller: password,
                // hintText: "Your Username",
              ), // TextFormField
              RaisedButton(
                child: Text("Login"),
                onPressed: () async {
                  if (_formKey.currentState.validate()) {
                    fetchData();
                  } else {
                    setState(() {
                      _autoValidate = true;
                    });
                  }
                },
              ), // RaisedButton
              SizedBox(height: size.height * 0.03),
            ], // <Widget>[]
          ), // Column
        ), // Form
      ), // SingleChildScrollView
    ), // Background
  ); // Scaffold
}
```

Figure 95 body.dart (3)

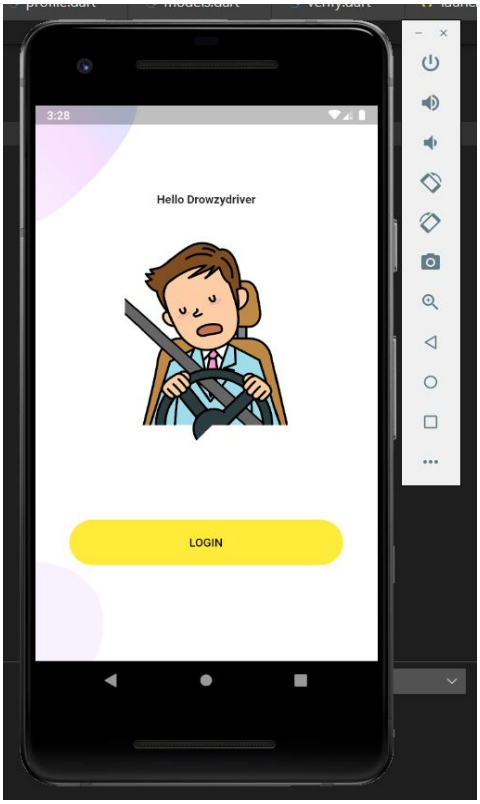


Figure 96 Login Screen (1)

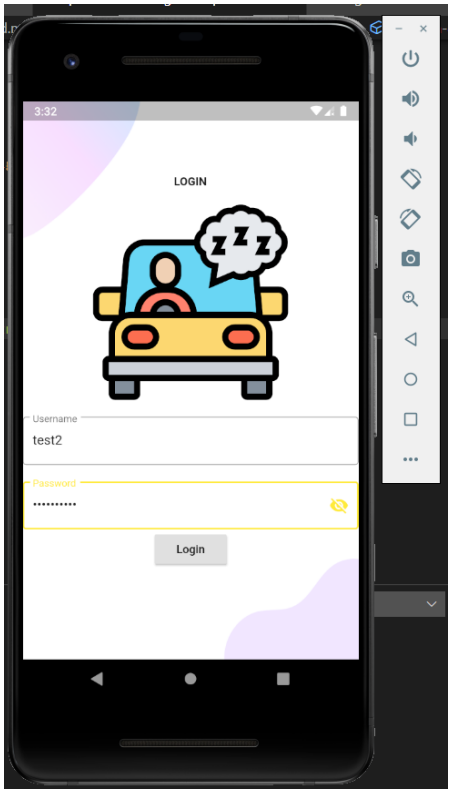


Figure 97 Login Screen (2)

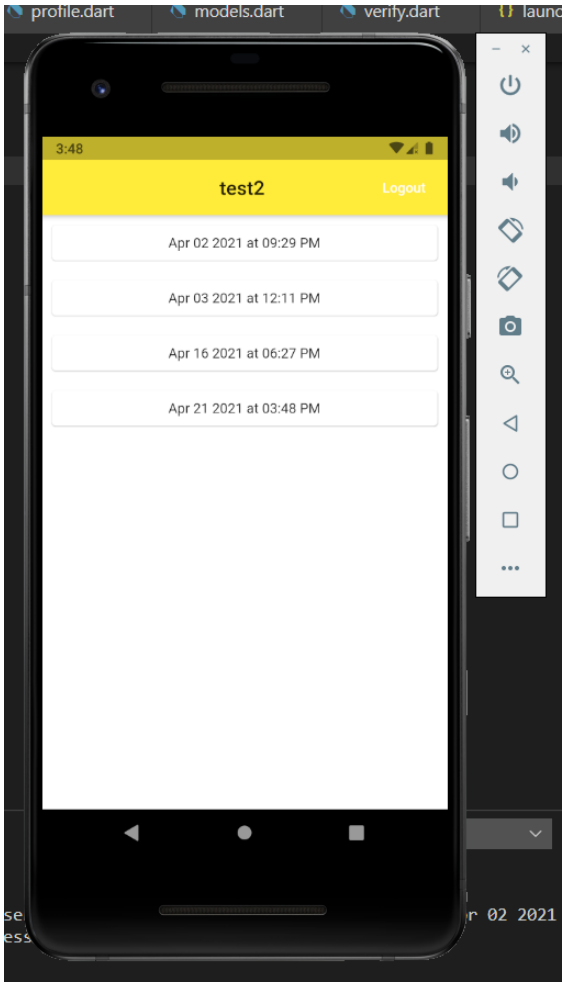


Figure 98 Drowsy Data in mobile

These are the results of the frontend code. There are 2 login screens for the design purpose seen in Login screen (1) and login screen (2). The drowsy data in mobile is basically the API content. The data seen in the picture above are retrieve from the above API. And at last there is a logout button.

More Project codes are on appendix section **(Click Here).**

CHAPTER 4: TESTING AND ANALYSIS

4.1 TEST PLAN

4.1.1 UNIT TESTING, TEST PLAN

Unit testing is the type of software testing where individual units or components of a software are tested. The purpose of unit testing is to validate that each unit of the software code performs as expected (guru99, 2021).

For unit testing, I will be testing each features of the project and provide proper screenshot of the outcomes. I will also make a table where I will mention all the actions, outcomes and results of the test.

4.1.2 SYSTEM TESTING, TEST PLAN

System testing is the level of testing that validates the complete and fully integrated software product (guru99, 2021).

For system testing, I will be testing all the outcomes while hitting API to show whether the API is working or not. I have also planned to test the overall run time speed of the project URL and I will be also testing database connections.

4.2 UNIT TESTING

4.2.1 Owner Login and Login Validation

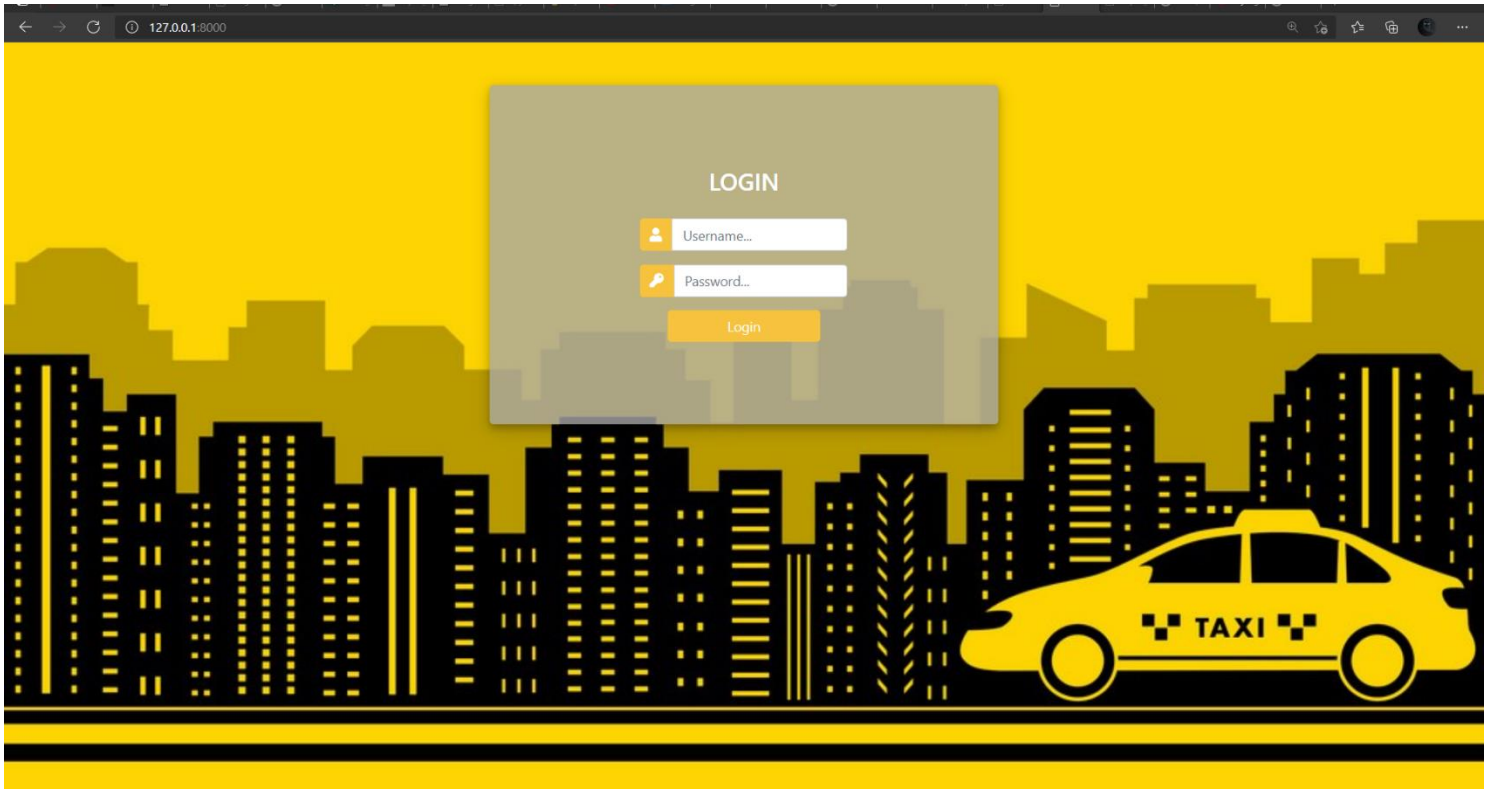


Figure 99 Owner Login Test (1)

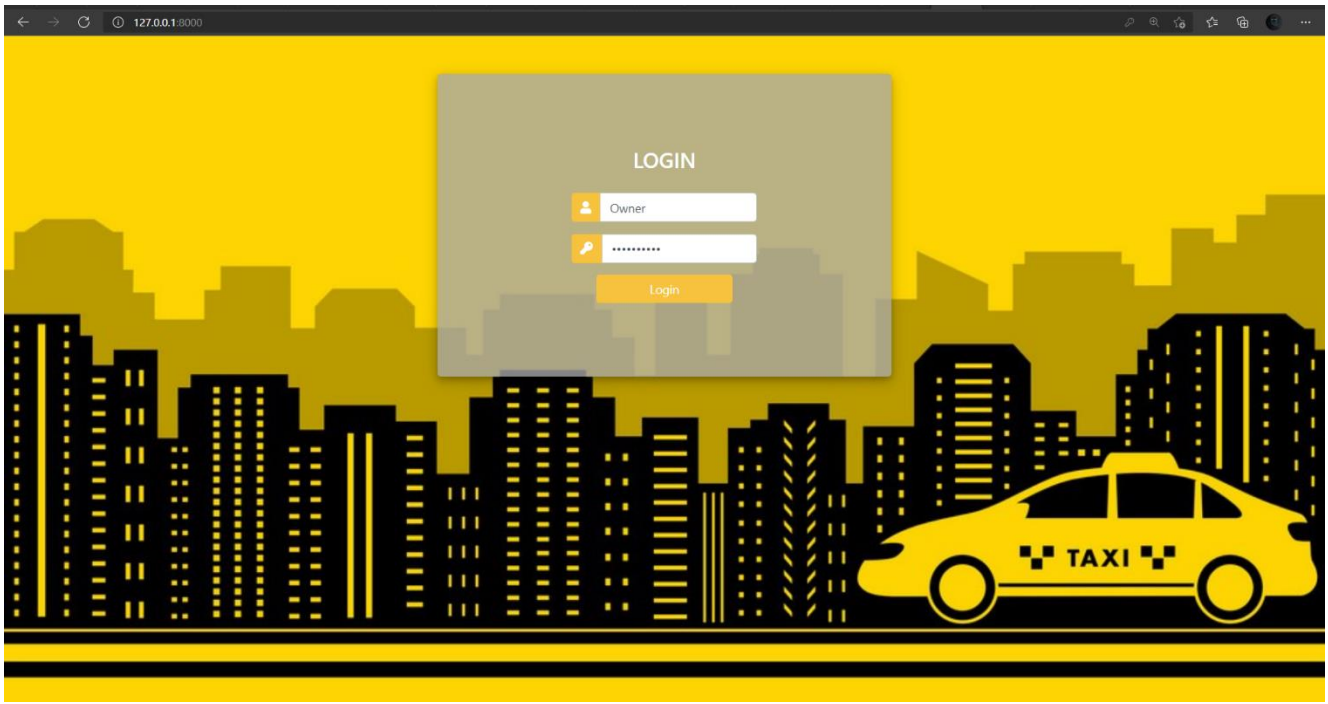


Figure 100 Owner Login Test (2)

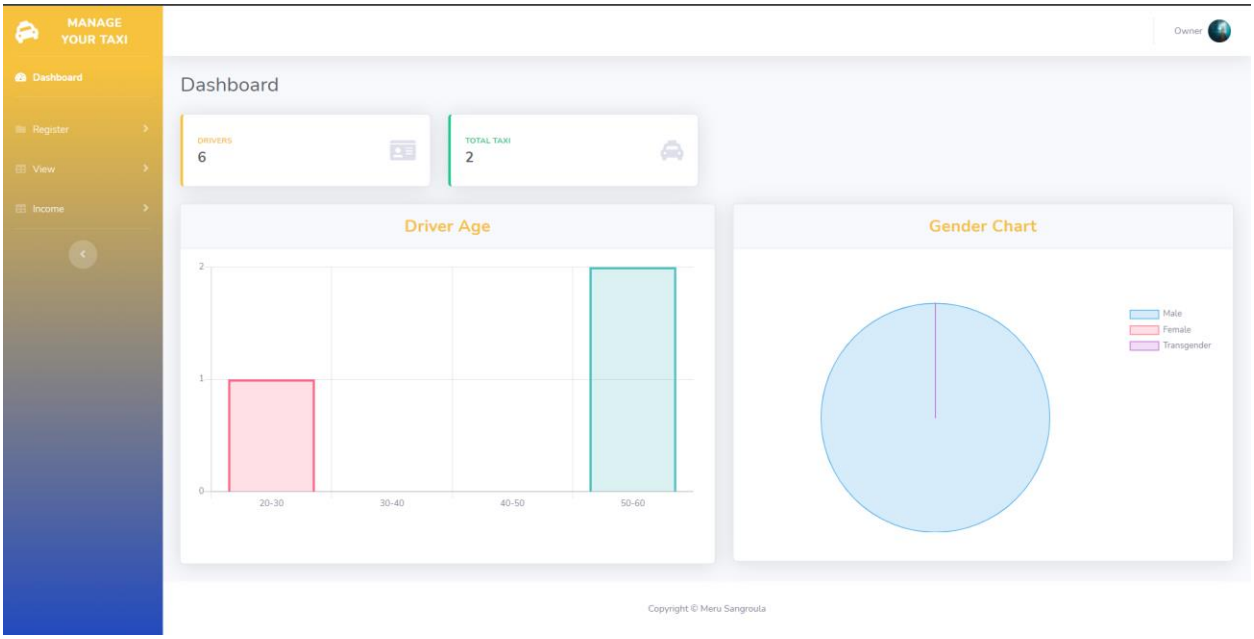


Figure 101 Owner Login Test (3)

Test No.	Action	Expected Outcome	Actual Outcome	Result
1.	Login Using valid Owner username and password	Login should be successful and dashboard page should be redirected.	Login was successful and user was redirected into dashboard page.	Passed.

Table 3 Owner Login Test

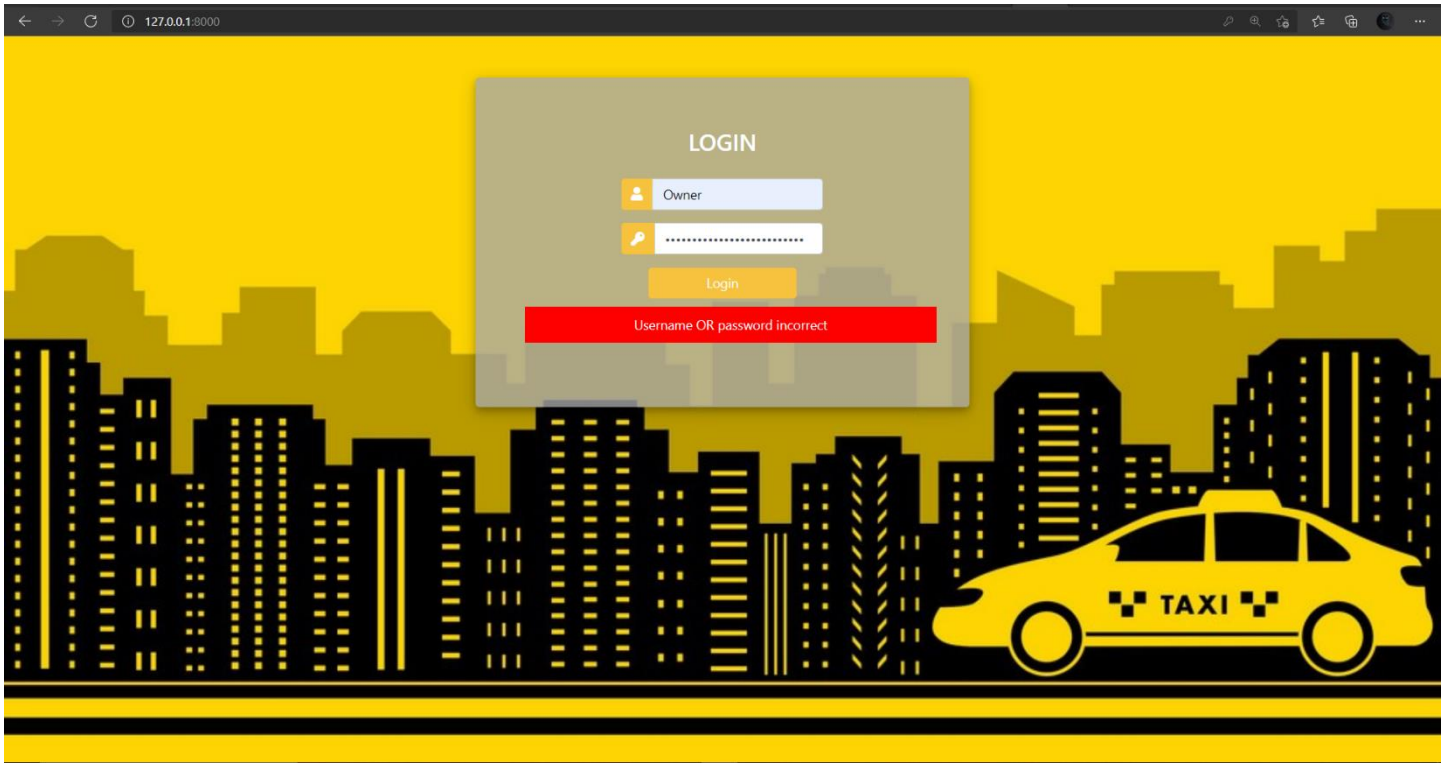


Figure 102 Owner Login Validation Test

Test No.	Action	Expected Outcome	Actual Outcome	Result
2.	Login Using invalid valid Owner username and password	An error message should be seen, and Login should not be successful.	An error message appeared.	Passed.

Table 4 Owner Login Validation Test

4.2.2 Register Taxi

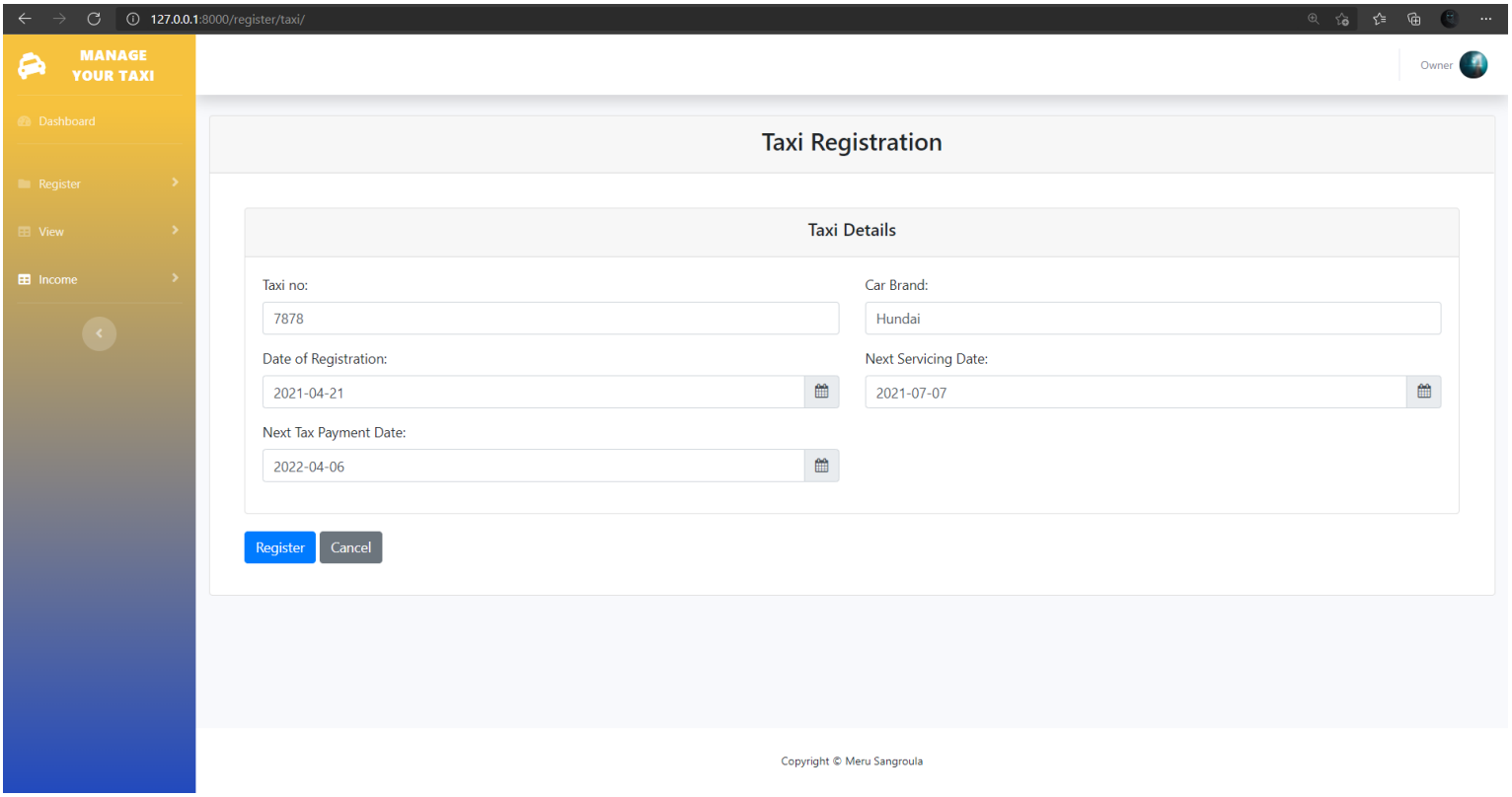


Figure 103 Taxi Register Test (1)

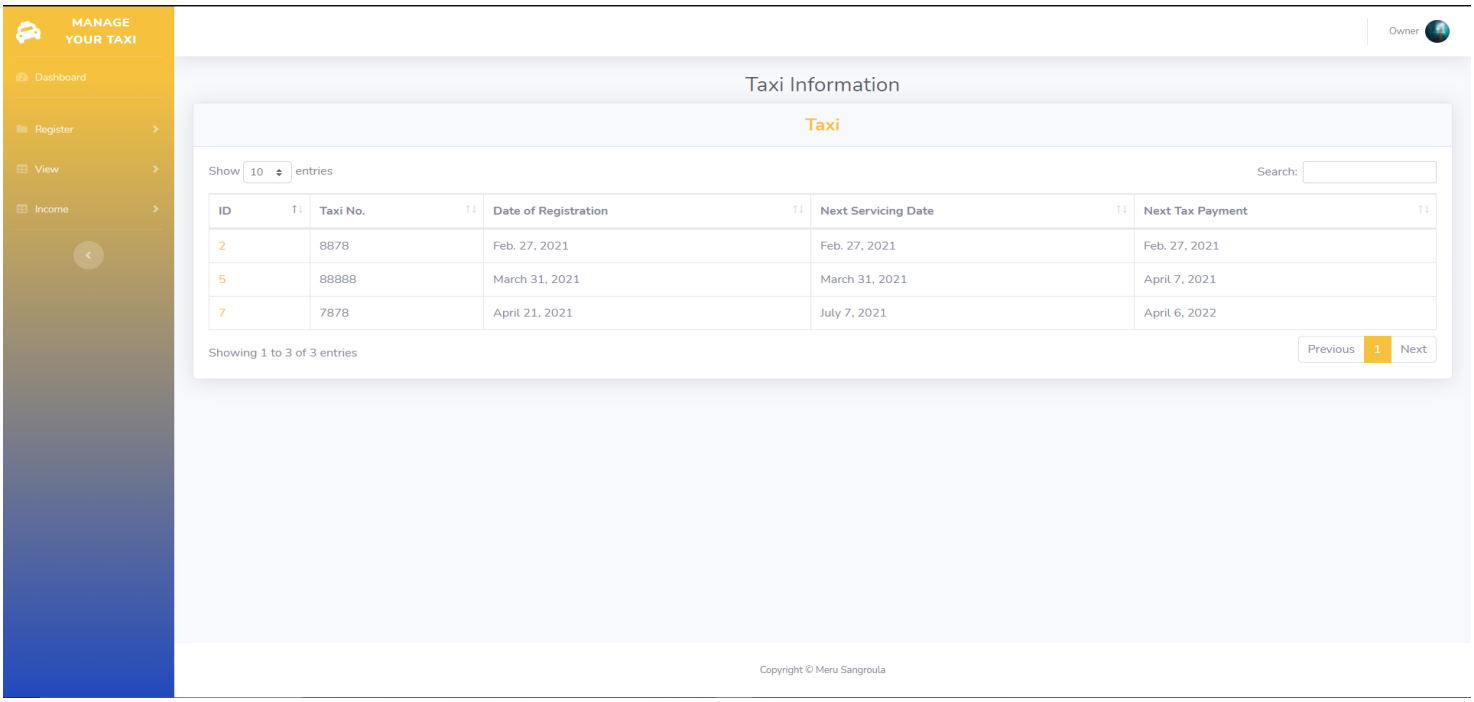


Figure 104 Taxi Register Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
3.	Filling taxi registration form and registering taxi.	After clicking the register button taxi should be registered and new page should be redirected.	The taxi was successfully registered, and new view taxi page was seen.	Passed.

Table 5 Taxi Register Test

4.2.3 View Taxi

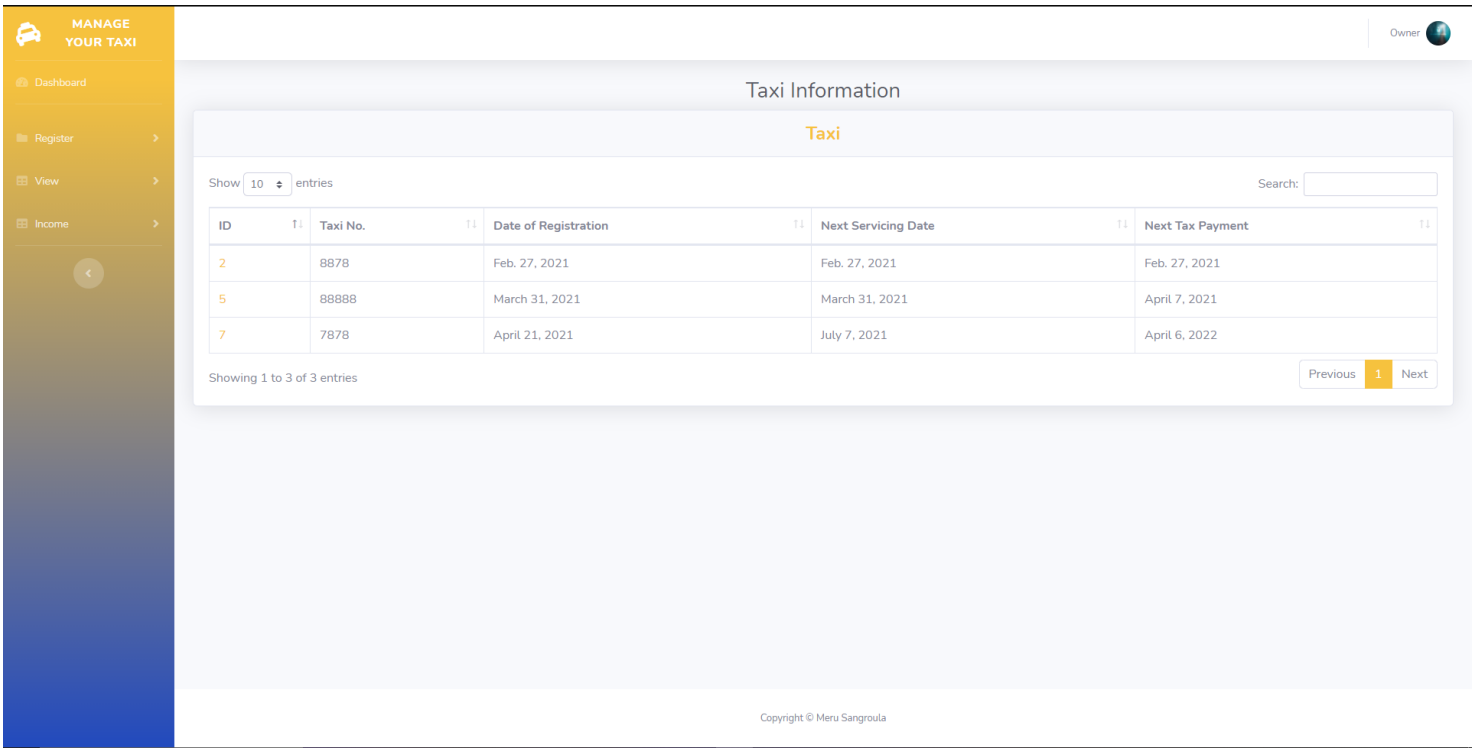


Figure 105 View Registered Taxi (Test 1)

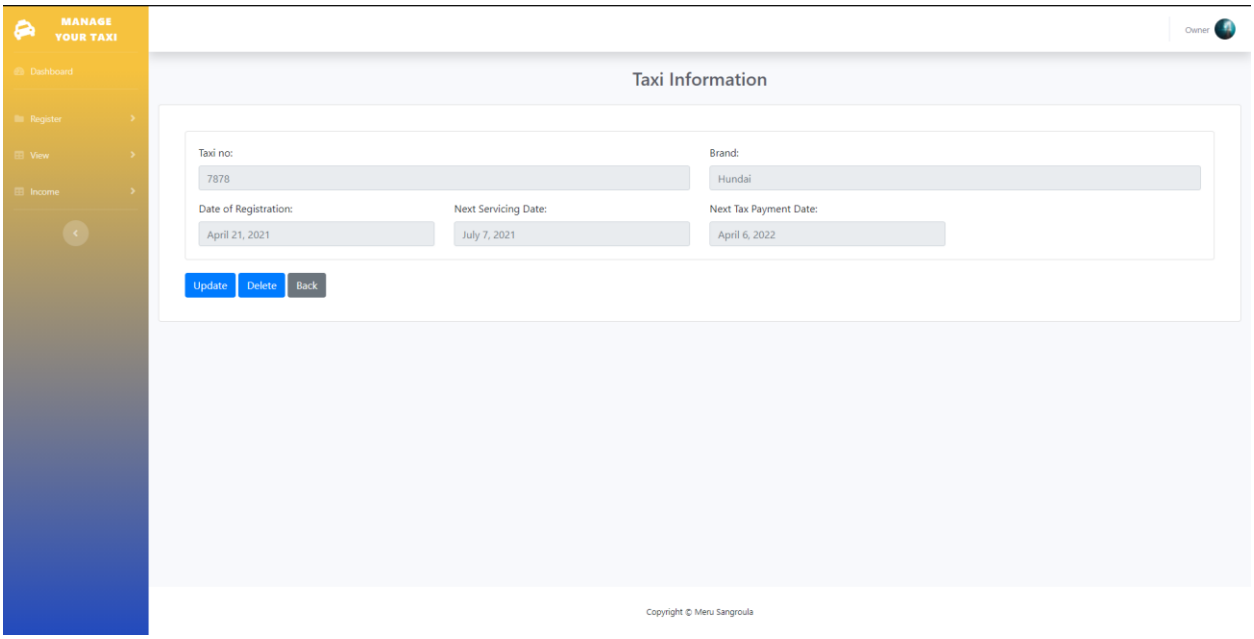


Figure 106 View Registered Taxi (Test 2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
4.	Clicking the registered taxi to view the previously filled form.	The same data should be available.	After clicking the registered taxi, the exact same data was seen.	Passed.

Table 6 View Registered Taxi Test

4.2.4 Update Taxi

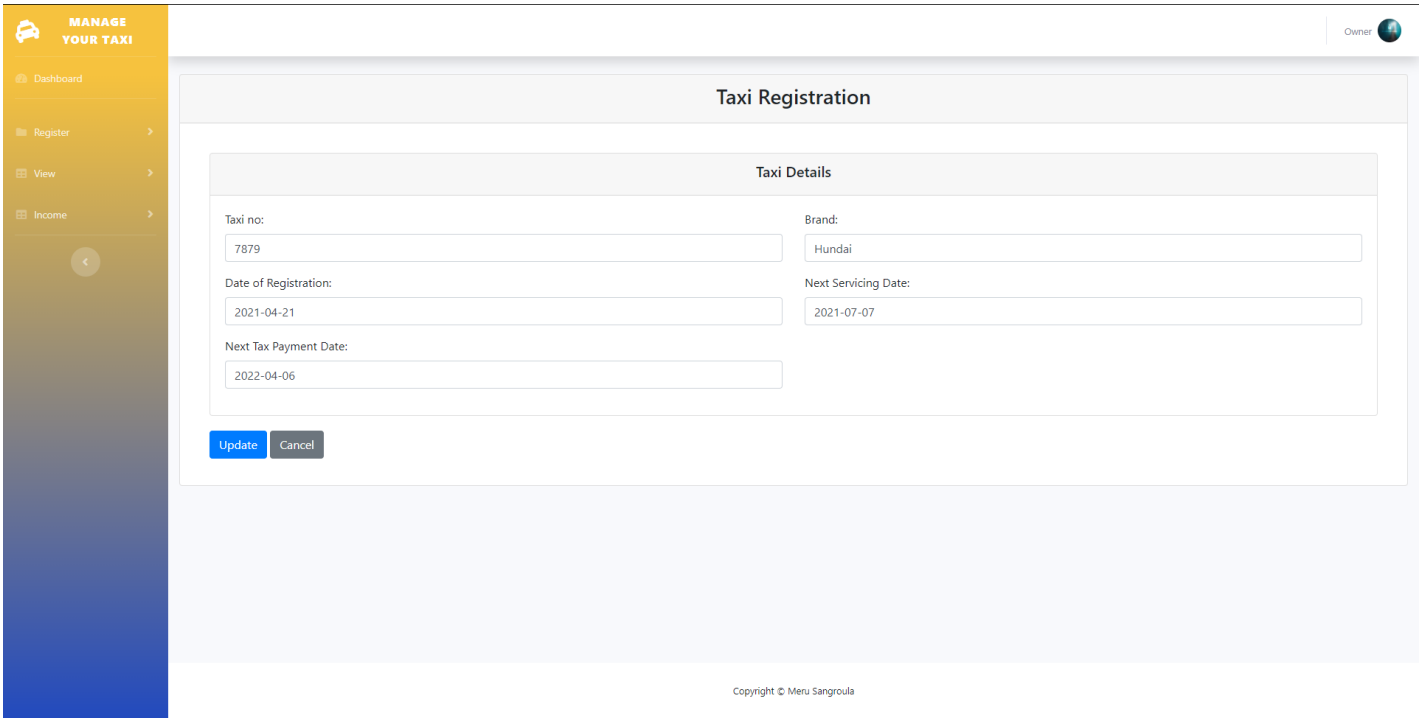


Figure 107 Update Registered Taxi (Test 1)

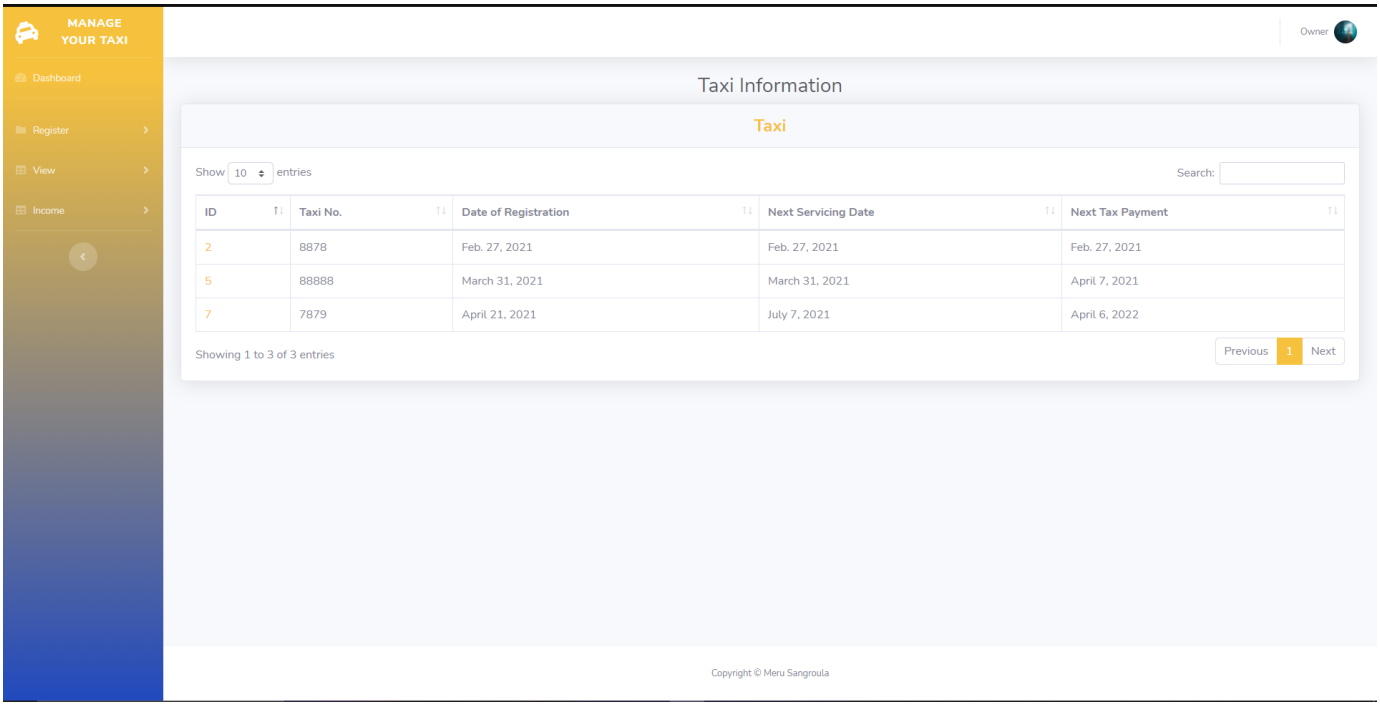


Figure 108 Update Registered Taxi (Test 2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
5.	Updating the taxi number and changing it to 7879.	The taxi number should be updated.	After clicking update, the taxi number was updated, and page was redirected to view page.	Passed.

Table 7 Update Registered Taxi Test

4.2.5 Delete Taxi

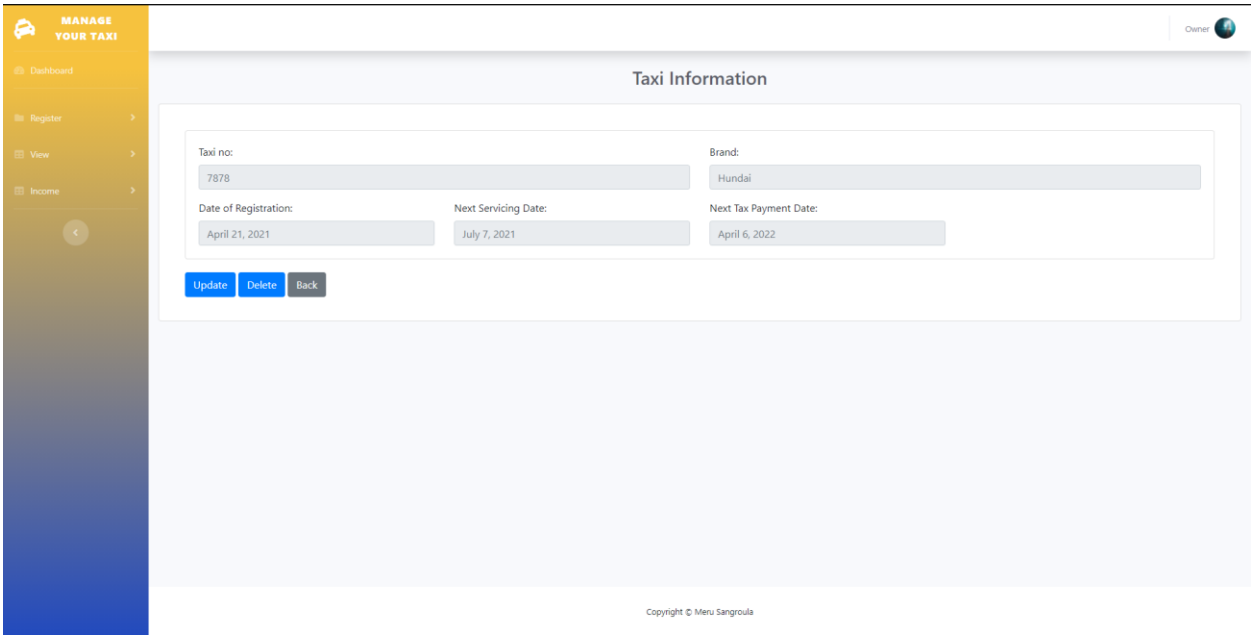


Figure 109 Delete Taxi Test (1)

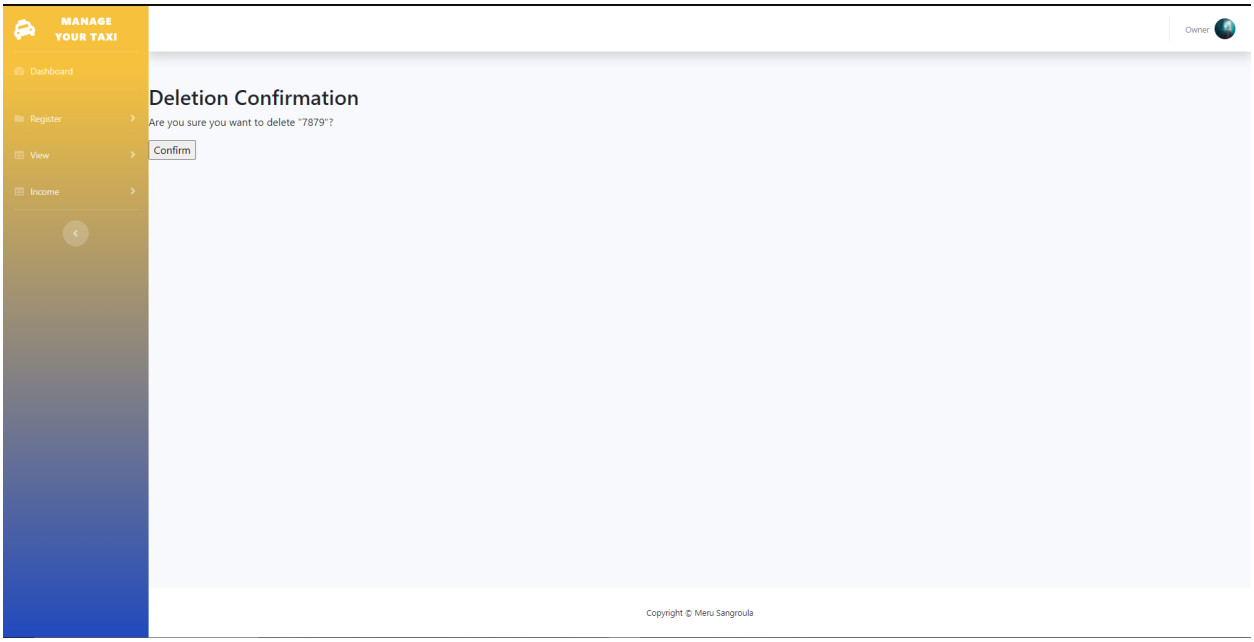


Figure 110 Delete Taxi Test (2)

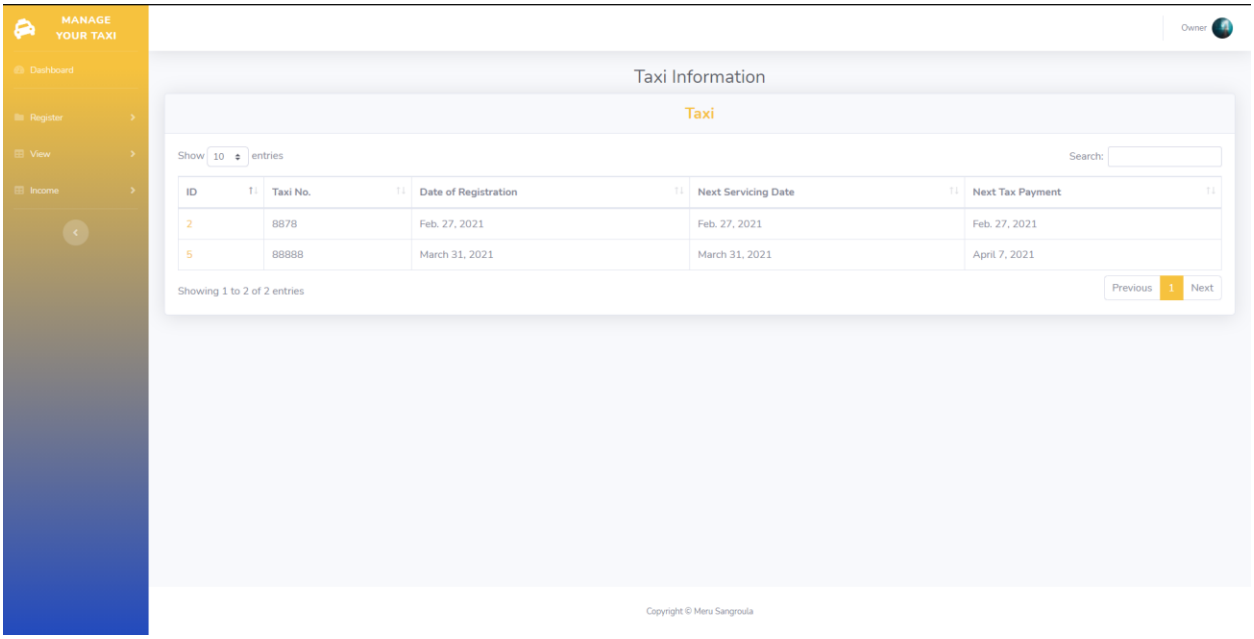


Figure 111 Delete Taxi Test (3)

Test No.	Action	Expected Outcome	Actual Outcome	Result
6.	Deleting the registered taxi.	The registered taxi should be deleted.	After clicking confirm delete, the taxi was deleted.	Passed.

Table 8 Delete Taxi Test

4.2.6 Register Driver

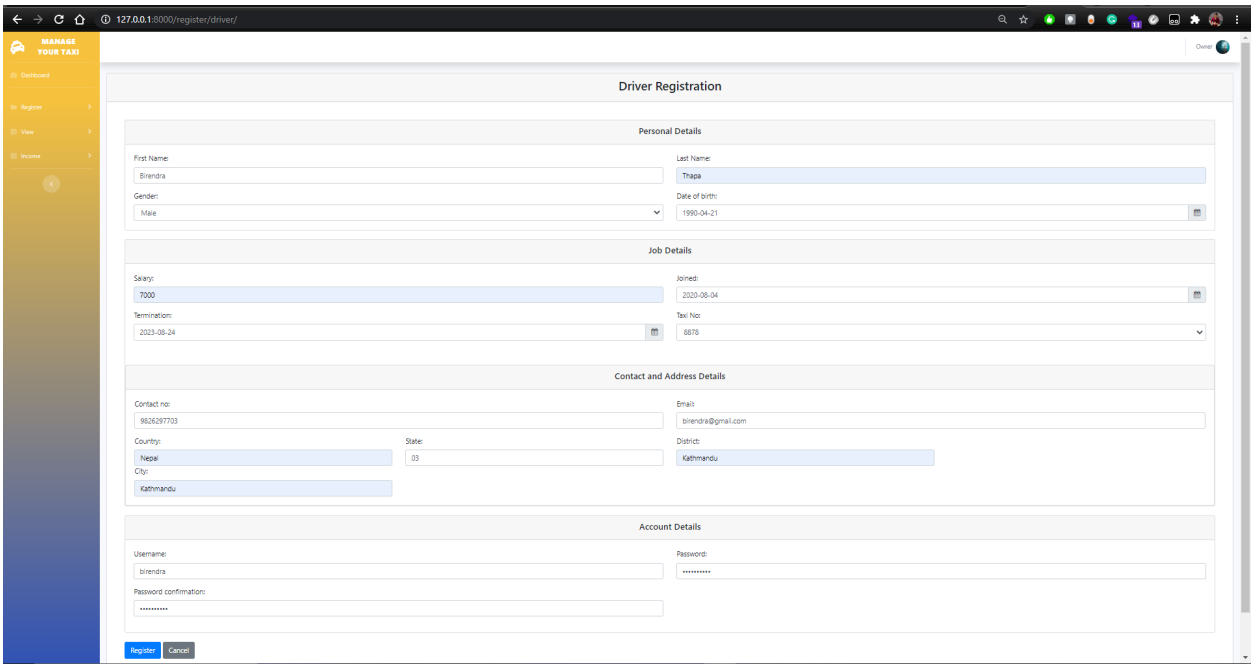


Figure 112 Register Driver Test (1)

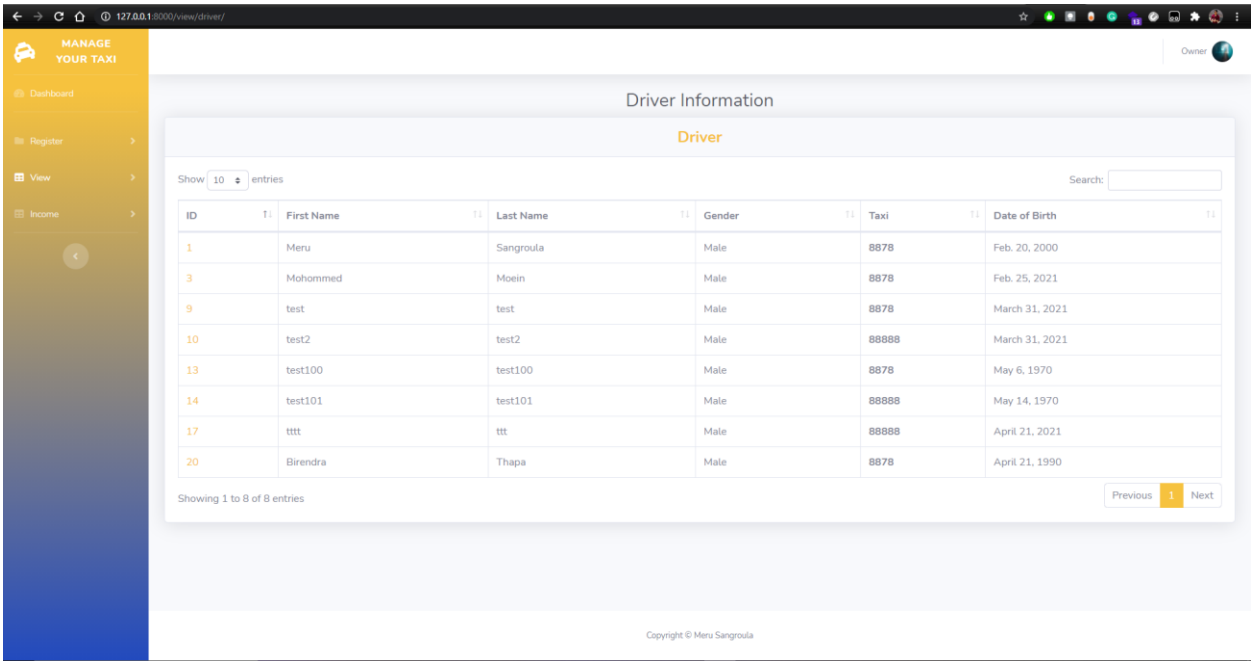


Figure 113 Register Driver Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
7.	Filling driver registration form and registering driver.	After clicking the register button driver should be registered and new page should be redirected.	The driver was successfully registered, and new view driver page was seen.	Passed.

Table 9 Register Driver Test

4.2.7 View Driver

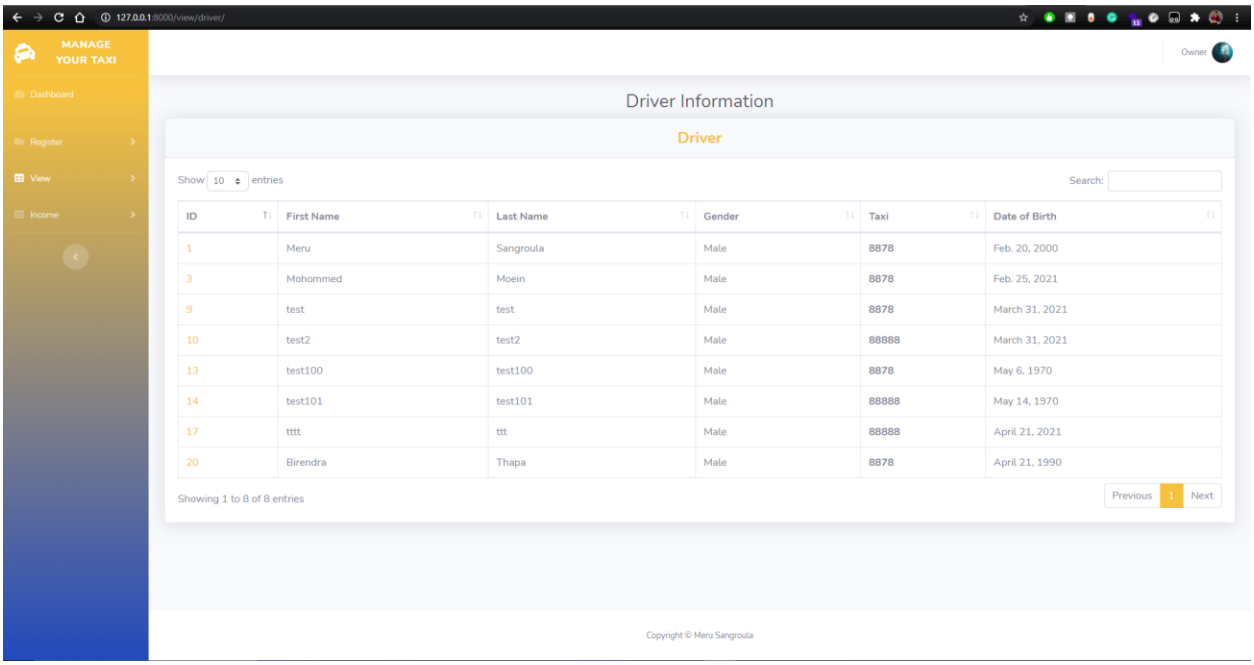


Figure 114 View Driver Test (1)

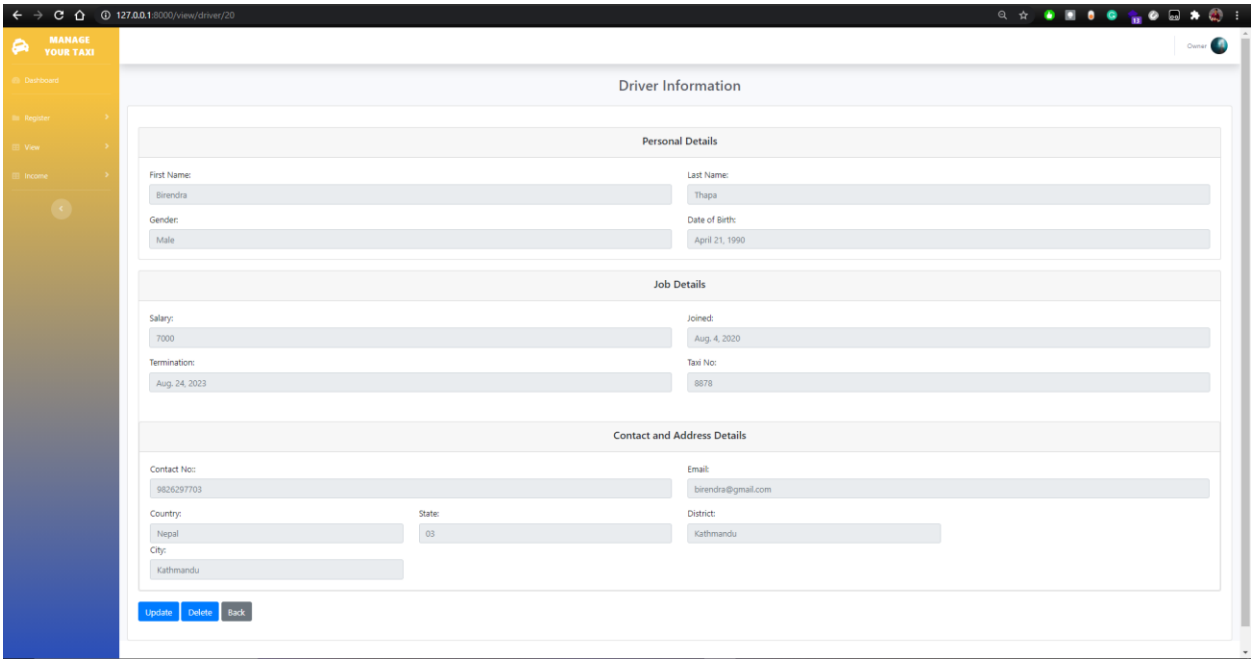


Figure 115 View Driver Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
8.	Clicking the registered driver to view the previously filled form.	The same data should be available.	After clicking the registered driver, the exact same data was seen.	Passed.

Table 10 View Driver Test

4.2.8 Update Driver

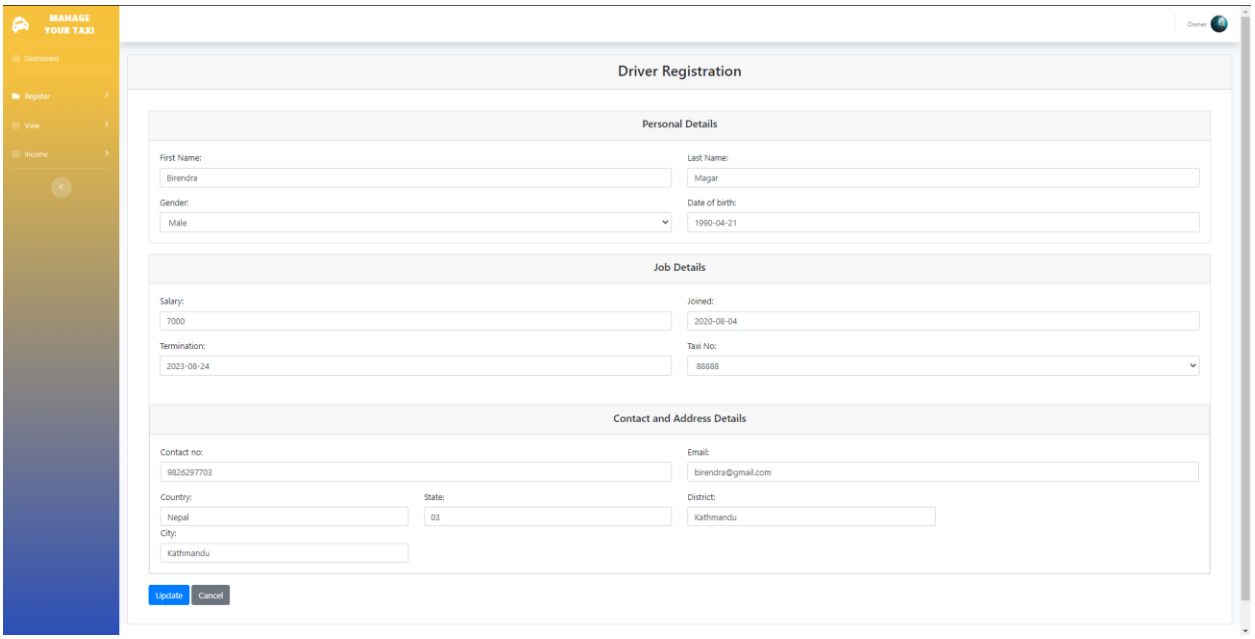


Figure 116 Update Driver Test (1)

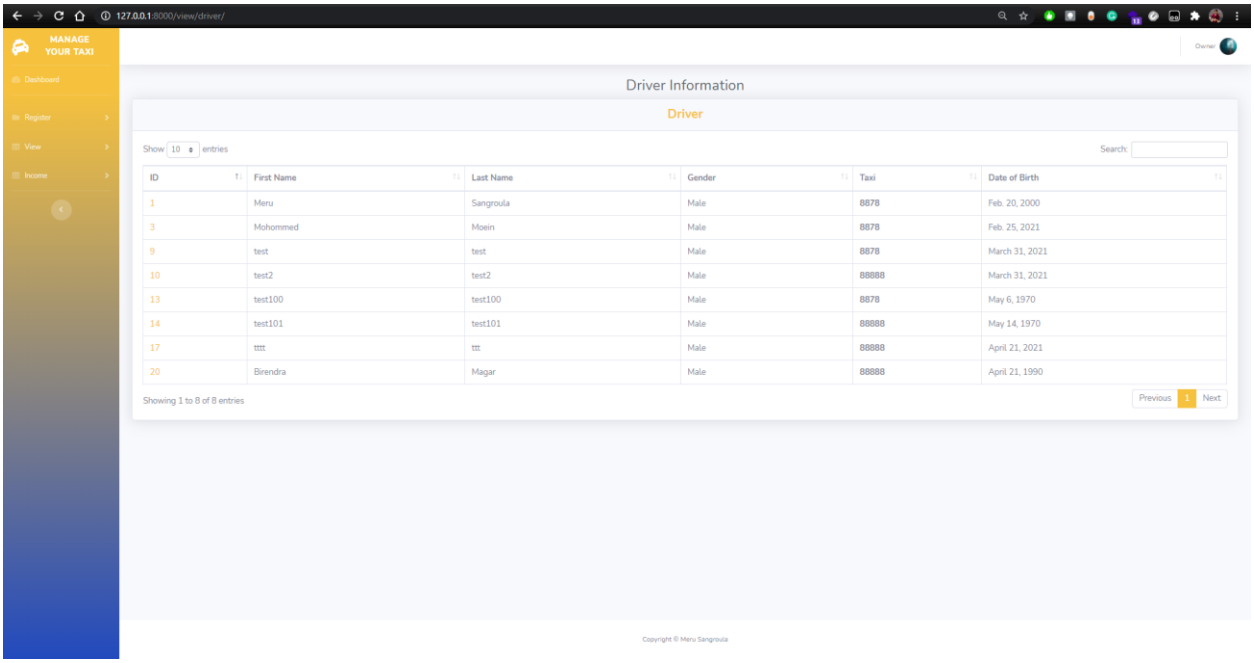


Figure 117 Update Driver Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
9.	Updating the driver last name and taxi no.	The last name and taxi number should be updated.	After clicking update, the last name and taxi number of the driver was updated.	Passed.

Table 11 Update Driver Test

4.2.9 Delete Driver

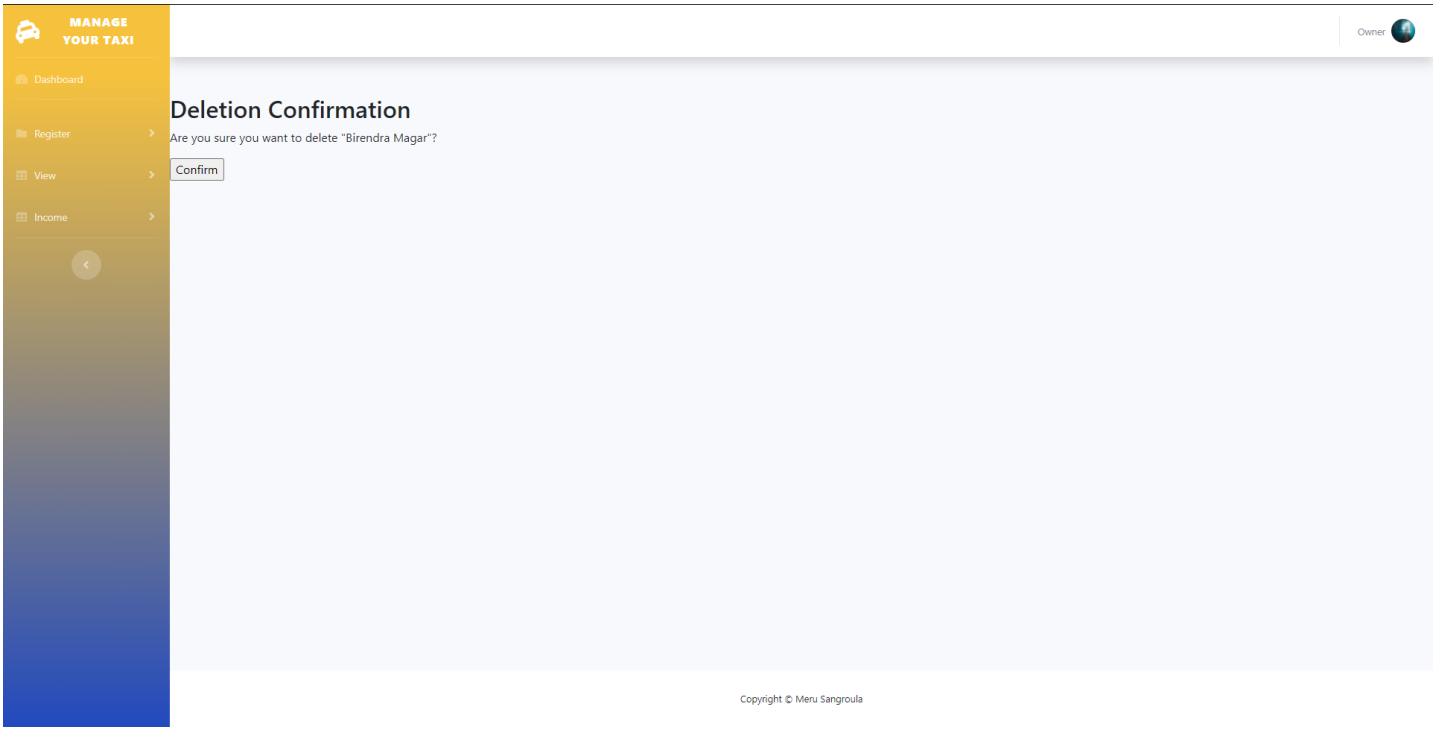


Figure 118 Delete Driver Test (1)

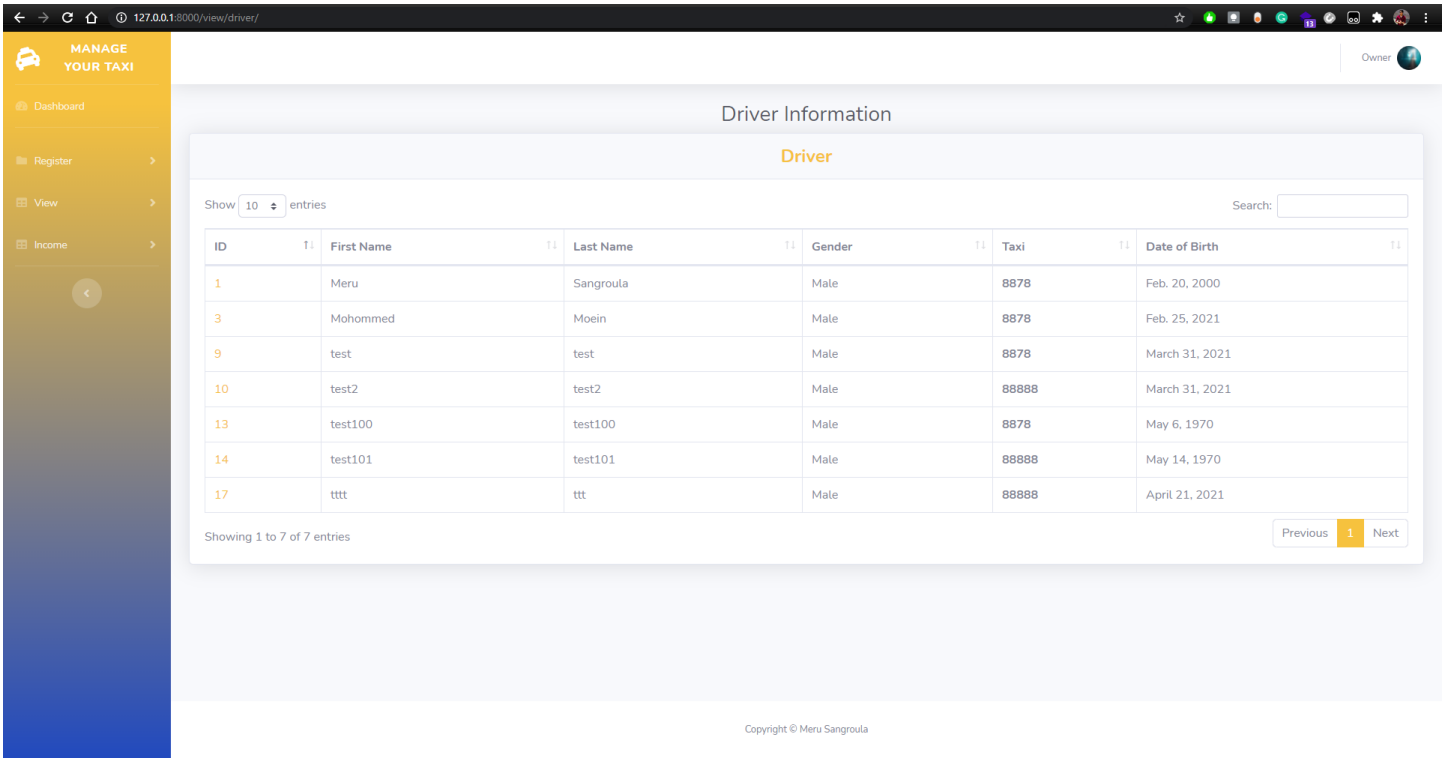


Figure 119 Delete Driver Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
10.	Deleting the registered driver.	The registered driver should be deleted.	After clicking confirm delete, the driver was deleted.	Passed.

Table 12 Delete Driver Test

4.2.10 Add Income

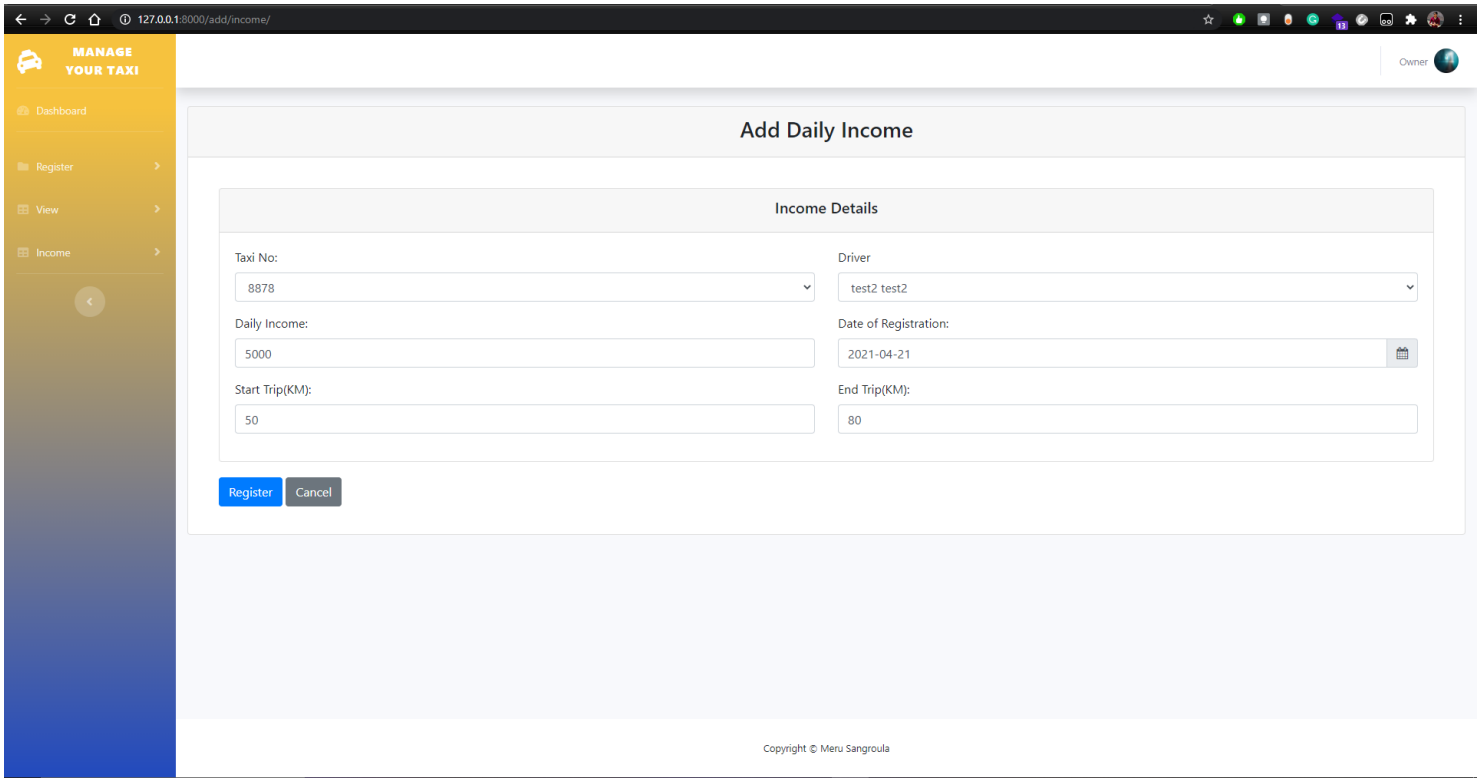


Figure 120 Add Income Test (1)

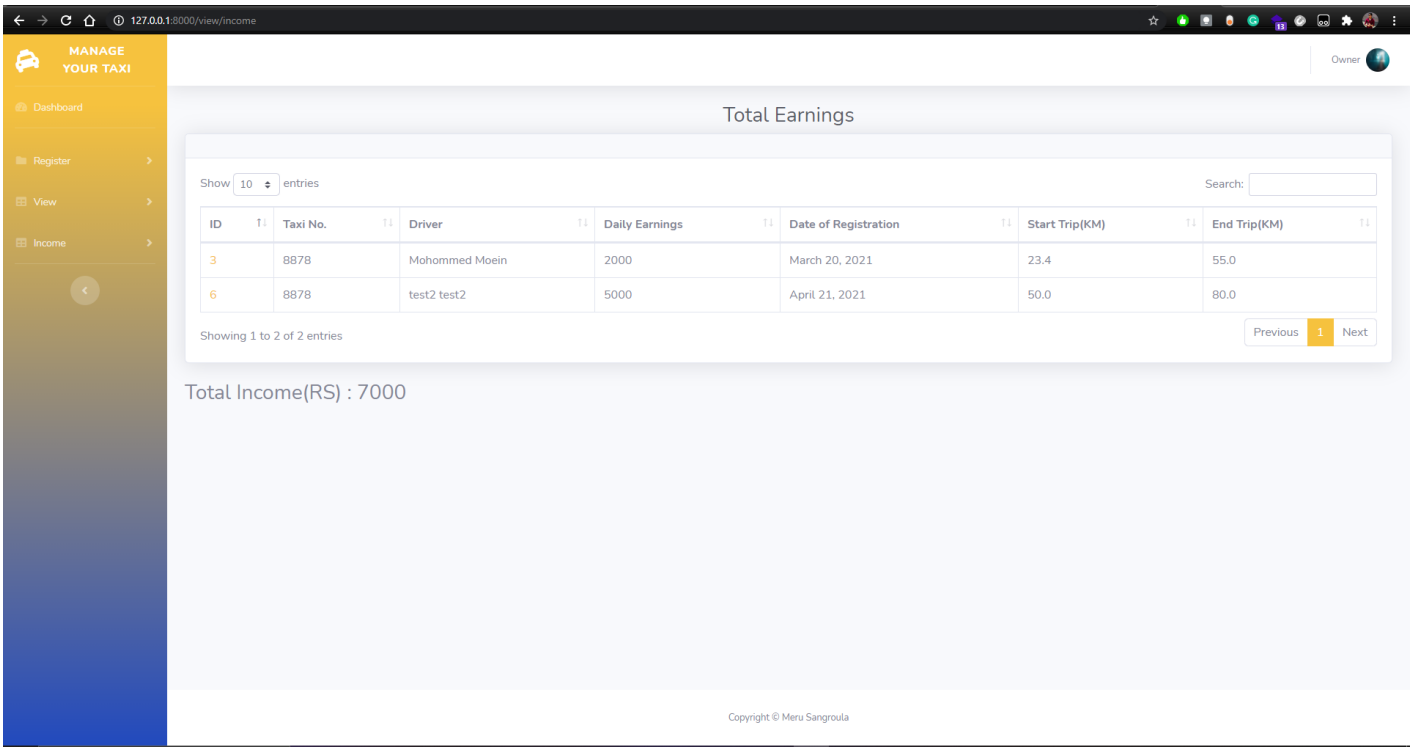


Figure 121 Add Income Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
11.	Adding the daily income earned by a driver in the system.	The income should be added.	After clicking register, the income was added and was redirected to view income page.	Passed.

Table 13 Add Income Test

4.2.11 View Income

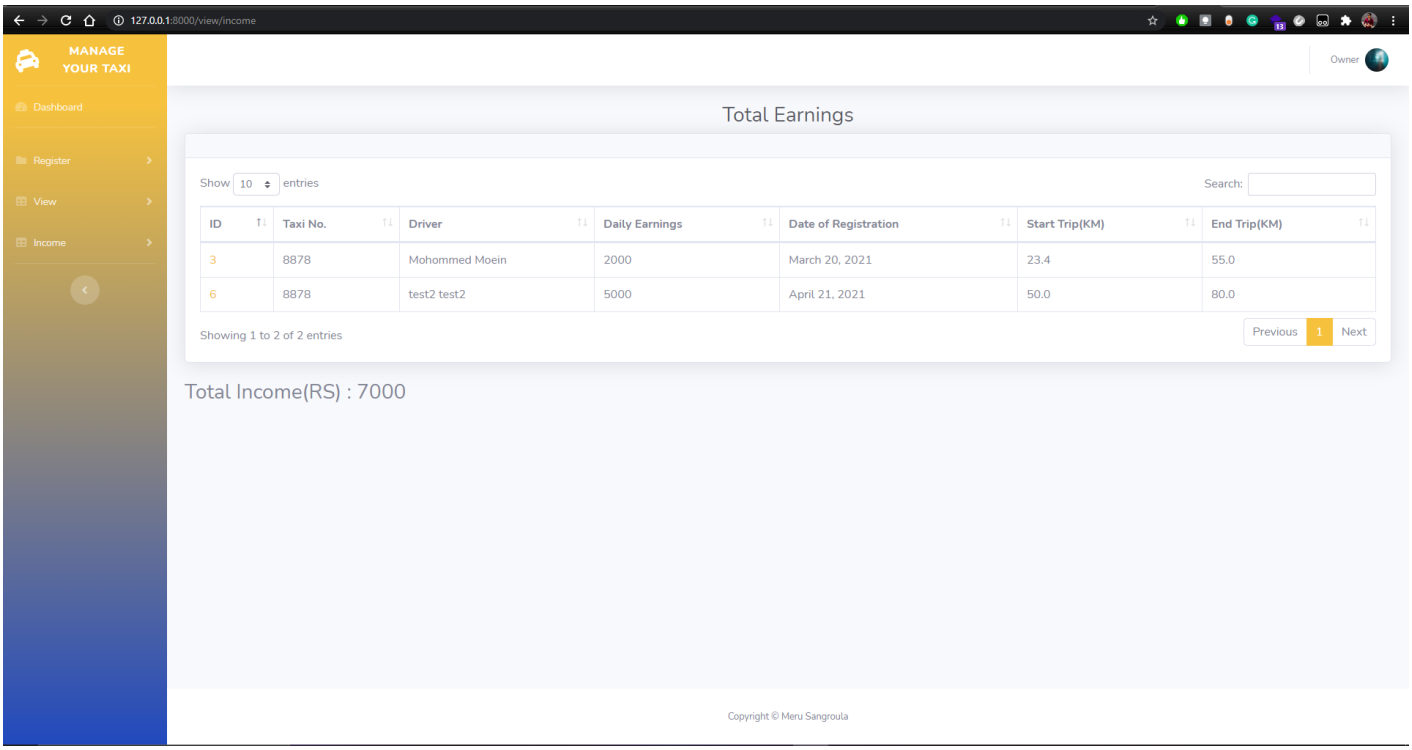


Figure 122 View Income Test

Test No.	Action	Expected Outcome	Actual Outcome	Result
12.	Viewing the total income generated after the income is added.	The income should be added, and total income should be changed.	After the income was added the income changed and before it was 2000 and now it is 5000.	Passed.

Table 14 View Income Test

4.2.12 Update Income

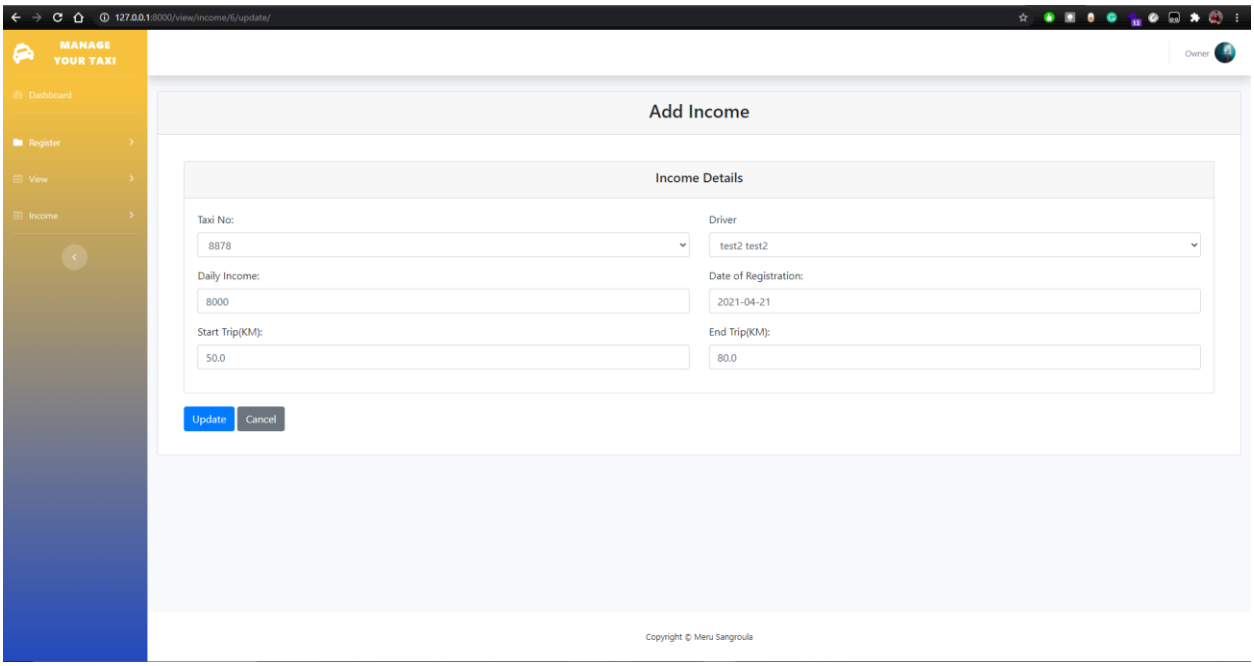


Figure 123 Update Income Test (1)

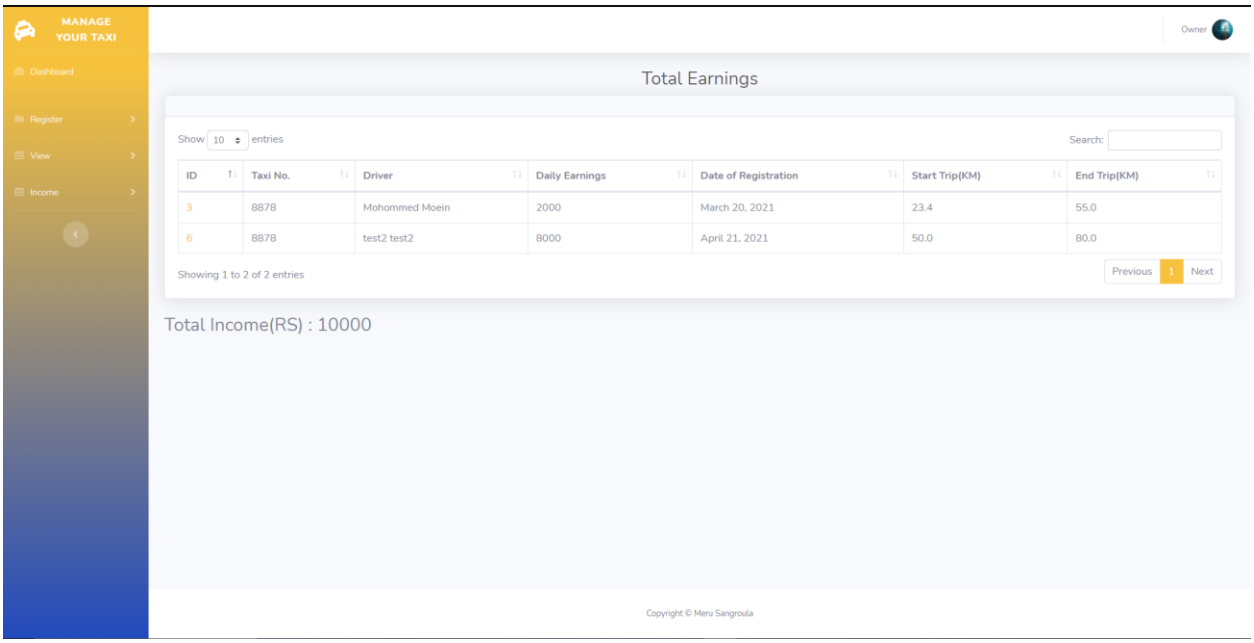


Figure 124 Update Income Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
13.	Updating the Income of Driver.	The income should be updated and total income should also be added.	After the daily income was updated the total income in the view also got updated.	Passed.

Table 15 Update Income Test

4.2.13 Delete Income

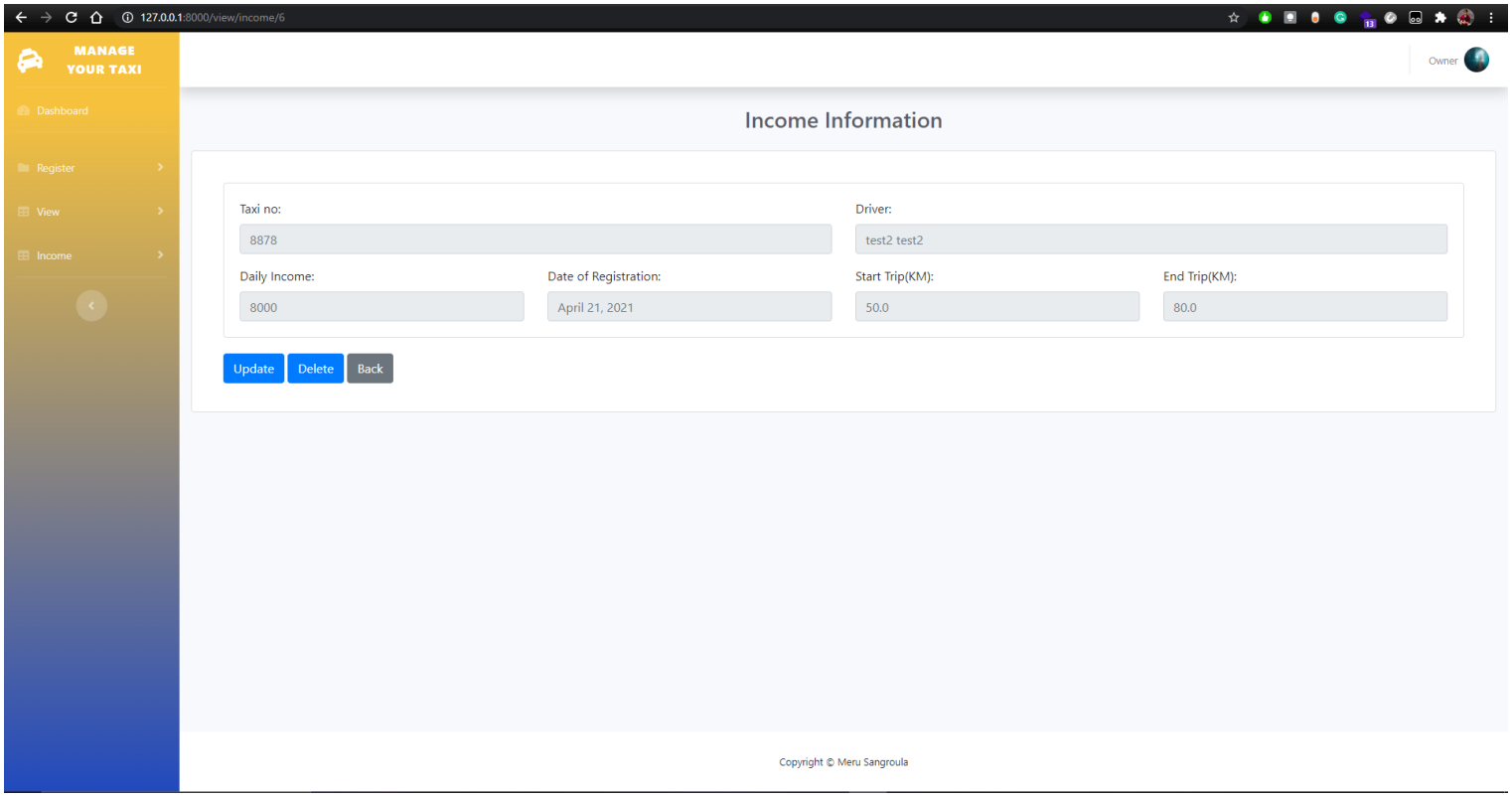


Figure 125 Delete Income Test (1)

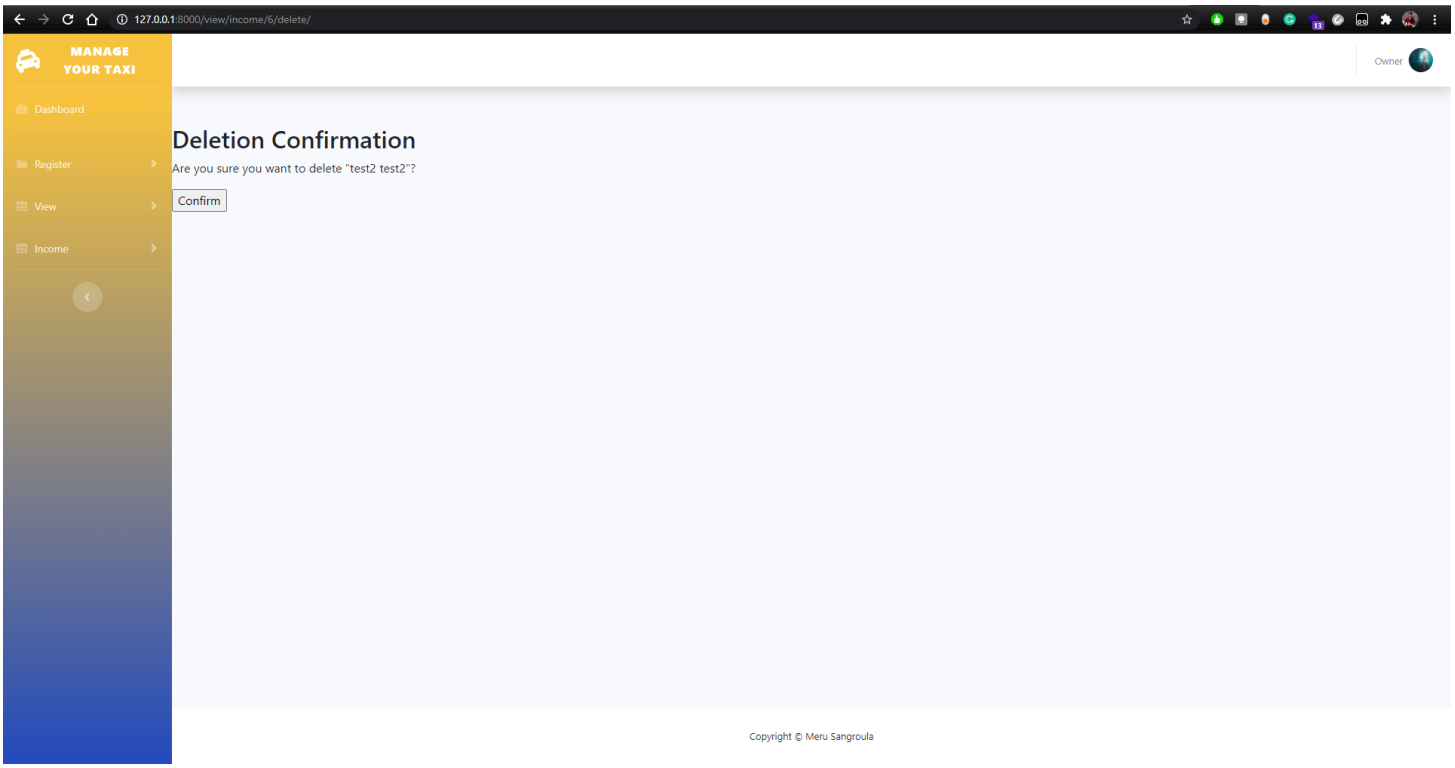


Figure 126 Delete Driver Test (2)

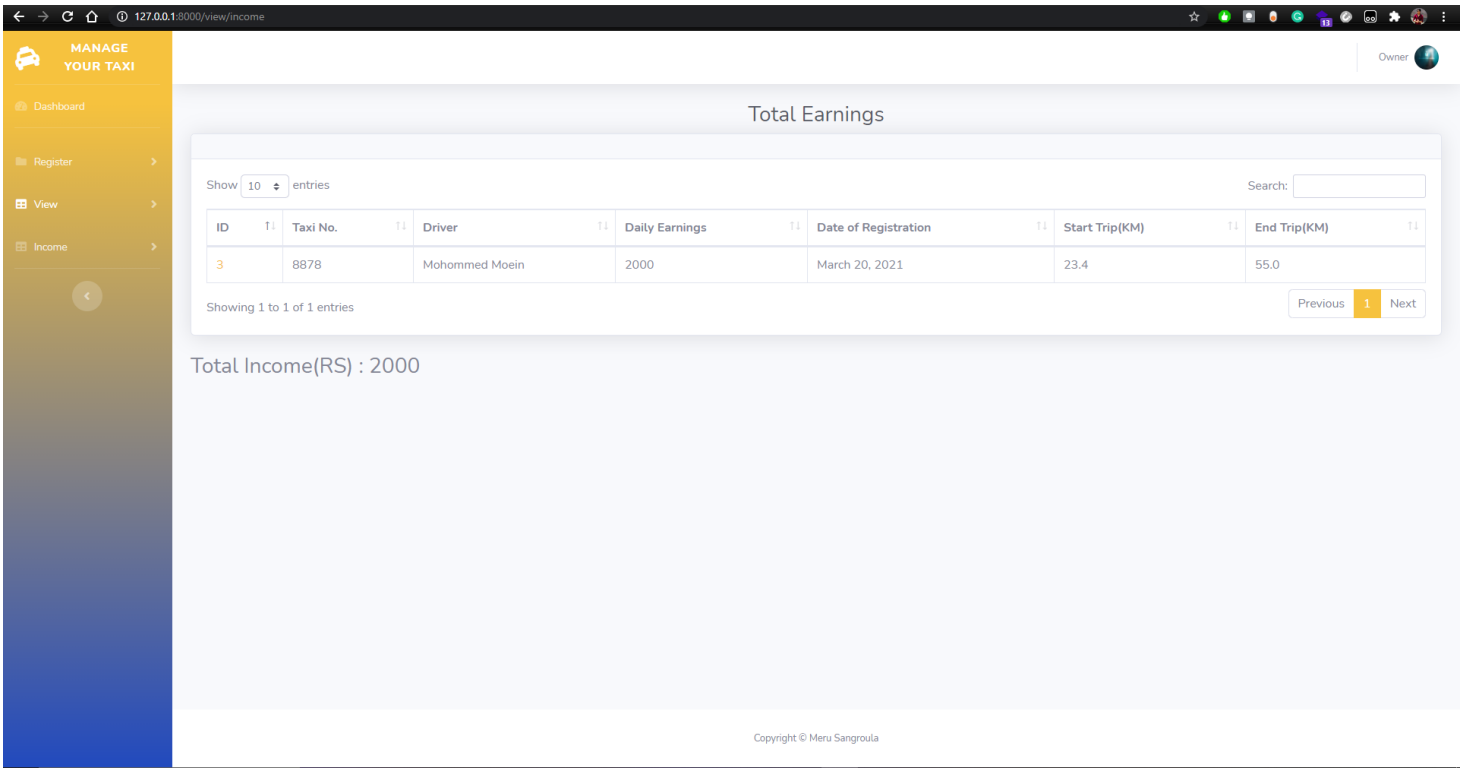


Figure 127 Delete Driver Test (3)

Test No.	Action	Expected Outcome	Actual Outcome	Result
14.	Deleting the Income of Driver.	The income should be deleted, and total income should also decrease.	After the daily income was deleted the total income in the view also got decreased.	Passed.

Table 16 Delete Driver Test

4.2.14 Driver Age Chart

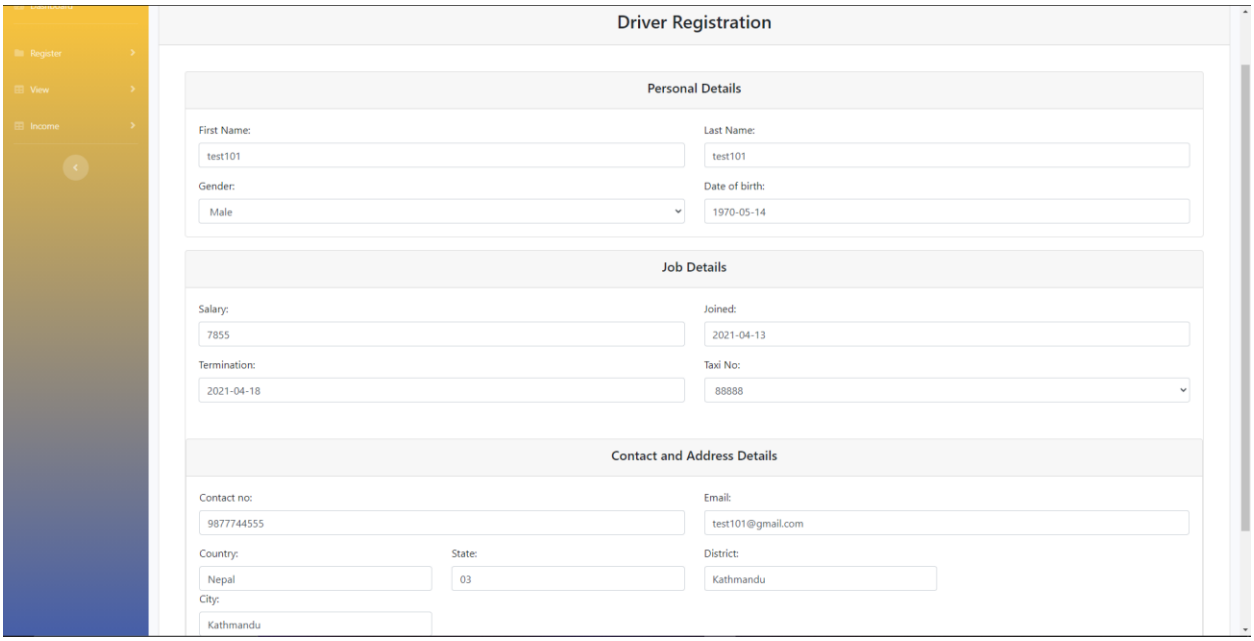


Figure 128 Driver Age Chart Test (1)

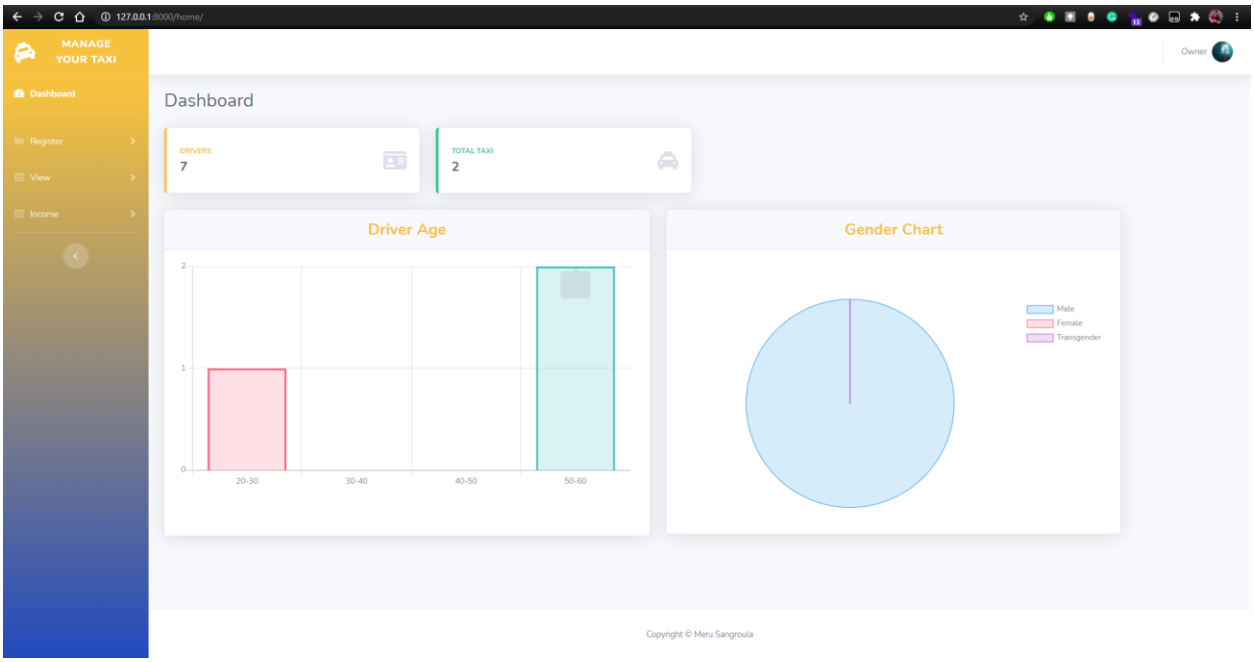


Figure 129 Driver Age Chart Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
15.	Adding a 51 years old driver to the database.	The driver age chart should show driver in 50-60 section.	The driver age chart showed driver in 50-60 section.	Passed.

Table 17 Driver Age Chart Test

4.2.15 Owner Logout

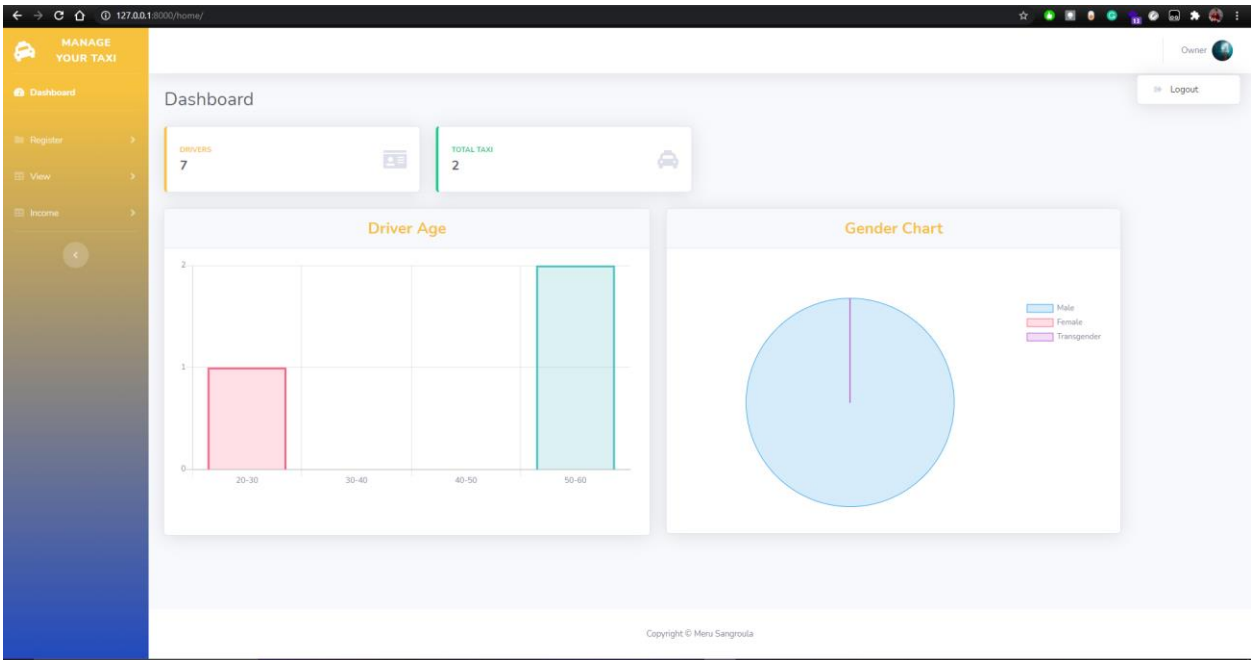


Figure 130 Owner Logout Test (1)

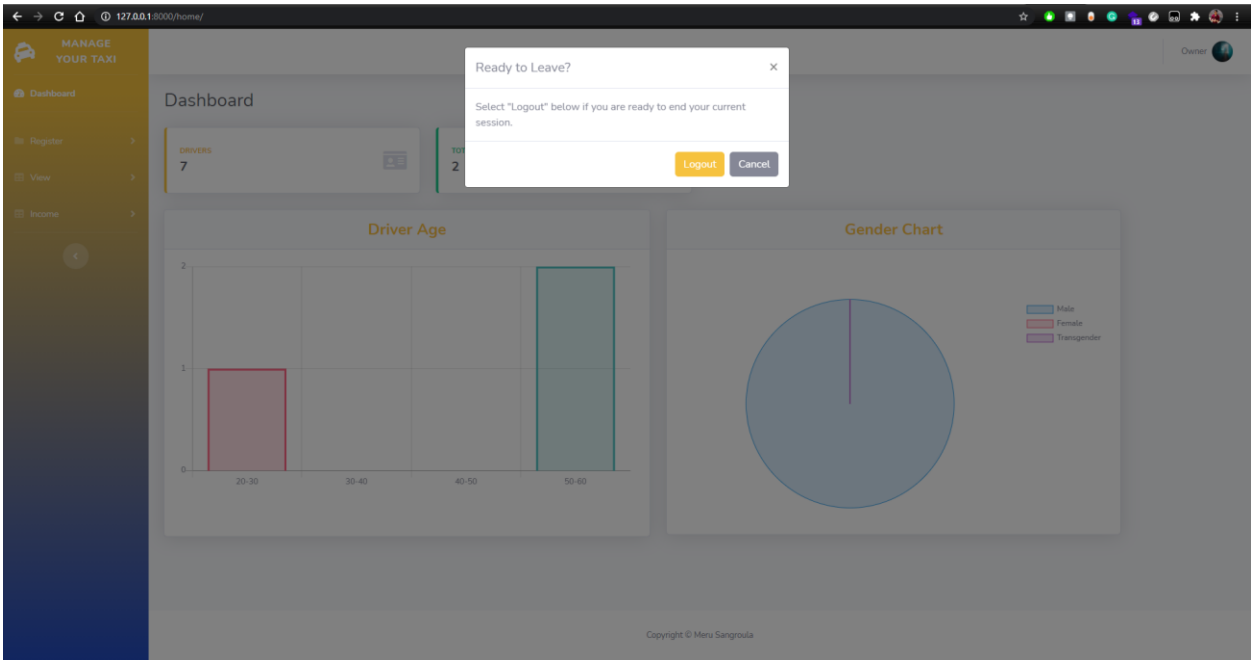


Figure 131 Owner Logout Test (2)

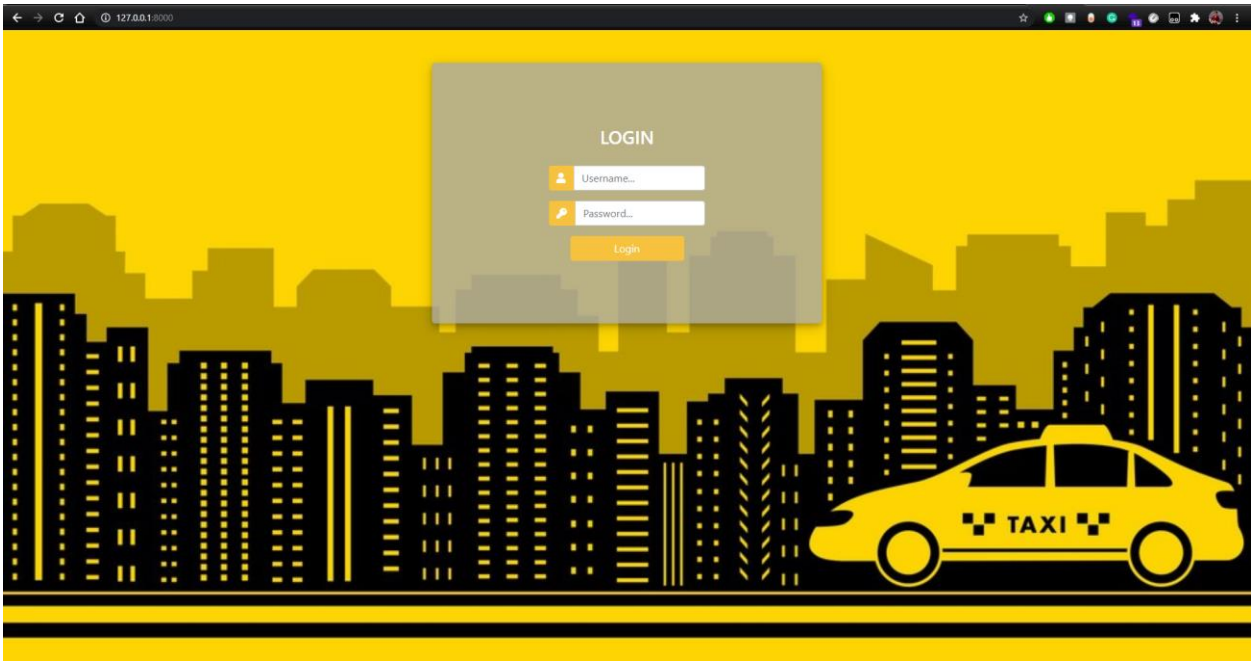


Figure 132 Owner Logout Test (3)

Test No.	Action	Expected Outcome	Actual Outcome	Result
16.	Logging out from owner dashboard.	The owner should be logged out and login page should appear.	Logout worked successfully and login page was available.	Passed.

Table 18 Owner Logout Test

4.2.16 Driver Login and Password Change

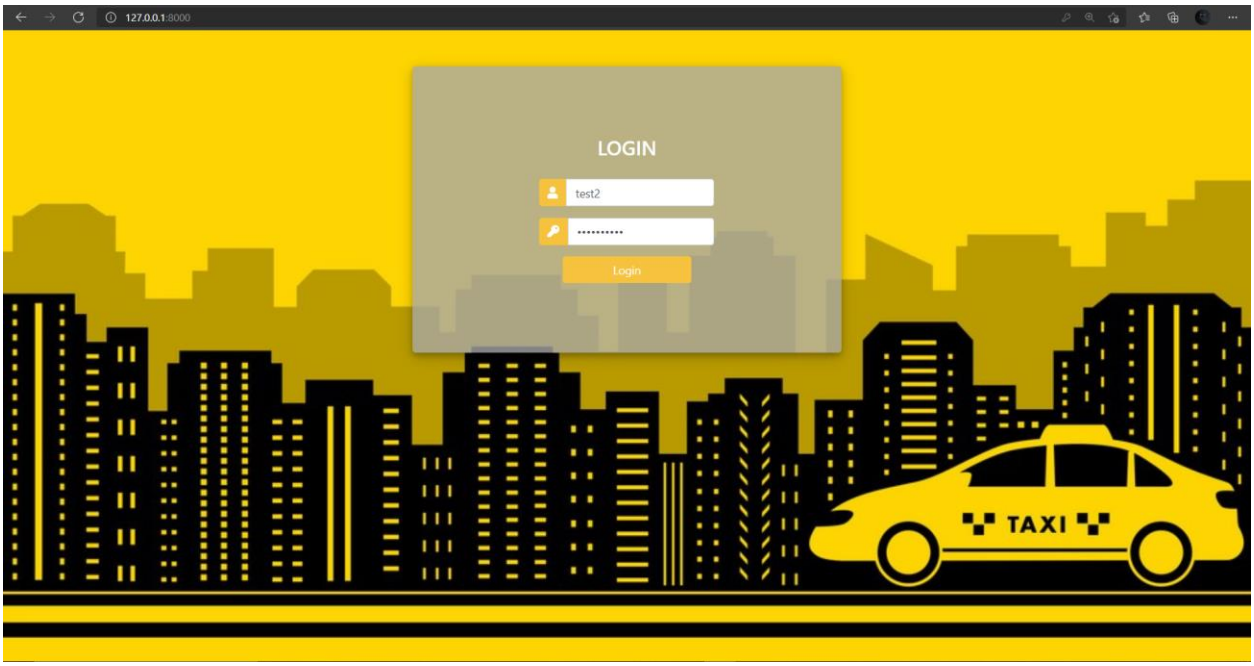


Figure 133 Driver Login Test (1)

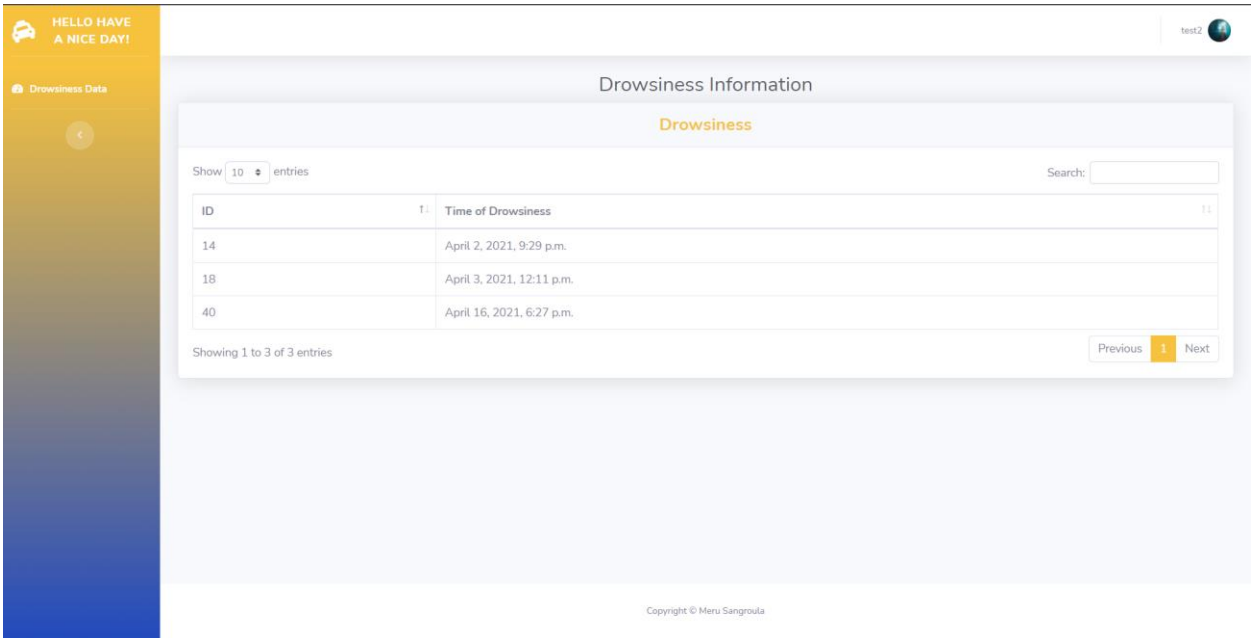


Figure 134 Driver Login Test (2)

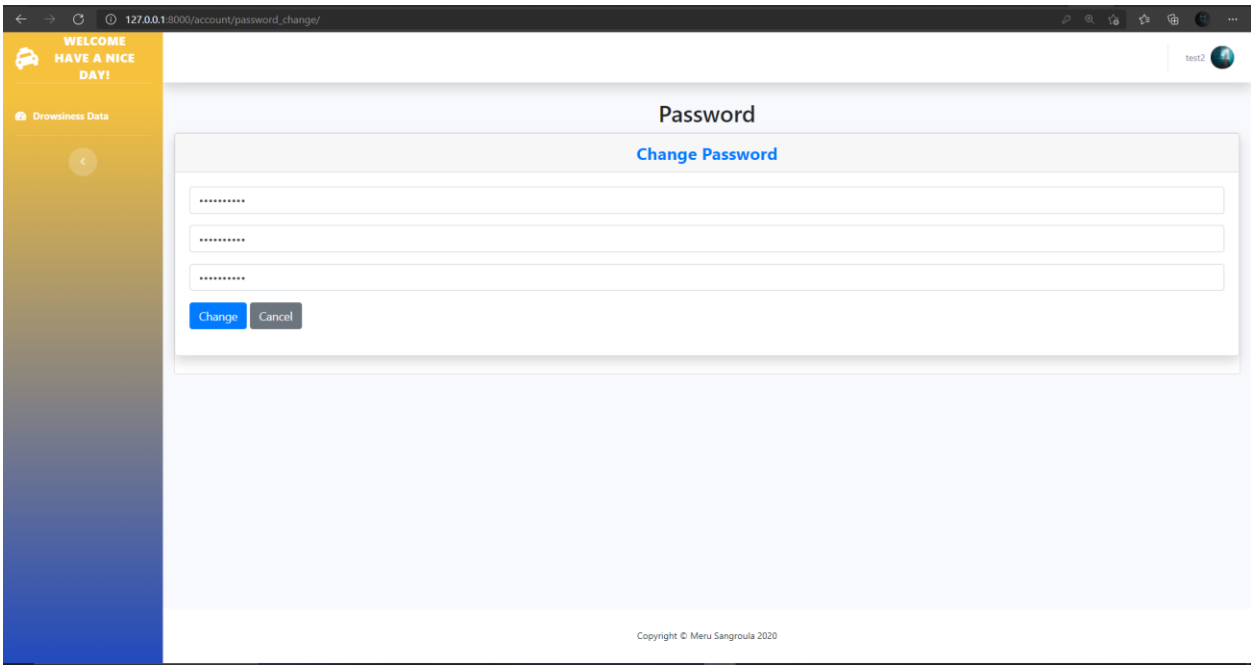


Figure 135 Change Password Test (1)

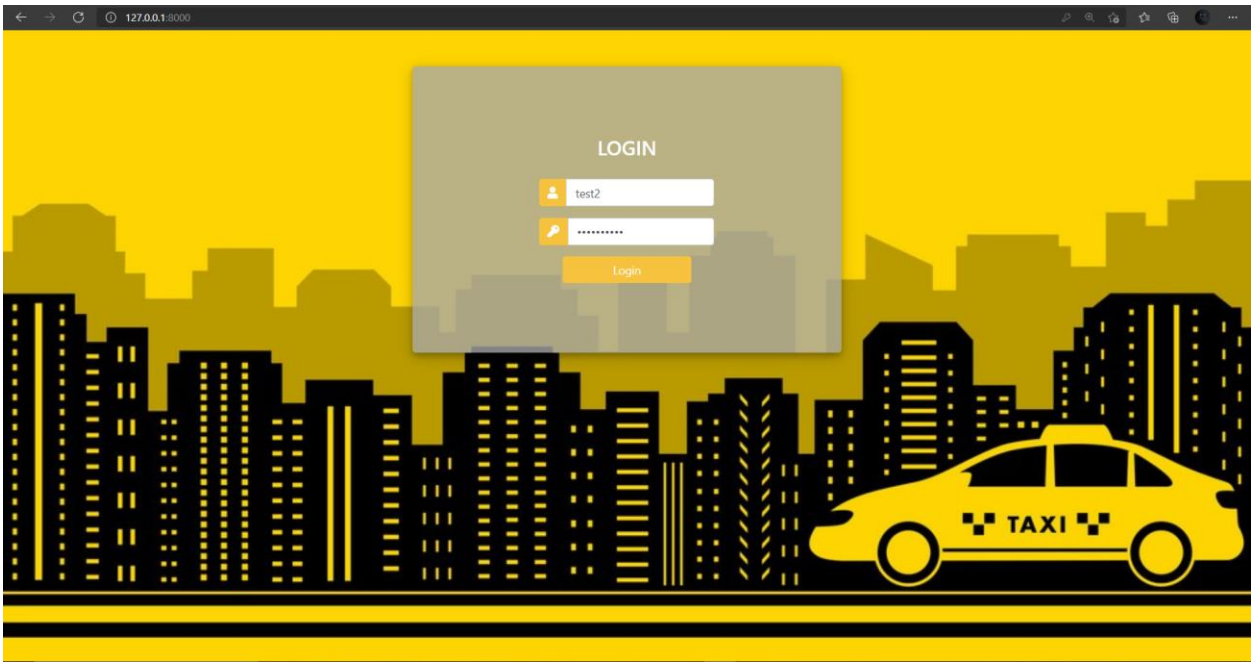


Figure 136 Change Password Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
17.	Logging out from owner dashboard.	The owner should be logged out and login page should appear.	Logout worked successfully and login page was available.	Passed.

Table 19 Change Password Test

4.2.17 Driver Login (Mobile)

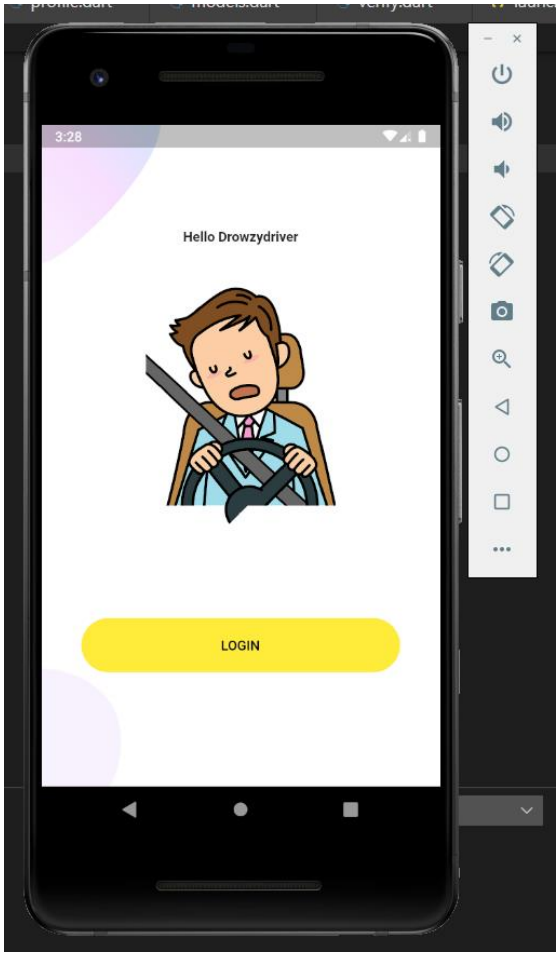
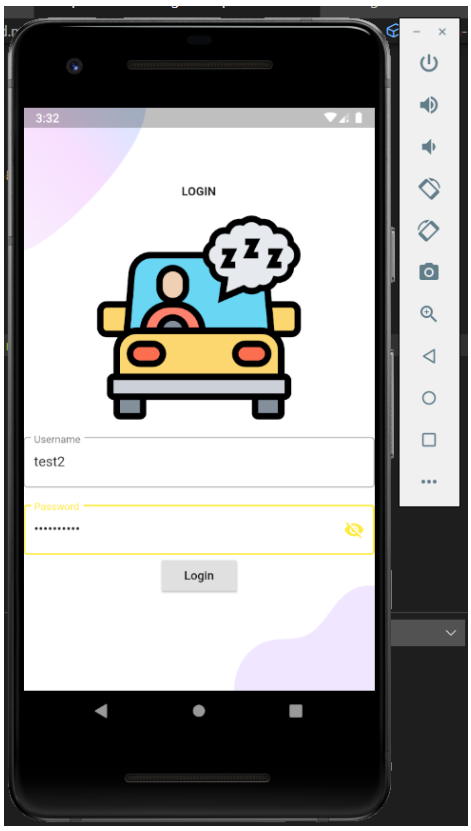


Figure 137 Driver Login Test (1)


3:32
LOGIN
Username
test2
Password
Login

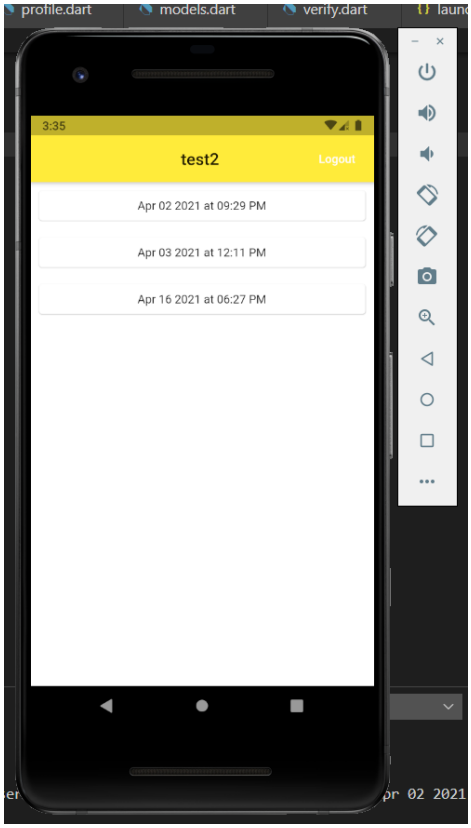


Figure 138 Driver Login Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
18.	Driver login using valid driver username and password.	Driver should be logged in and drowsiness data should be seen.	Driver was successfully logged in.	Passed.

Table 20 Driver Login Test

4.2.18 Driver Drowsiness Data (Mobile)

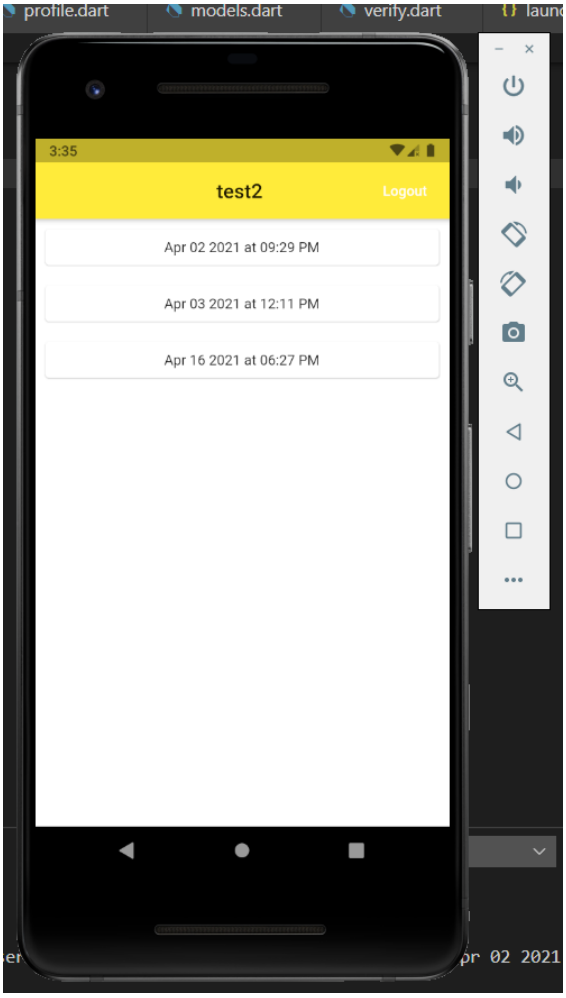


Figure 139 Driver Drowsiness Data Test (1)

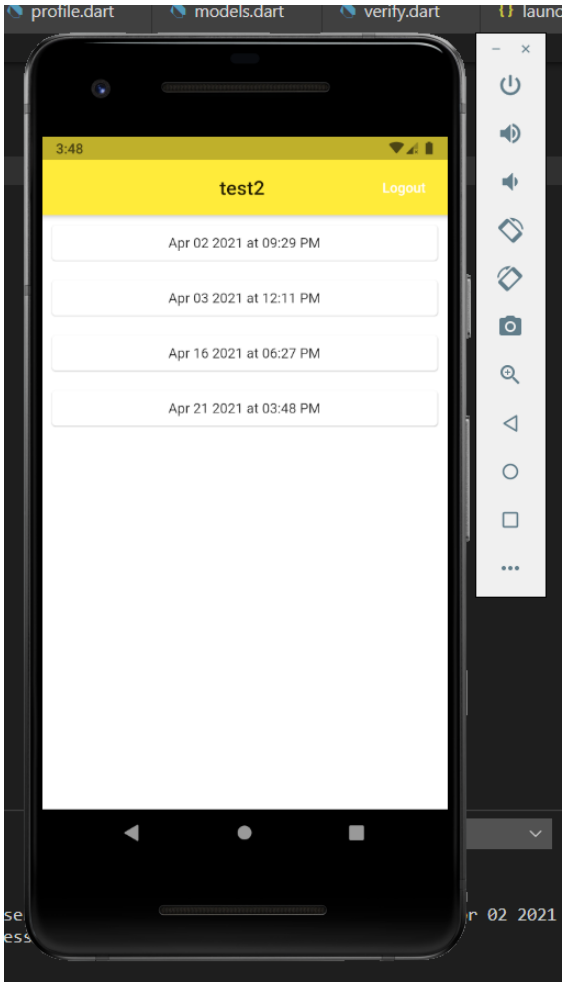


Figure 140 Driver Drowsiness Data Test (1)

Test No.	Action	Expected Outcome	Actual Outcome	Result
19.	Updating drowsiness data in backend and showing in screen.	New updated data should be added	New updated data was added to the older data.	Passed.

Table 21 Driver Drowsiness Data Test

4.2.19 Owner Login in mobile app

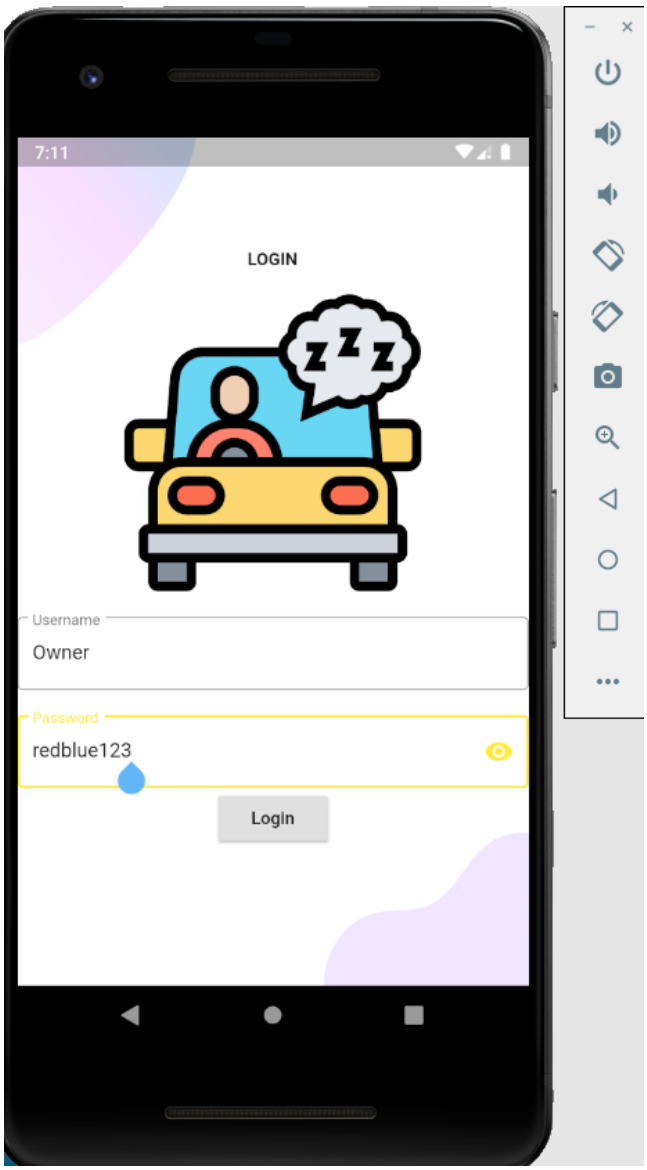


Figure 141 Owner Login in mobile app Test (1)

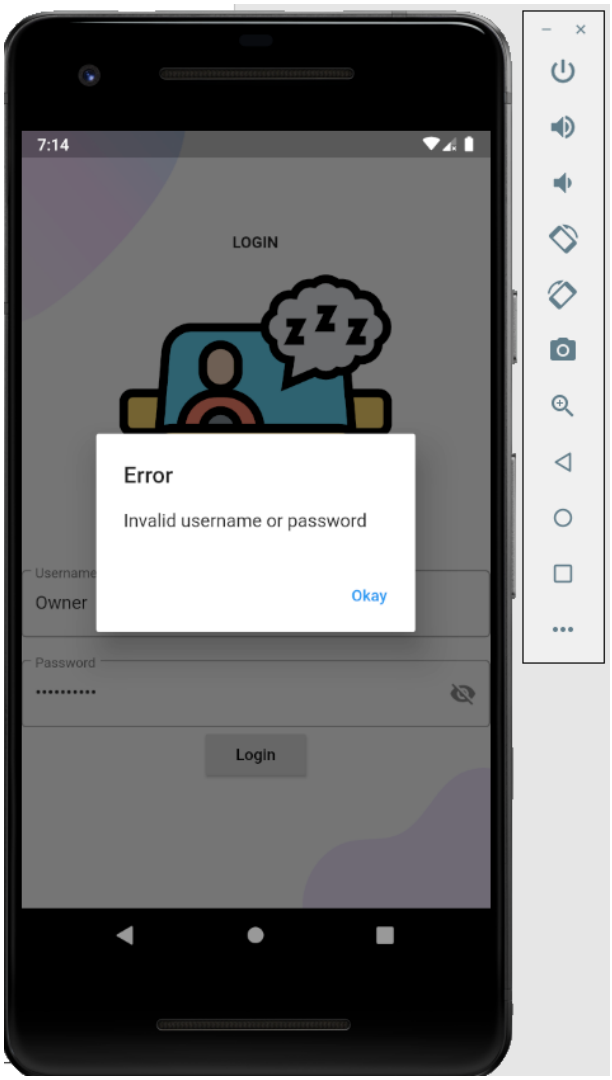


Figure 142 Owner Login in mobile app Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
20.	Login into mobile app using owner information.	Should not login as this app is only for drivers.	Was not able to login.	Passed.

Table 22 Owner Login in mobile app Test

4.2.20 Driver Login Validation (Mobile)

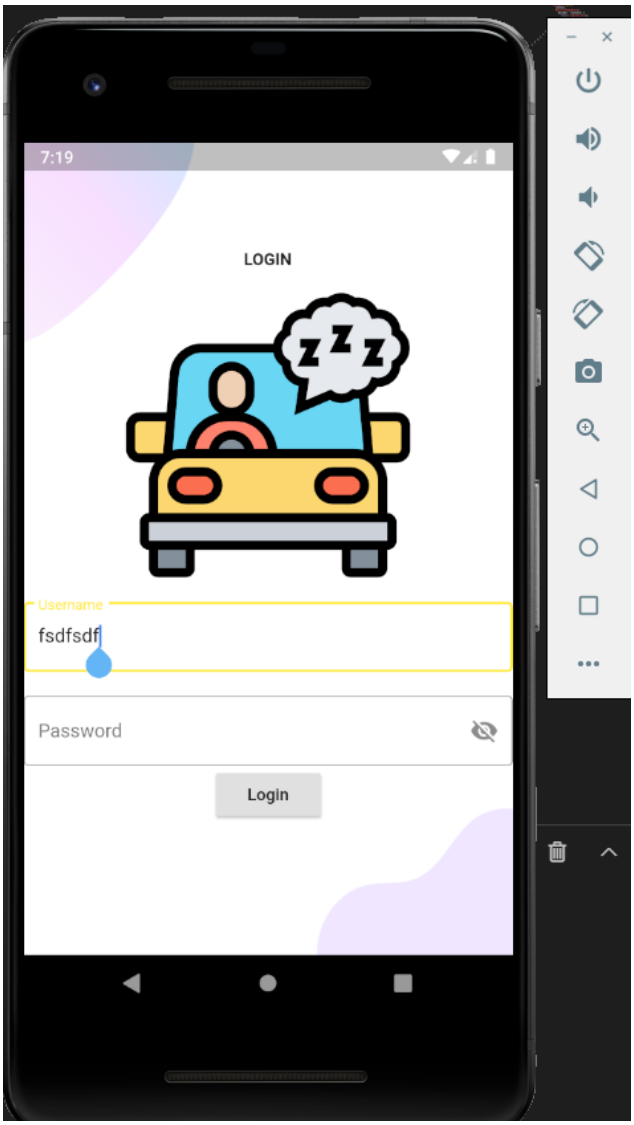


Figure 143 Driver Login Validation in mobile Test (1)

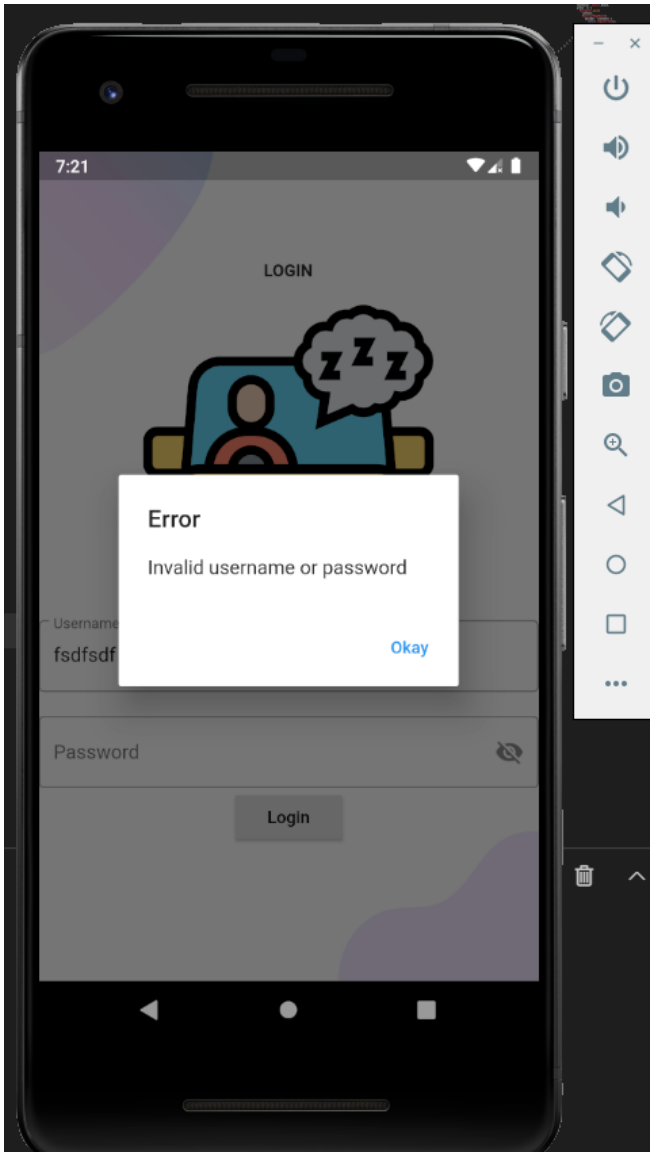


Figure 144 Driver Login Validation in mobile Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
21.	Login into mobile app using invalid data.	An error message should pop up.	An error message popped up and login was failed.	Passed.

Table 23 Driver Login Validation in mobile Test

4.2.21 Eye Detection Test (Drowsiness Detection)

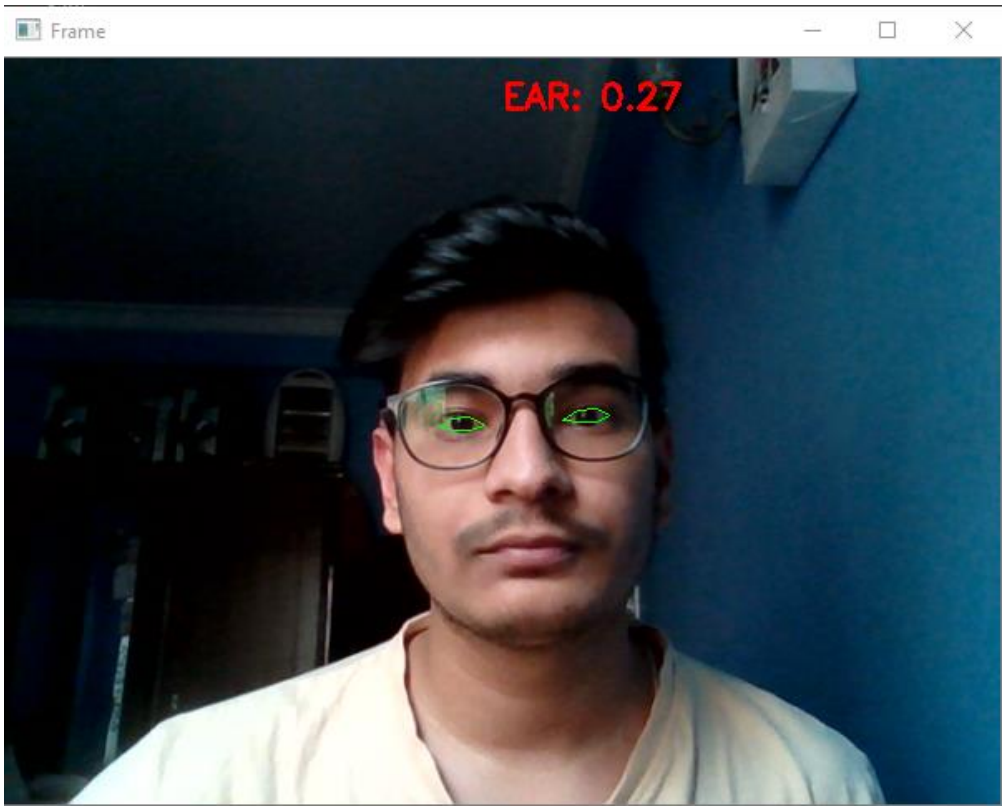


Figure 145 Running Drowsiness Detection Software and tracking eye Test (1)

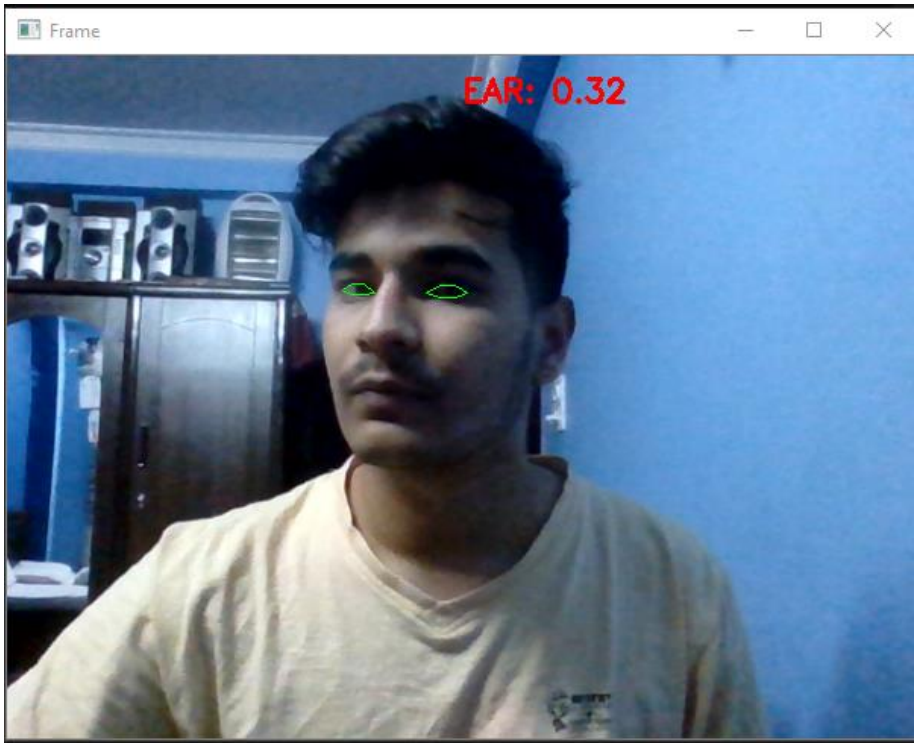


Figure 146 Running Drowsiness Detection Software and tracking eye Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
22.	The drowsiness software was opened for eye tracking.	The eye area should be tracked from any movement.	The eye was tracked from any movement.	Passed.

Table 24 Eye Detection Test

4.2.22 Drowsiness Alert Test (Drowsiness Detection)

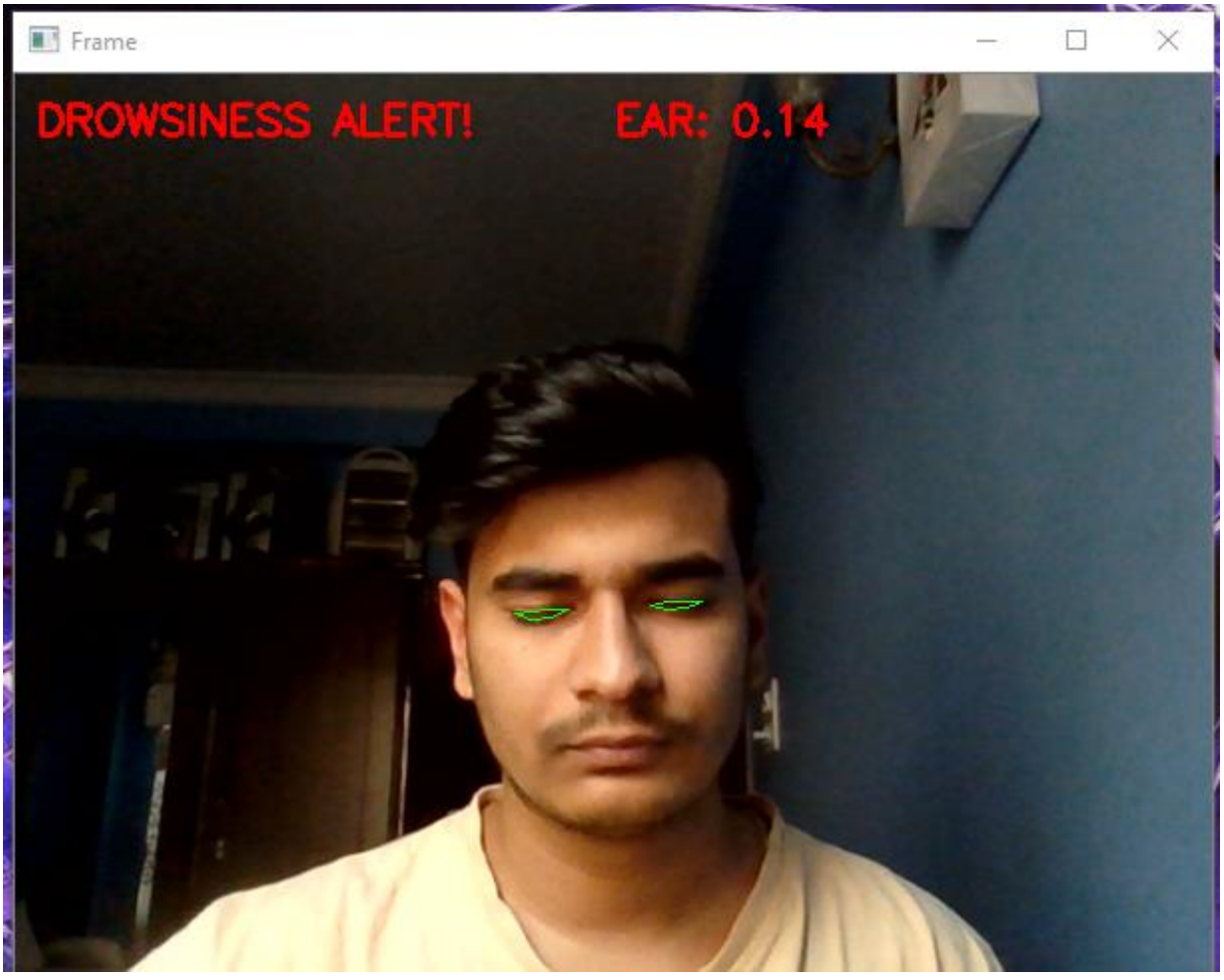


Figure 147 Drowsiness alert after feeling drowsiness Test

Test No.	Action	Expected Outcome	Actual Outcome	Result

23.	The user was feeling drowsy.	The software should alert the user by showing alert and buzzing alarm.	The eye was tracked from any movement.	Passed.

Table 25 Drowsiness Alert Test

4.2.23 Update Drowsiness Data (Web App)

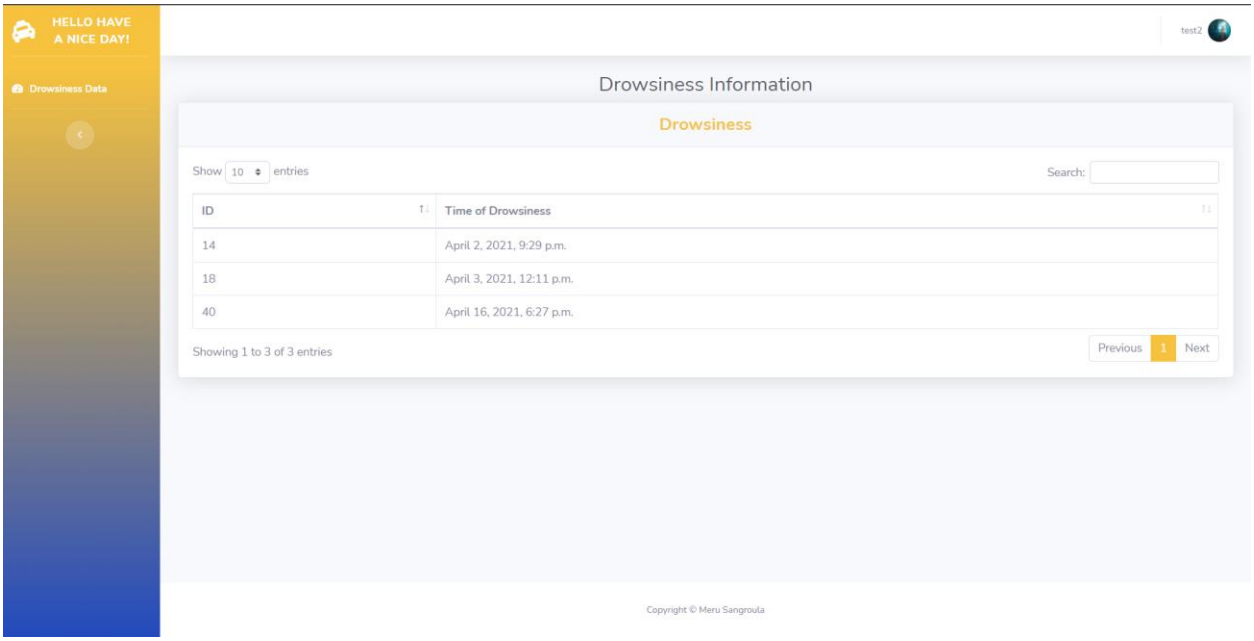


Figure 148 Updated Drowsiness Data Test (1)

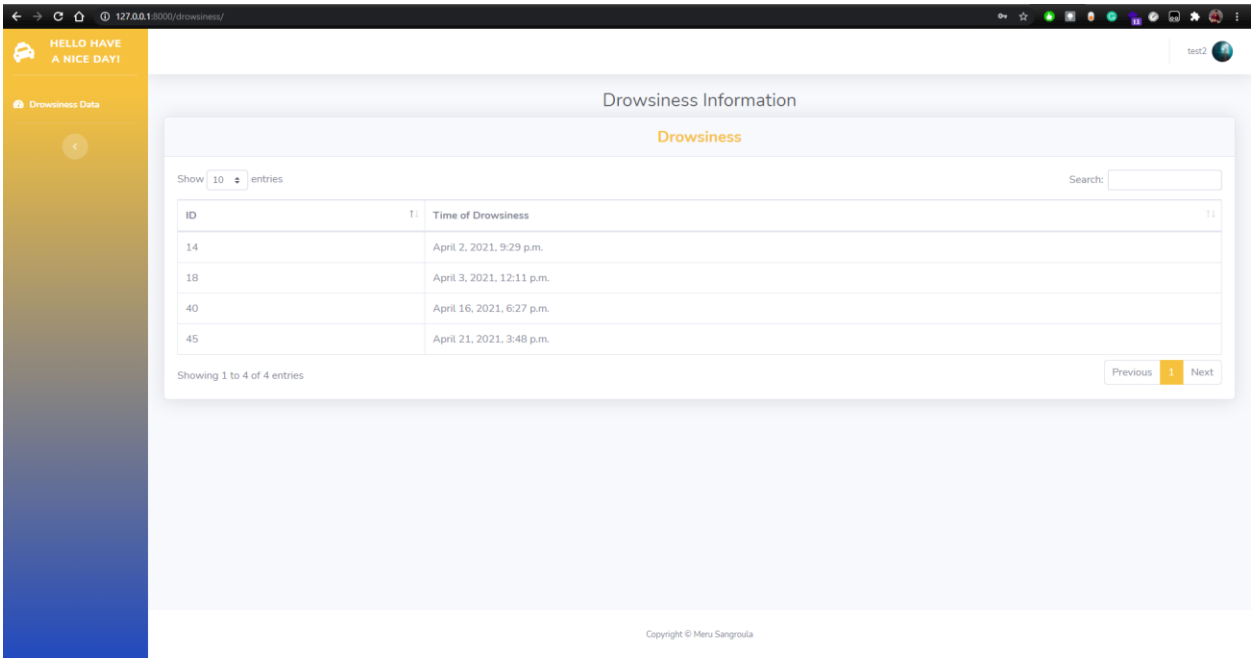


Figure 149 Updated Drowsiness Data Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
24.	The web application was opened after the drowsiness test.	The web application should show updated drowsiness data after the drowsiness test.	The web application showed updated drowsiness data after the drowsiness test.	Passed.

Table 26 Updated Drowsiness Data Test

4.3 SYSTEM TESTING

4.3.1 Sending Drowsiness API (Django)

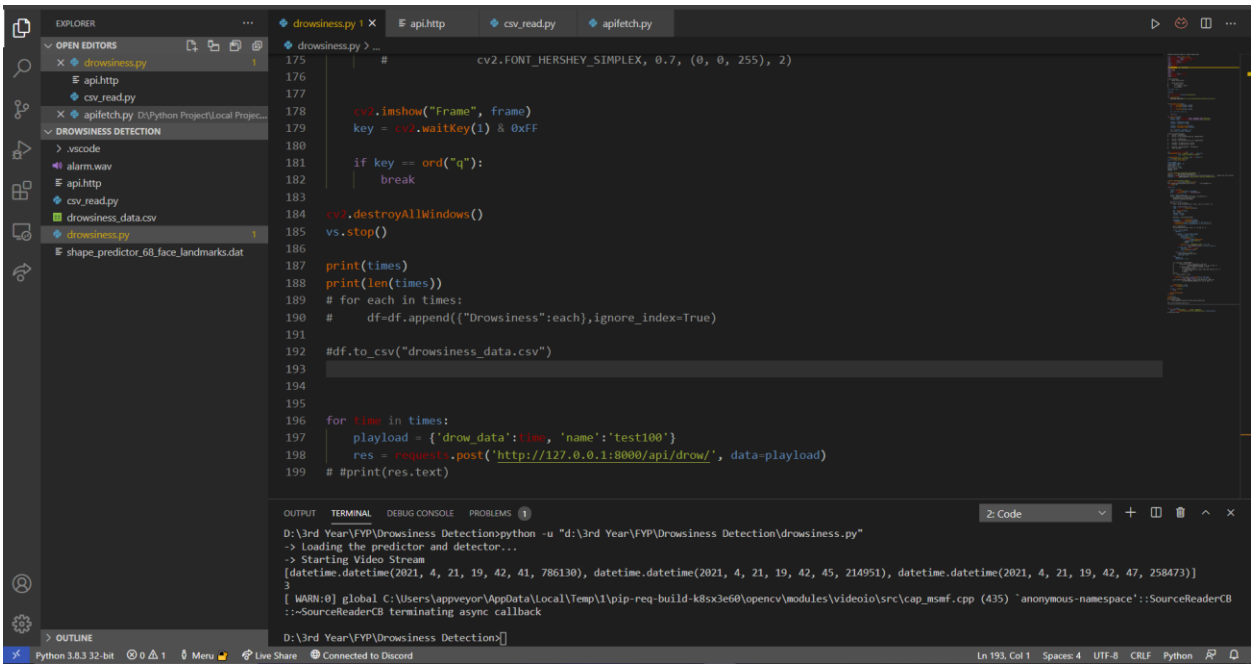


Figure 150 Sending Drowsiness API Test (1)

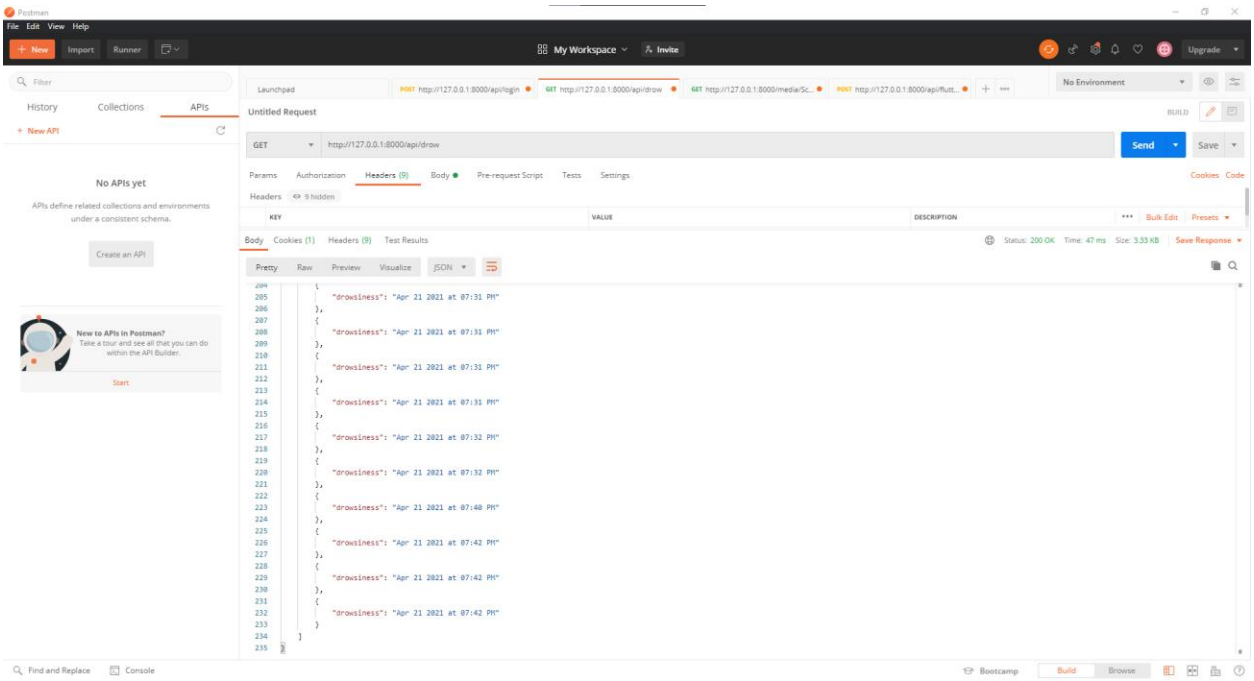


Figure 151 Sending Drowsiness API Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
25.	The post request was sent to the given URL after the software was run.	The API should get the date and time once after the drowsiness software is stopped.	The API got the data after the software stopped and the date and time were correct.	Passed.

Table 27 Sending Drowsiness API Test

4.3.2 Sending Drowsiness API and User Token for Mobile

```
class AuthToken(ObtainAuthToken):
    def post(self, request, *args, **kwargs):
        serializer = self.serializer_class(data=request.data,
                                           context={'request': request})
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        driver_data = Driver.objects.get(user=user.id)
        drowsiness_data = Drow.objects.filter(driver=driver_data)
        serializerr = DrowsinessSerializer(drowsiness_data, many=True)
        token, created = Token.objects.get_or_create(user=user)
        context = {'token': token.key,
                   'user_id': user.id,
                   'username': user.username,
                   'serilizer': serializerr.data,
                   }
        return Response(context)
```

Figure 152 Sending Drowsiness API and User Token Test (1)

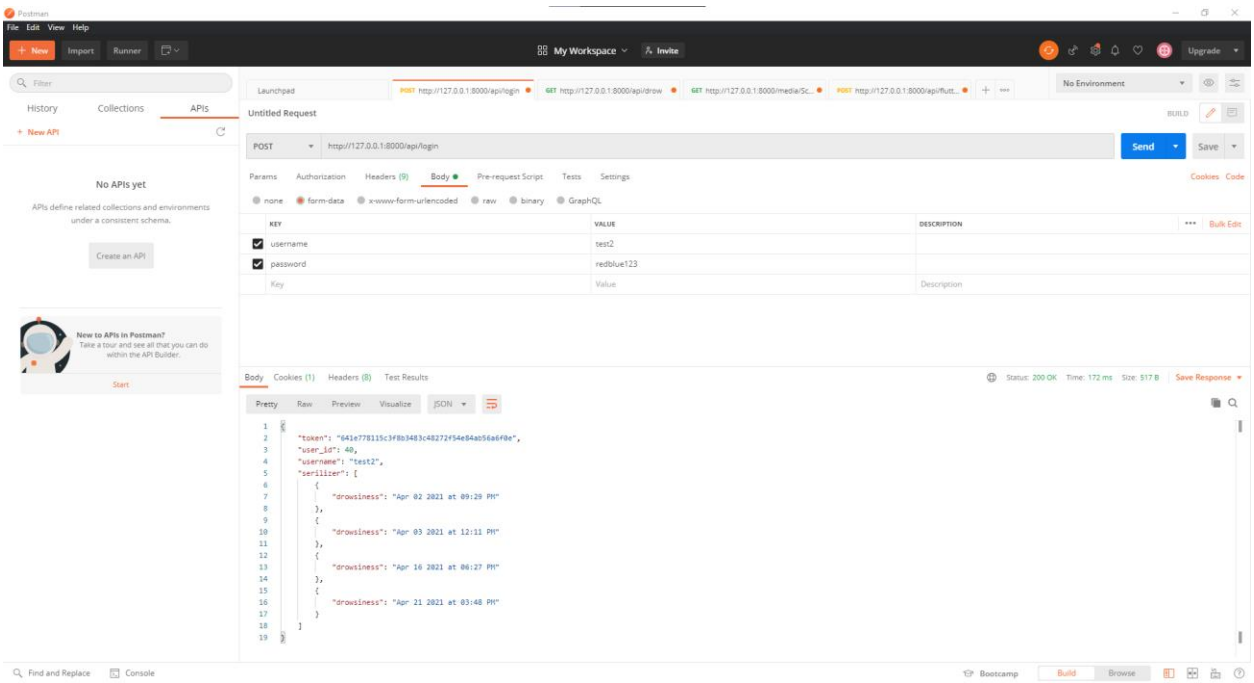


Figure 153 Sending Drowsiness API and User Token Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
26.	A post request containing token and driver drowsiness data was sent.	After hitting the API, driver token, drowsiness data and username should come.	Using the correct username and password driver token, username and drowsiness data was successfully received.	Passed.

Table 28 Sending Drowsiness API and User Token Test

4.3.3 Database Connection and Data Visualization

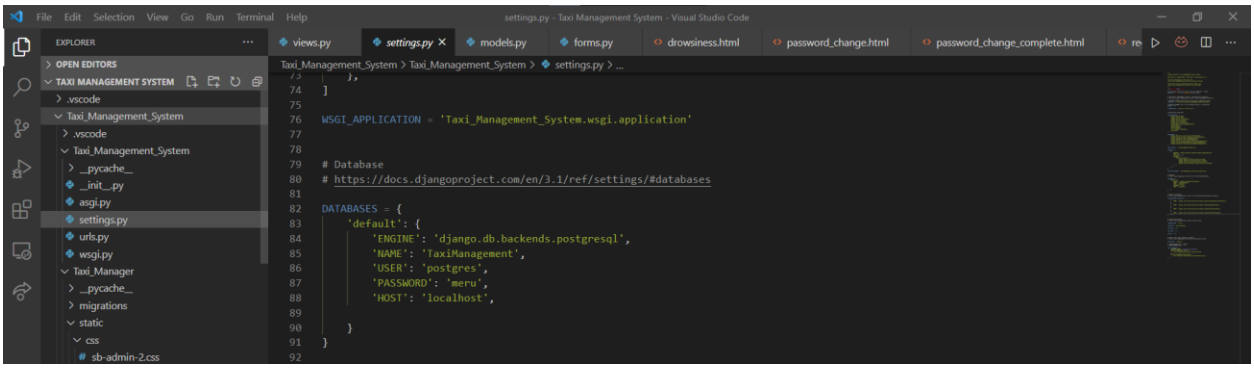


Figure 154 Database connection and data visualization Test (1)

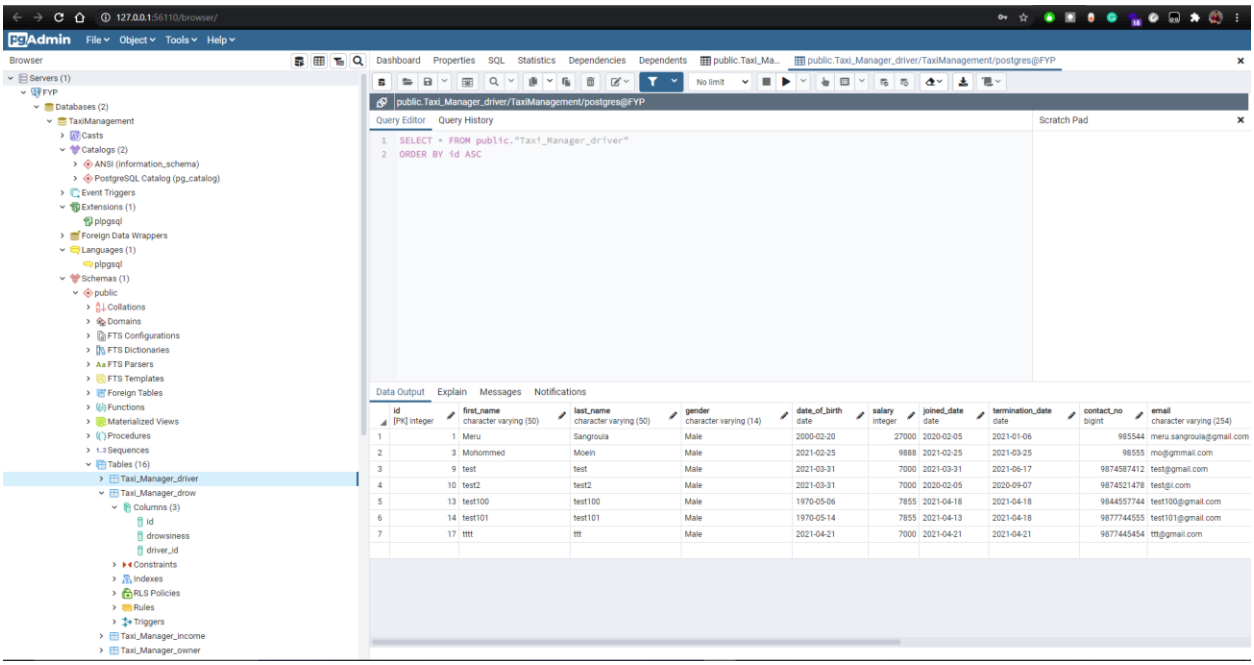


Figure 155 Database connection and data visualization Test (2)

Test No.	Action	Expected Outcome	Actual Outcome	Result
27.	Loading pg admin and showing all the tables and data used in the project.	All the data should be available in the database.	After database login, all the table and data were available and were secured.	Passed.

Table 29 Database connection and data visualization Test

4.3.4 URL Run Time

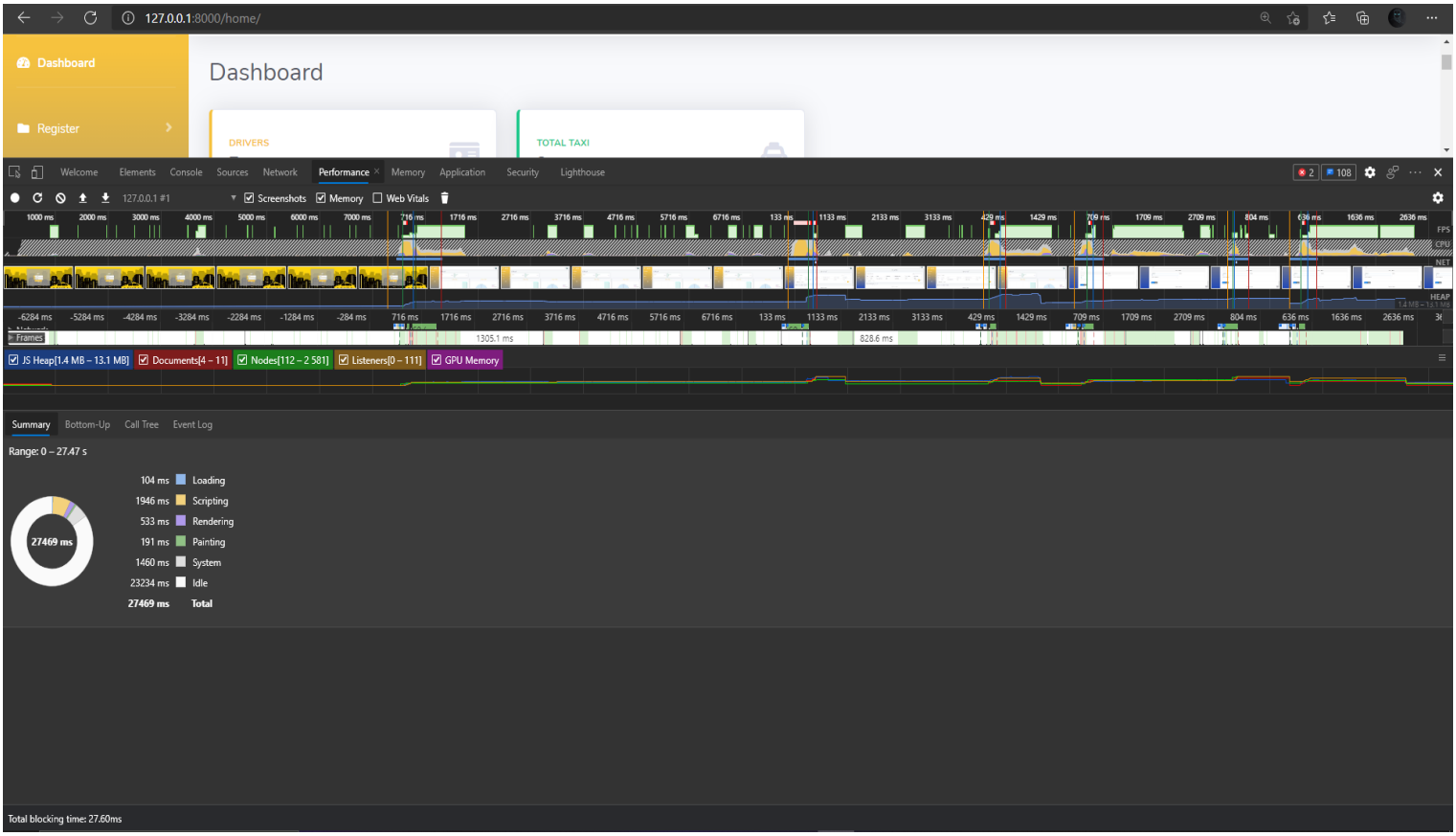


Figure 156 URL Run Time

Test No.	Action	Expected Outcome	Actual Outcome	Result
28.	Recording the web app using time up to 27.60 ms	Less response time was expected.	From 0-27.60 ms, the web app got performance of 27469 ms.	Passed.

Table 30 URL Run Time

4.4 CRITICAL ANALYSIS

After carrying out both system and unit testing, all the test were successfully concluded. There was no such big failure between testing, however some small errors appeared, and I also handled it properly.

4.4.1 Testing Analysis

During unit testing, all the features were working successfully and had no big issue. Testing was done in my own laptop and screenshot of all my project features were shown. In unit testing, total of 23 test were done, which included web app feature testing, mobile app feature testing and drowsiness software feature testing. During testing, the local host server sometimes used to stop responding and I use to reload the local host time to time. Eye detection testing for moving object was very difficult because it is very hard to show testing of moving object in the form of screenshot.

In system testing, 4 tests were done. Testing the URL run time was completely new thing for me. I recorded the browser run time and showed in the test. The results were decent. In system testing, the I have also shown the proper working of API and database.

4.4.2 Error and Error Handling

During making of the project and testing lots of errors occurred. Here are some of the screenshots of error occurred and different ways applied to solve the error:

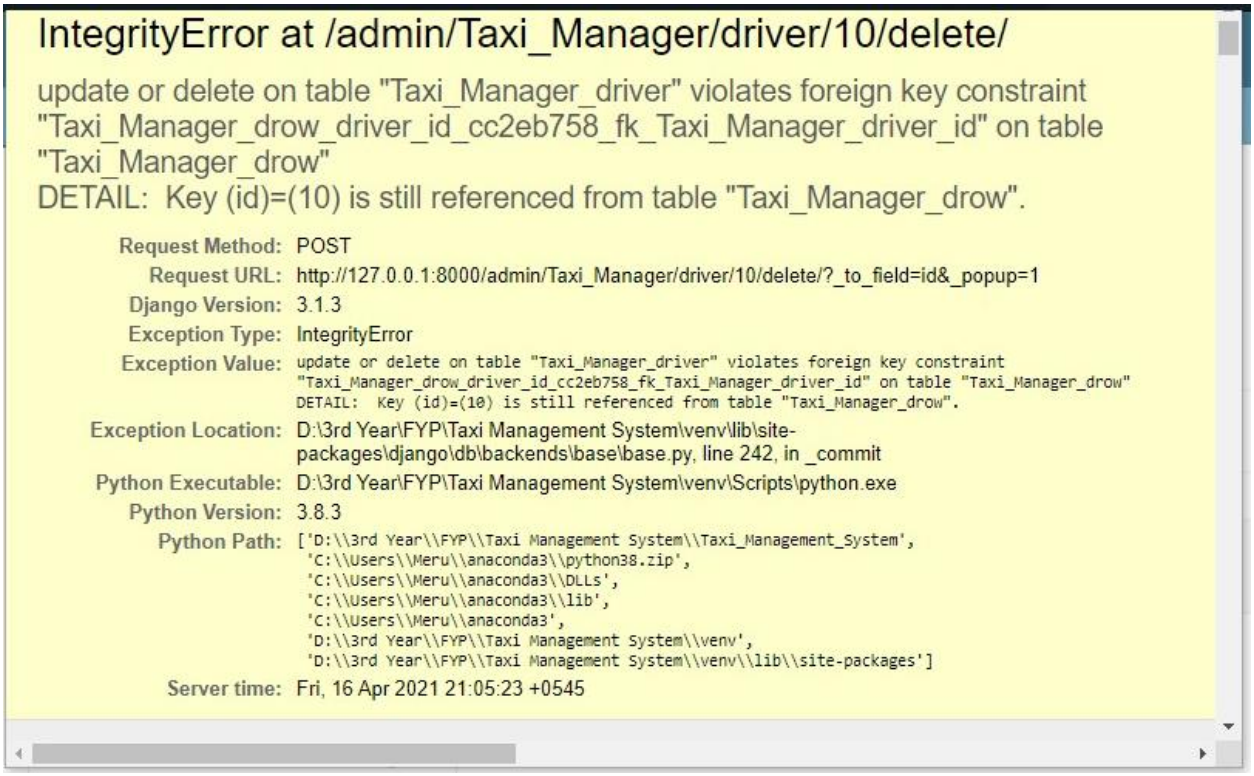


Figure 157 Integrity Error

This was a database error where there was some conflict between foreign key from driver table. I was unable to delete the driver while doing CRUD operations.

```
class Drow(models.Model): #All the drowsiness data of driver will be stored here
    #driver = models.ForeignKey(Driver, on_delete=models.CASCADE)
    drowsiness = models.DateTimeField(null=True)
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE, null=True)
```

Figure 158 Integrity Error Solution

As a solution, I used on_delete function in the models and used CASCADE. It basically deletes all the foreign key related data. In this way I fixed this problem.

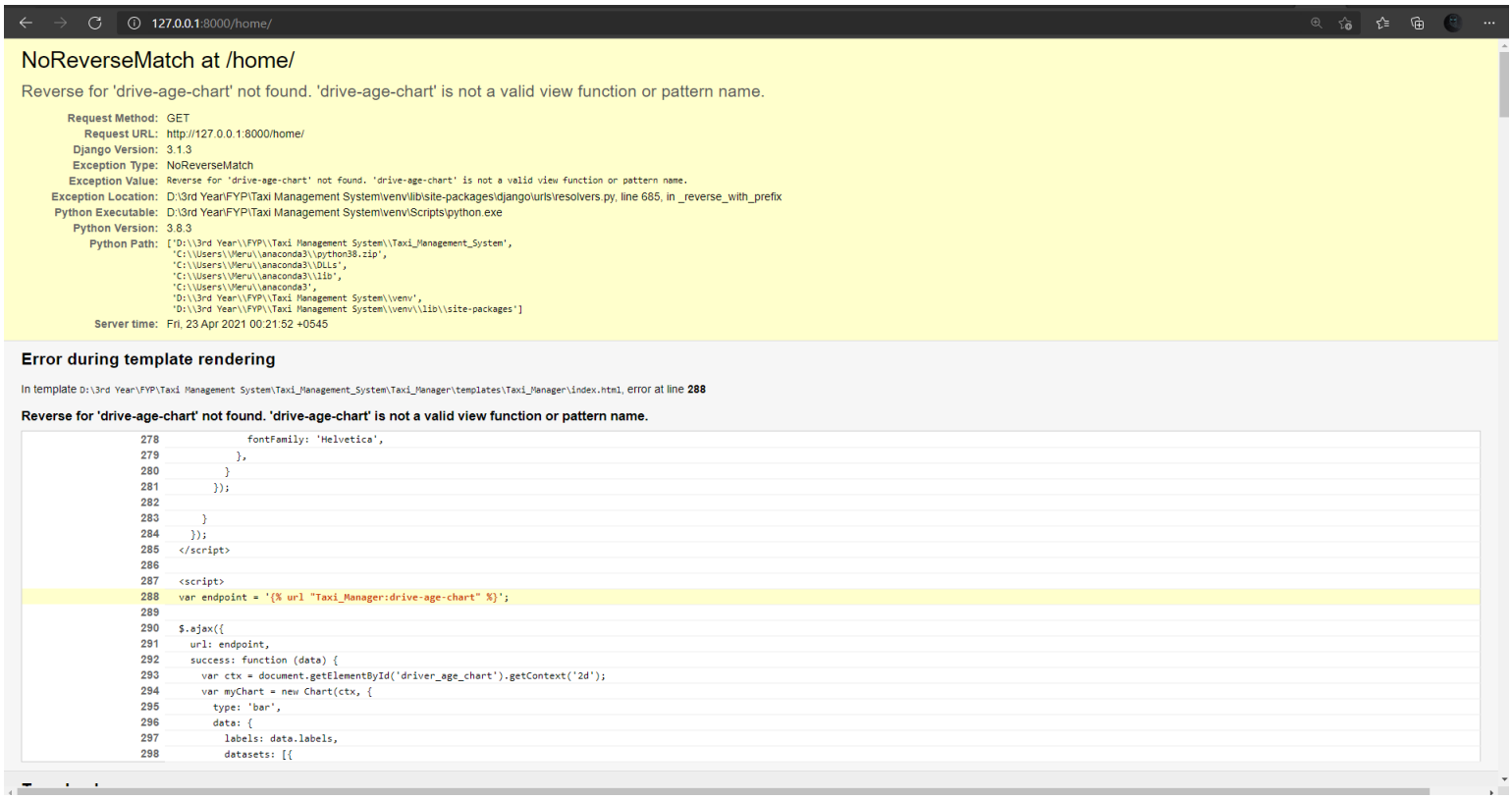


Figure 159 Invalid View

This error was very easily solved. Here, inside the index.html file, I accidently called the wrong function which was not even made. Here, Django itself gave me the line number so I understood very easily that I called the wrong function.

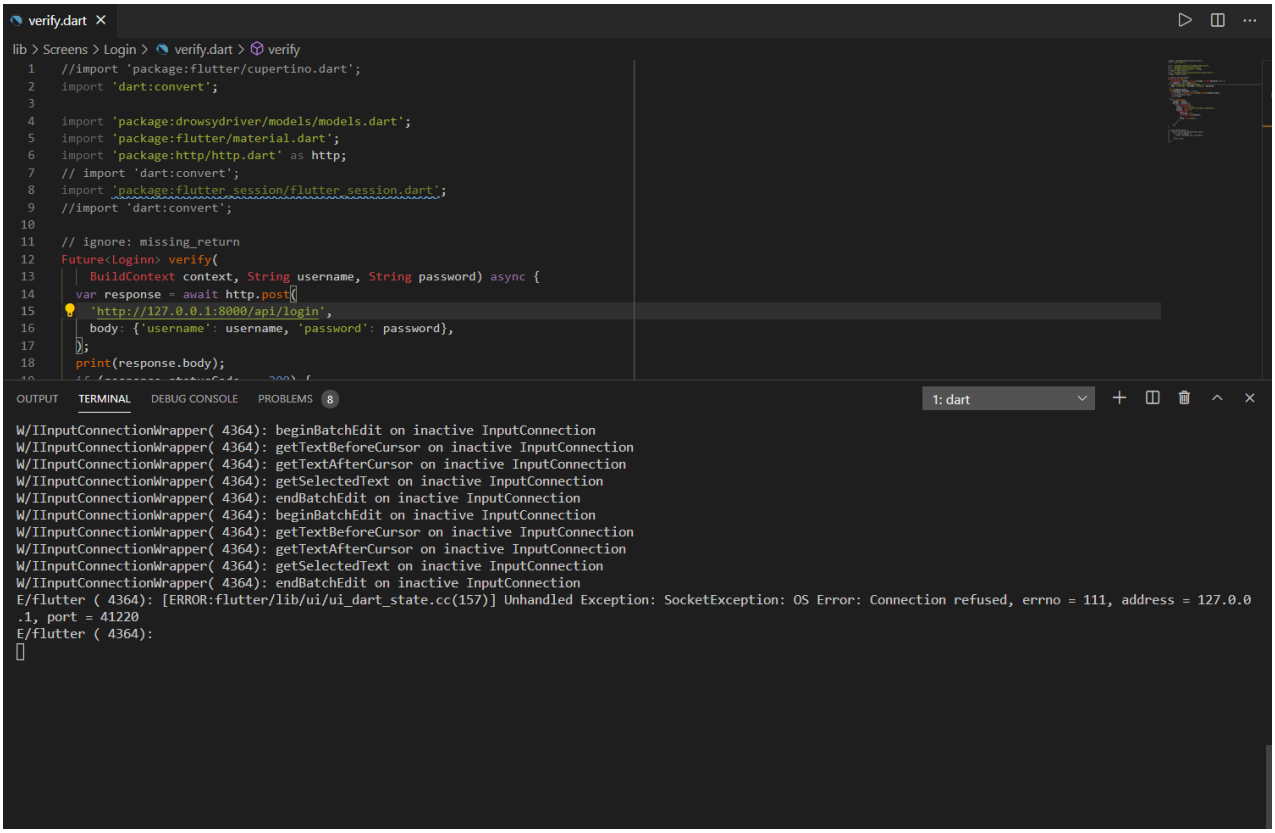


Figure 160 Flutter API Error

Among all the error, this error took some time because I competed flutter app in very less time. This was an API error. The API wasn't getting any request but from the Django rest framework it was working. The problem was in line 15. Flutter does not support the Django API URL instead it has its own local host.

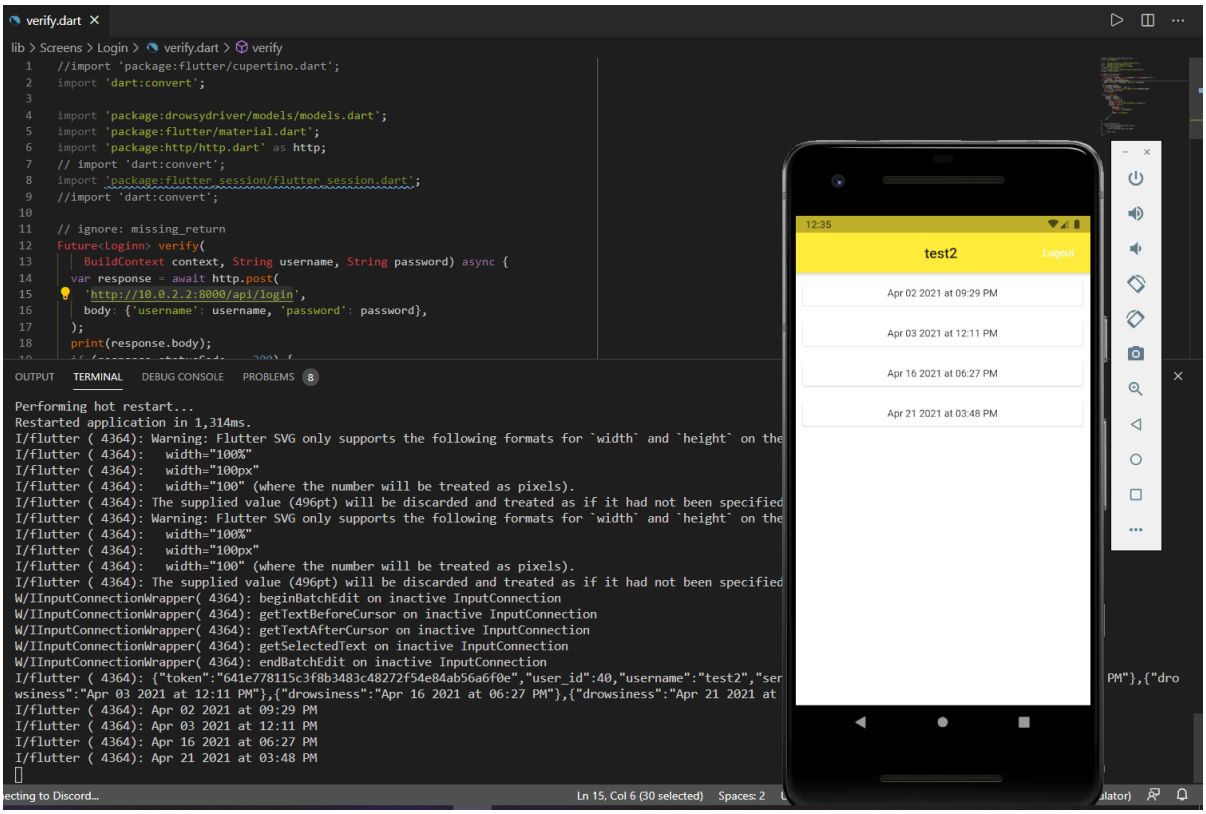


Figure 161 Flutter API Error Solved

Here, in verify.dart I simply needed to use the flutter local host number instead of Django URL and it worked.

All the above errors are not all the errors that I got during testing or during development but other errors were of similar kind. Solving errors is also one of the main sill that we need to have as a developer and while doing the project I had chance to experience that.

CHAPTER 5: CONCLUSION

5.1 LEGAL, SOCIAL AND ETHICAL ISSUES

5.1.1 LEGAL ISSUES

Legal Issues basically refers to law related issues. Legal issues requires court decision and can also refer to a point in which the evidence is undisputed, the outcome of which depends on the court's interpretation of the law (uslegal, 2021).

According to the law of Nepal, every people in Nepal have right to start their own business. My project only focuses on comfort of business owners by using my project and the safety of drivers. The project does not violate any kind of legal laws. The purpose of this project is very straight forward and will not get involved in any kind of legal issues. During making of the project, all the tools and technology used in the project are open source. There is not any copying of code or any kind of pirated software used in the project.

Some legal steps which needs to be taken when the project is deployed in real life scenario is domain of website needs to be registered for web application and same goes for mobile app to be live on play store.

5.1.2 SOCIAL ISSUES

Social Issues refers to issues that directly or indirectly affects society. Social issue generates unnecessary hatred between people or society.

This project does not violets social issues from any directions. There is not any discrimination between gender, sex, or races in my project. Everyone can use the project. The project even has 3 different types of gender as it respects all types of gender and no one can misuse the project. My project does not support any kind of racism, cyber bullying, or violent conversions because it is for the corporate organization where records are kept rather than conversation.

If a driver of the company does not want drowsiness detection software in his/her taxi then the decision should be respected rather than forcing to have the software in taxi by the owner which is a complete violation of social issues.

5.1.3 ETHICAL ISSUES

Ethical issues occur when a given decision, scenario or activity creates a conflict with a society's moral principle. Both individuals and business can be involved in these conflicts (myaccountingcourse, 2021).

The project itself does not violates any kind of ethical issues but it basically depends upon the driver who my or may not cheat the owner.

For example, in my system, the driver has to fill up the daily income form which contains earning of the day and total km of a day. The driver can cheat the owner easily by lying about total earnings and keep the money himself. This action cannot be considered illegal, but it breaches company's Code of Conduct. Here, the driver can either lie about the daily income and make some extra cash or could be ethical and do not cheat to the company. Therefore, in this way ethical issue can be violated.

5.2 ADVANTAGES

The project is specifically for taxi business company rather than general people. This project basically focuses on managing taxi business more smother then it uses to be, and it is also about keeping driver safe from accidents caused by drowsiness. The advantages of this projects are as follows:

- This project comes with unique feature called "**Drowsiness Detection Software"**. According to survey, very few have known about implementation of drowsiness detection software in taxi. Well in Nepal, I have never known such software out in the market and the main advantage of this software is that it can help driver to prevent from read accidents caused due to drowsiness.
- The other advantage of my project is mobile application for each driver. The mobile application is just made to make the driver more self-aware. For example, if a driver got sleepy then the alarm automatically turns on to wake the driver and prevent driver from possible road accident. Now if the driver wants to know at what time he/she got drowsy then he/she can login into the mobile app or web app to view all the records of drowsiness.

- This project is for the owner of the taxi business as well. This project provides Dashboard for the owners. Some advantages of using Taxi Management System are given below:
    1. Makes record keeping more simpler and easier.
    2. User can add, update, delete both taxi and driver. Suppose if new driver replaces the old driver of particular taxi can keeping the record suing this system is less time consuming.
    3. User can see the overall income generated by the company.

5.3 LIMITATIONS

My three different of features have their own limitation which is mentioned below:

5.3.1 Drowsiness Detection Software

- The distraction software is most effective in bright environment and might fail to track eyes under dim light (evening scenarios).
- Since the distraction software is a prototype software, so the code must run in the computer which will be kept next to driver seat. This software does not fulfill the real-world implications.

5.3.2 Web Application

- There will be only one owner who will be using the dashboard. If multiple owner are made by the admin then both will have authority to access all the features and same database will be modified.
- Owner cannot change the password himself/herself.
- Driver can only view the drowsiness data and nothing more.
- Driver cannot view his/her own password but it the driver knows the password then can change the password.

5.3.3 Mobile Application

- Mobile application is only for driver.
- Driver does not have permission to write but only read the data.

5.4 FUTURE WORK

My project includes all the features which I mentioned earlier in proposal and except that I have also made additional mobile application. Due to limited time, I could not complete my mobile application. For now, I have only done login and data read part. In future, I will make the app more responsive and will also include following features:

- View Profile
- Edit Profile
- Payment Option

My project lacks payment option, where owner can directly send monthly salary via E-SEWA or Khalti to the drivers. In future, I plan to add payment system in my web application and plan to add GPS tracking. For now, in my web application, driver need to manually update the daily income earned by them. After the GPS tracking system, owner will have full control over the km and there will be less chance of fraud by the driver.

My main feature, Drowsiness Detection Software is just a prototype software and this software has lots of limitations and needs a lot of future work. As a computer with the script running in the background is required for my software to run correctly. I have already mentioned this is a prototype model and cannot be relied upon real world case. For future, with external hardware like Raspberry Pi or some good compute capabilities minicomputers and a GPU, the software will be ready to run on real word. Some research on future work are on appendix section **(click here)**.

CHAPTER 6: REFERENCES

Brownlee, J. (2019) *machinelearningmastery* [Online]. Available from: https://machinelearningmastery.com/what-is-computer-vision/ [Accessed 12 November 2020].

educba. (2020) *iterative methodology* [Online]. Available from: https://www.educba.com/iterative-methodology/ [Accessed 16 December 2020].

flutter. (2021) *flutter* [Online]. Available from: https://flutter.dev/ [Accessed 2 April 2021].

getsetproject. (2020) *Taxi Management System* [Online]. Available from: https://getsetproject.com/info-project.php?id=276&name=Taxi+Management+System [Accessed 20 December 2020].

Guru 99. (2020) *Prototyping Model in Software Engineering: Methodology, Process, Approach* [Online]. Available from: https://www.guru99.com/software-engineering-prototyping-model.html [Accessed 16 December 2020].

guru99. (2021) *Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE* [Online]. Available from: https://www.guru99.com/unit-testing-guide.html [Accessed 19 April 2021].

guru99. (2021) *What is System Testing? Types & Definition with Example* [Online]. Available from: https://www.guru99.com/system-testing.html#:~:text=SYSTEM%20TESTING%20is%20a%20level,a%20larger%20computer%2Dbased%20system. [Accessed 19 April 2021].

International Bank for Reconstruction and Development/The World Bank. (2020) *Delivering Road Safety*. Washington DC: International Bank for Reconstruction and Development/The World Bank.

Jugnoo Taxi. (2020) *Jugnoo Taxi* [Online]. Available from: https://jugnoo.io/jugnootaxi/ [Accessed 20 December 2020].

Jugnoo. (2020) *About Us* [Online]. Available from: https://jugnoo.io/about-us/ [Accessed 16 December 2020].

Jugnoo. (2020) *Linkedin* [Online]. Available from: https://www.linkedin.com/company/jugnoo/?originalSubdomain=in [Accessed 16 December 2020].

K, J. (2020) *acodez* [Online]. Available from: https://acodez.in/12-best-software-development-methodologies-pros-cons/#Prototype_Methodology [Accessed 16 December 2020].

Kruchten, P. (2004) *The Rational Unified Process: An Introduction*. Boston : Pearson Education.

McCormick, M. (2012) *Waterfall vs. Agile Methodology*. Michael McCormick.

Mrsic, M. (2017) *Rational Unified Process (RUP)* [Online]. Available from: https://activecollab.com/blog/project-management/rational-unified-process-rup [Accessed 15 October 2020].

myaccountingcourse. (2021) *What are Ethical Issues?* [Online]. Available from: https://www.myaccountingcourse.com/accounting-dictionary/ethical-issues#:~:text=What%20Does%20Ethical%20Issues%20Mean,with%20a%20society's%20moral%20principles.&text=Ethical%20issues%20are%20challenging%20because,guidelines%20or%20precedents%20are%20known [Accessed 19 April 2021].

postgresqltutorial. (2020) *What is PostgreSQL?* [Online]. Available from: https://www.postgresqltutorial.com/what-is-postgresql/ [Accessed 15 October 2020].

Pyimagesearch. (2017) *Drowsiness detection with OpenCV* [Online]. Available from: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/ [Accessed 20 December 2020].

S.Balaji, D.M.S.M. (2012) WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. *WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC*, 2(1), p.30.

SUBEDI, K.N. (2020) *The Himalayan Times* [Online]. Available from: https://thehimalayantimes.com/business/increasing-the-number-of-taxis-brings-forth-different-viewpoint/ [Accessed 12 October 2020].

testbytes. (2019) *What is Rational Unified Process Methodology?* [Online]. Available from: https://www.testbytes.net/blog/rational-unified-process/ [Accessed 15 October 2020].

The Economic Times. (2020) *Definition of 'Waterfall Model'* [Online]. Available from: https://economictimes.indiatimes.com/definition/waterfall-model [Accessed 14 October 2020].

uslegal. (2021) *Legal Issue Law and Legal Definition* [Online]. Available from: https://definitions.uslegal.com/l/legal-issue/#:~:text=Legal%20issue%20or%20issue%20of,court's%20interpretation%20of%20the%20law. [Accessed 15 April 2021].

visual-paradigm. (2021) *What is Data Flow Diagram?* [Online]. Available from: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/ [Accessed 15 April 2021].

World Health Ranking. (2020) *NEPAL: ROAD TRAFFIC ACCIDENTS* [Online]. Available from: https://www.worldlifeexpectancy.com/nepal-road-traffic-accidents#:~:text=According%20to%20the%20latest%20WHO,Nepal%20%2376%20in%20the%20world. [Accessed 30 September 2020].

Yelowsoft. (2020) *Yelowsoft* [Online]. Available from: https://www.yelowsoft.com/ [Accessed 20 December 2020].

CHAPTER 7: BIBLIOGRAPHY

djangoproject. (2021) *User authentication in Django* [Online]. Available from: https://docs.djangoproject.com/en/3.2/topics/auth/ [Accessed 15 January 2021].

djangoproject. (2021) *View Decorators* [Online]. Available from: https://docs.djangoproject.com/en/3.2/topics/http/decorators/ [Accessed 12 March 2021].

django-rest-framework. (2021) *Authentication* [Online]. Available from: https://www.django-rest-framework.org/api-guide/authentication/ [Accessed 06 February 2021].

django-rest-framework. (2021) *settings* [Online]. Available from: https://www.django-rest-framework.org/api-guide/settings/ [Accessed 04 February 2021].

flutter. (2021) *flutter* [Online]. Available from: https://flutter.dev/ [Accessed 2 April 2021].

flutter. (2021) *Widget catalog* [Online]. Available from: https://flutter.dev/docs/development/ui/widgets [Accessed 14 March 2021].

postgresqltutorial. (2020) *What is PostgreSQL?* [Online]. Available from: https://www.postgresqltutorial.com/what-is-postgresql/ [Accessed 15 October 2020].

Pyimagesearch. (2017) *Drowsiness detection with OpenCV* [Online]. Available from: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/ [Accessed 20 December 2020].

visual-paradigm. (2021) *What is Data Flow Diagram?* [Online]. Available from: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/ [Accessed 15 April 2021].

World Health Ranking. (2020) *NEPAL: ROAD TRAFFIC ACCIDENTS* [Online]. Available from: https://www.worldlifeexpectancy.com/nepal-road-traffic-accidents#:~:text=According%20to%20the%20latest%20WHO,Nepal%20%2376%20in%20the%20world. [Accessed 30 September 2020].

CHAPTER 8: APPENDIX

8.1 APPENDIX A: PRE-SURVEY

8.1.1 PRE-SURVERY FORM

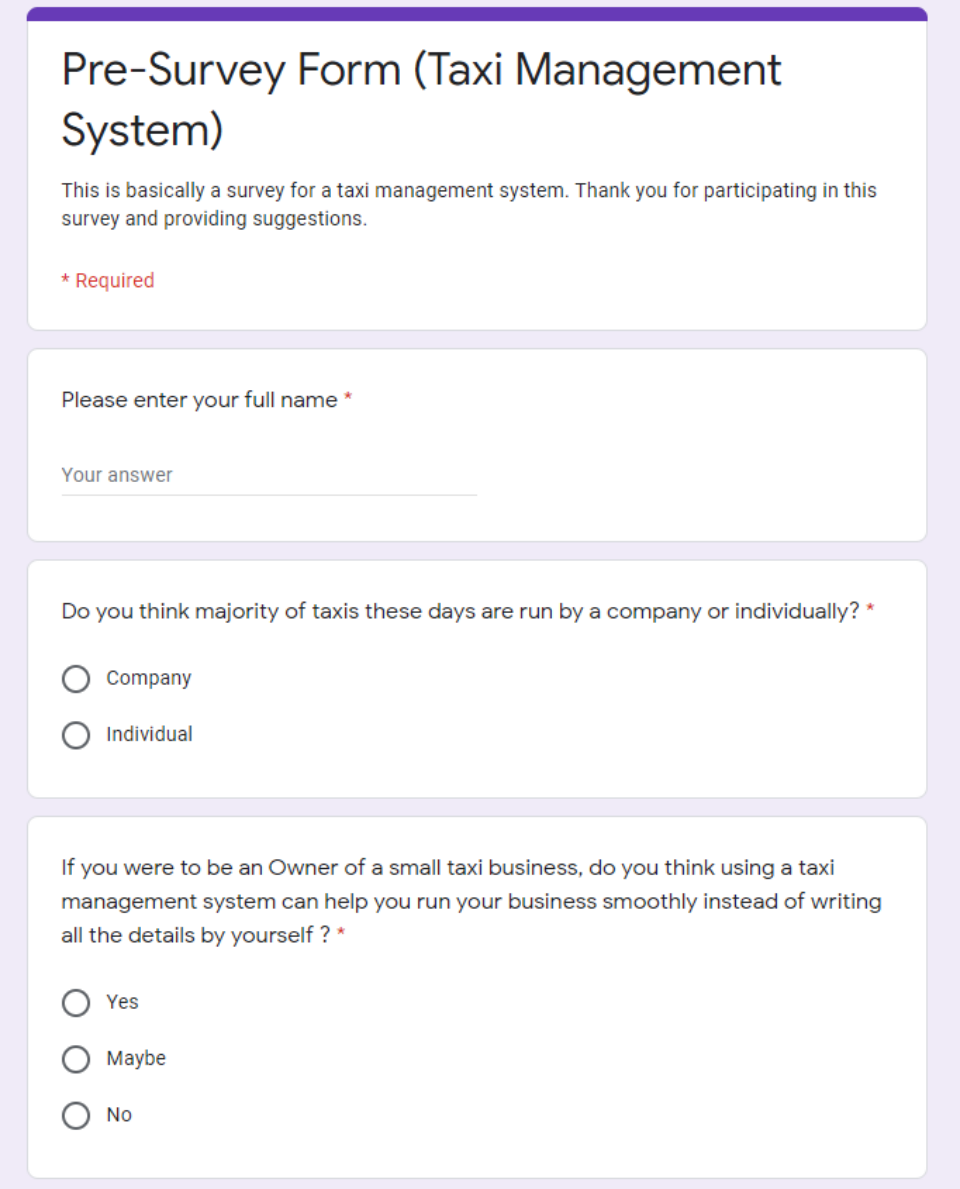


Figure 162 Pre-Survey (1)

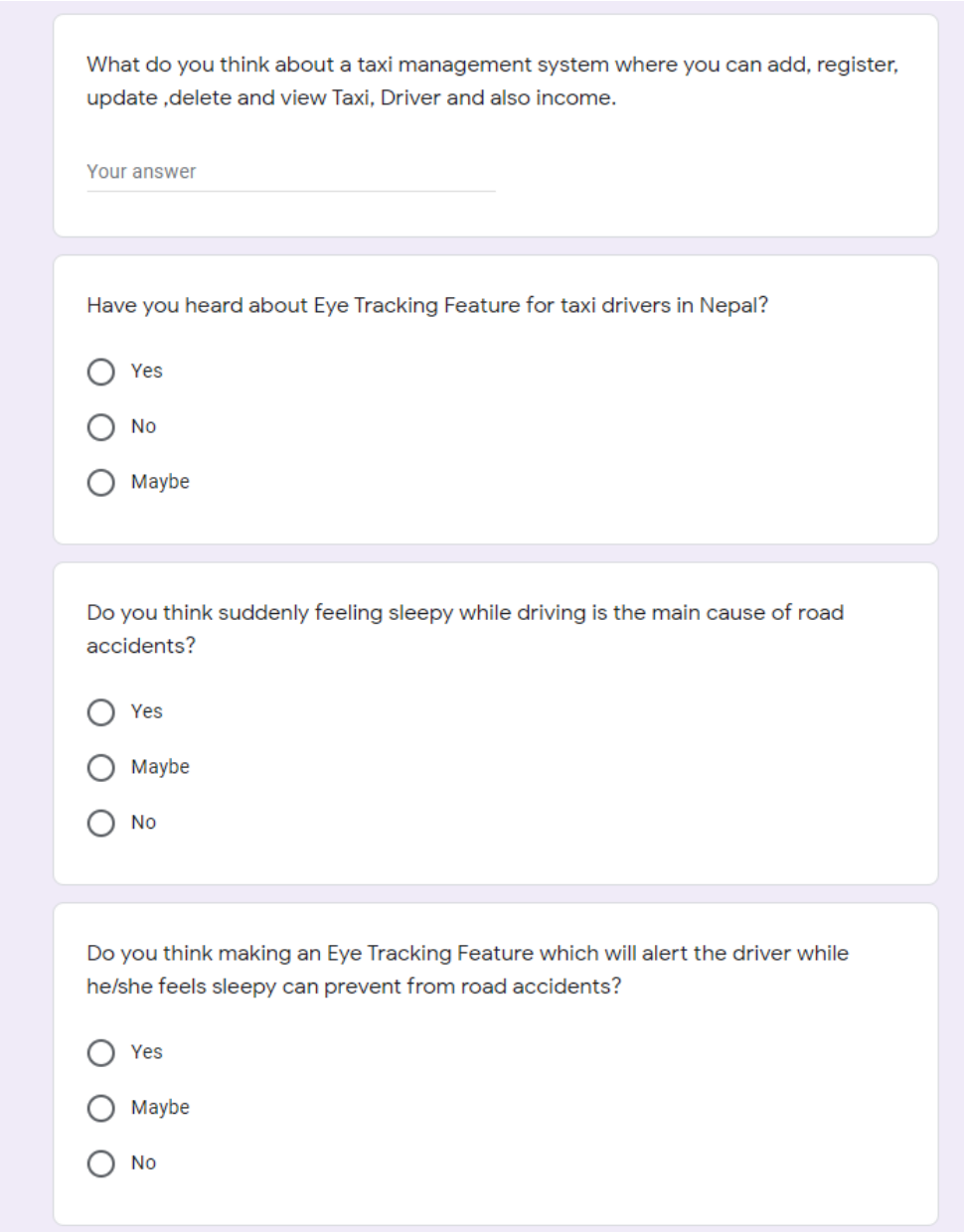


Figure 163 Pre-Survey (2)

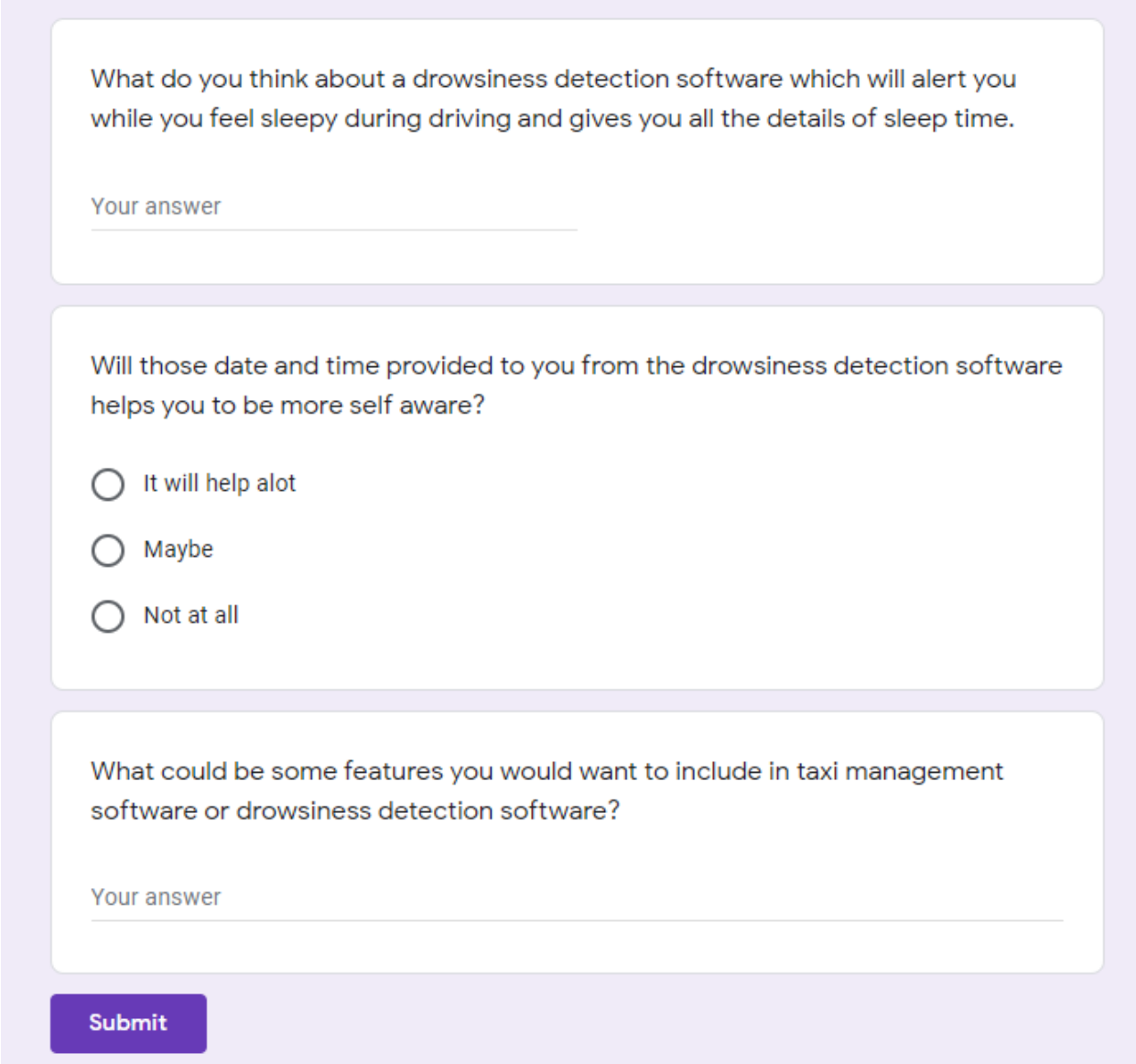


Figure 164 Pre-Survey (3)

8.1.2 SAMPLE OF FILLED PRE-SURVEY FORMS

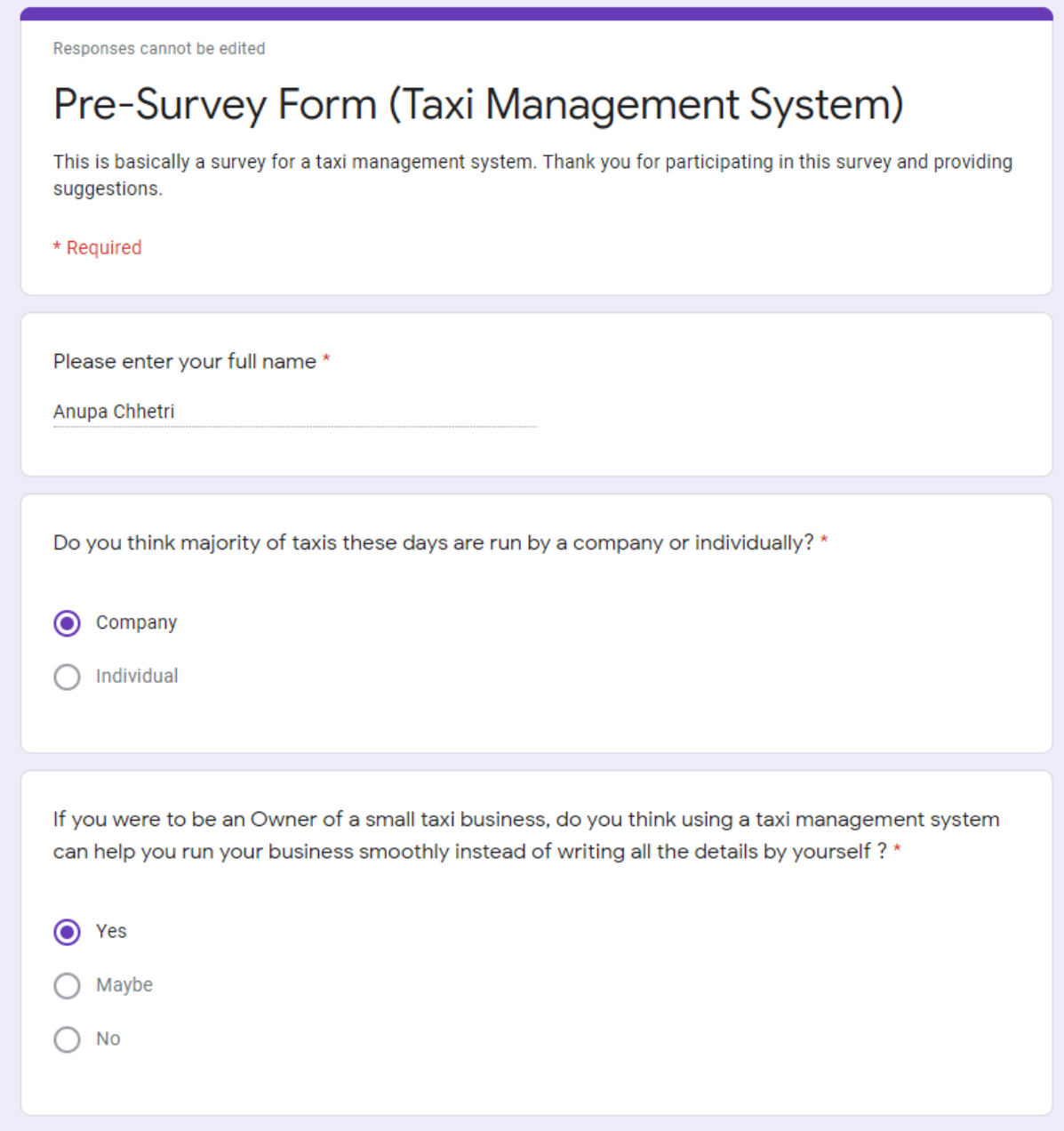


Figure 165 Pre-Survey Sample (1)

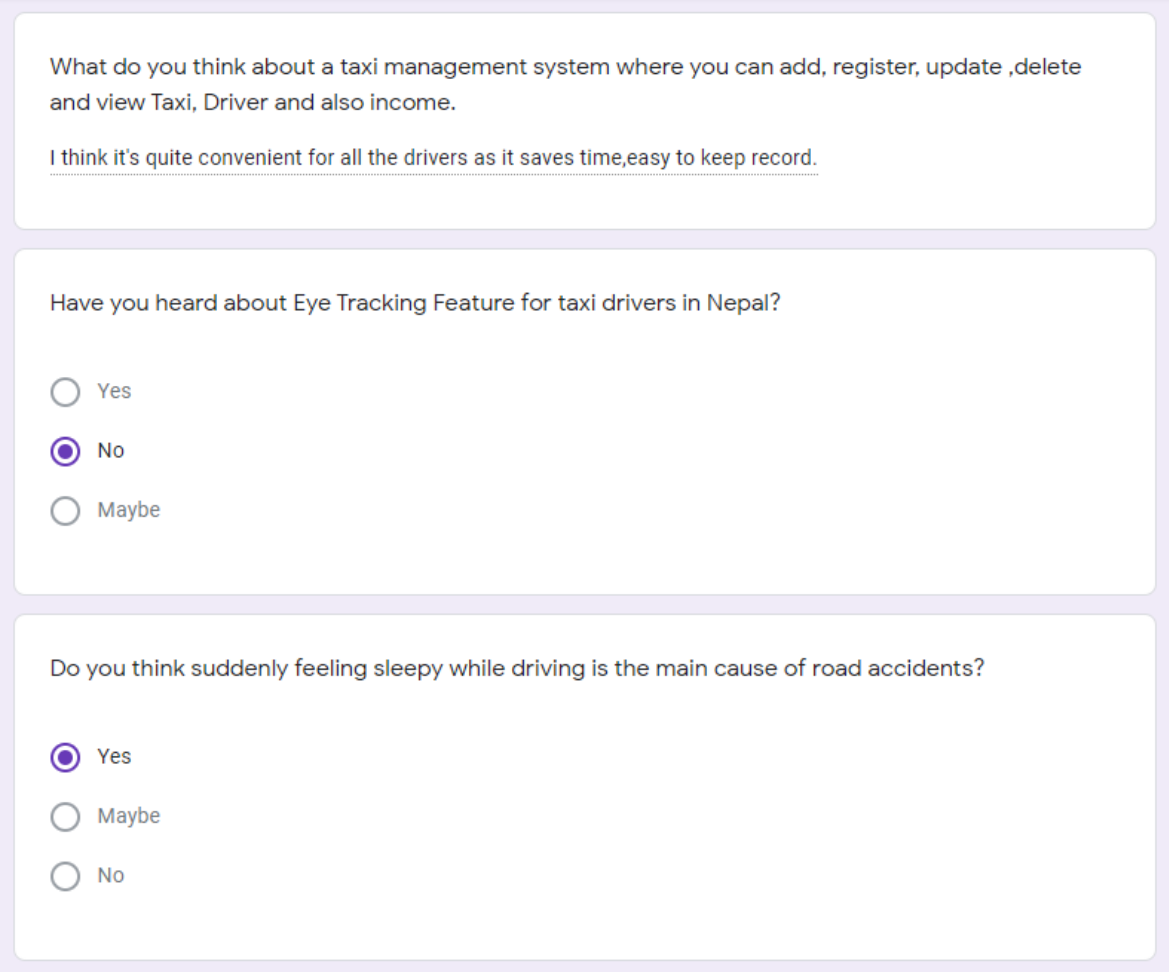


Figure 166 Pre-Survey Sample (2)

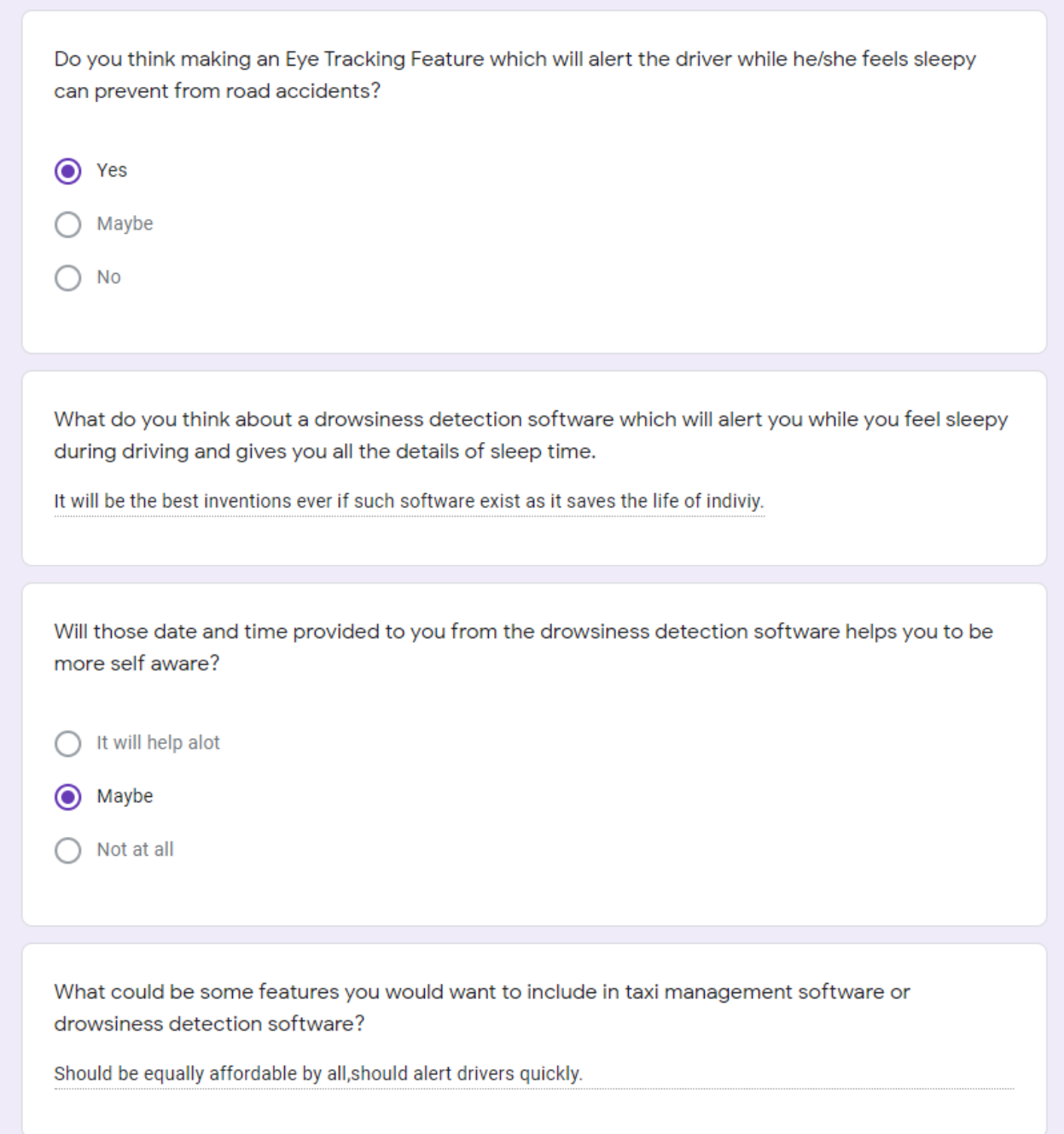


Figure 167Pre-Survey Sample (3)

8.2 APPENDIX B: POST-SURVEY

8.2.1 POST-SURVEY FORM

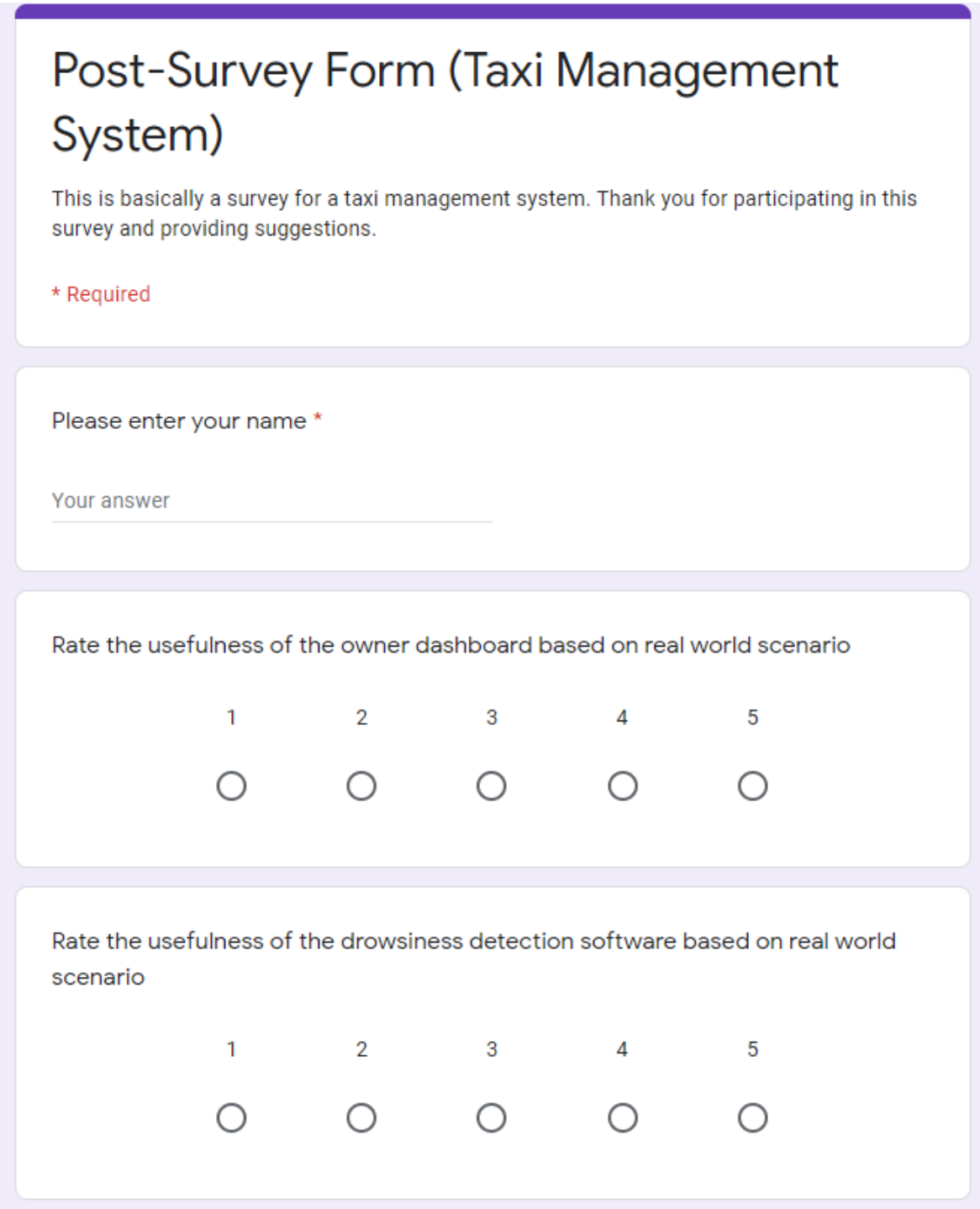


Figure 168 Post Survey (1)

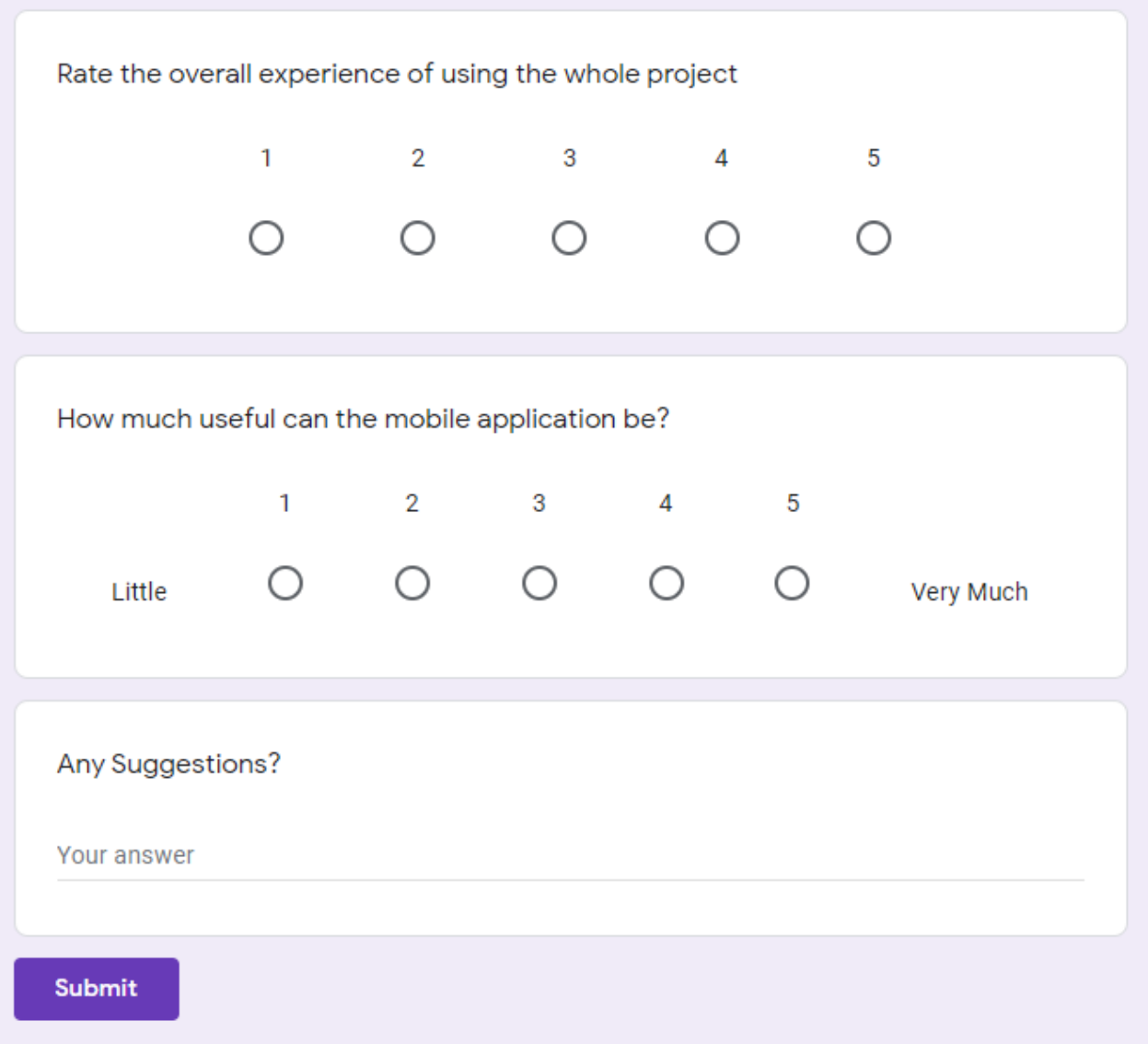


Figure 169 Post Survey (2)

8.2.2 SAMPLE OF FILLED POST-SURVEY FORMS

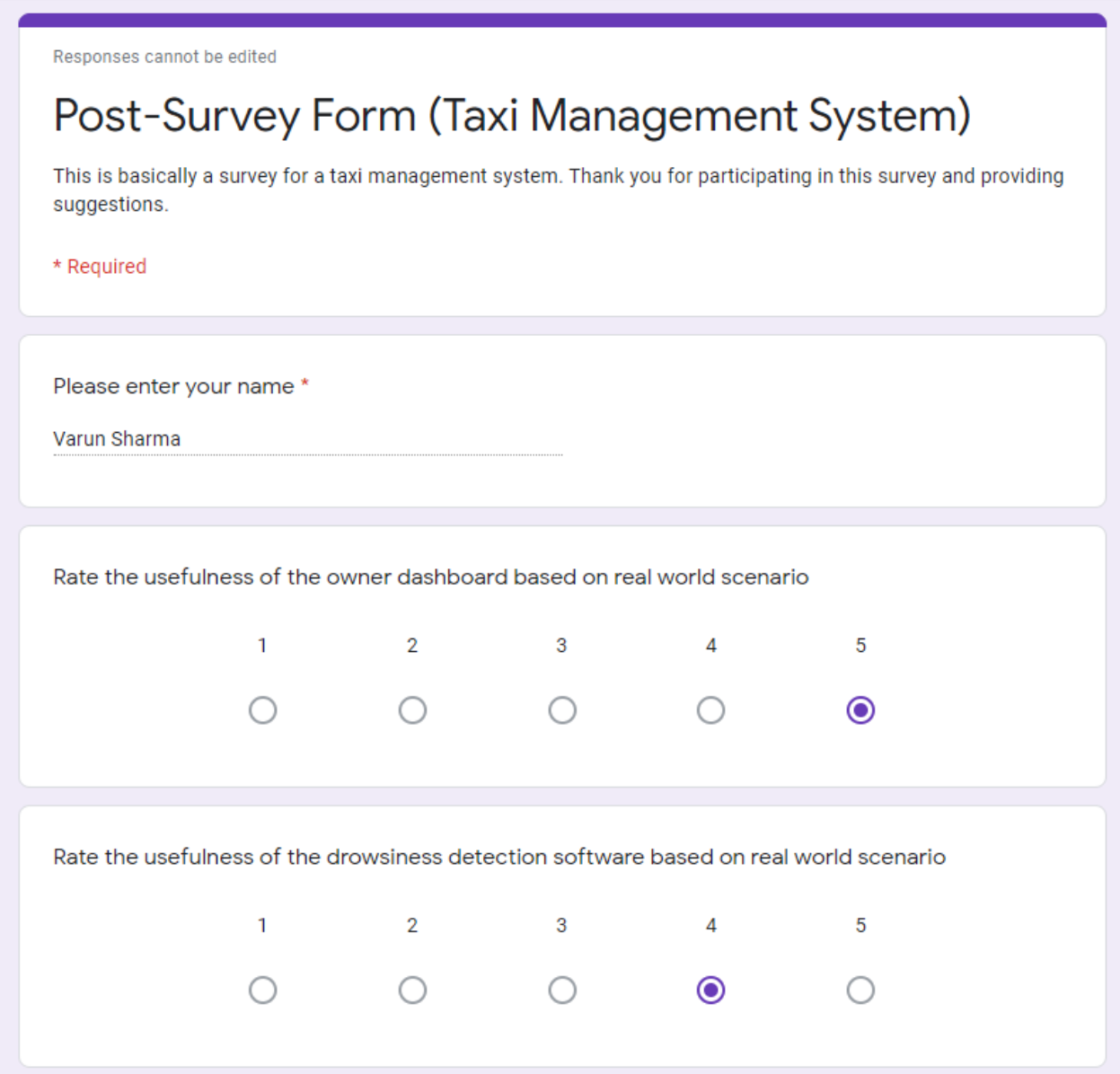


Figure 170 Post Survey Sample (1)

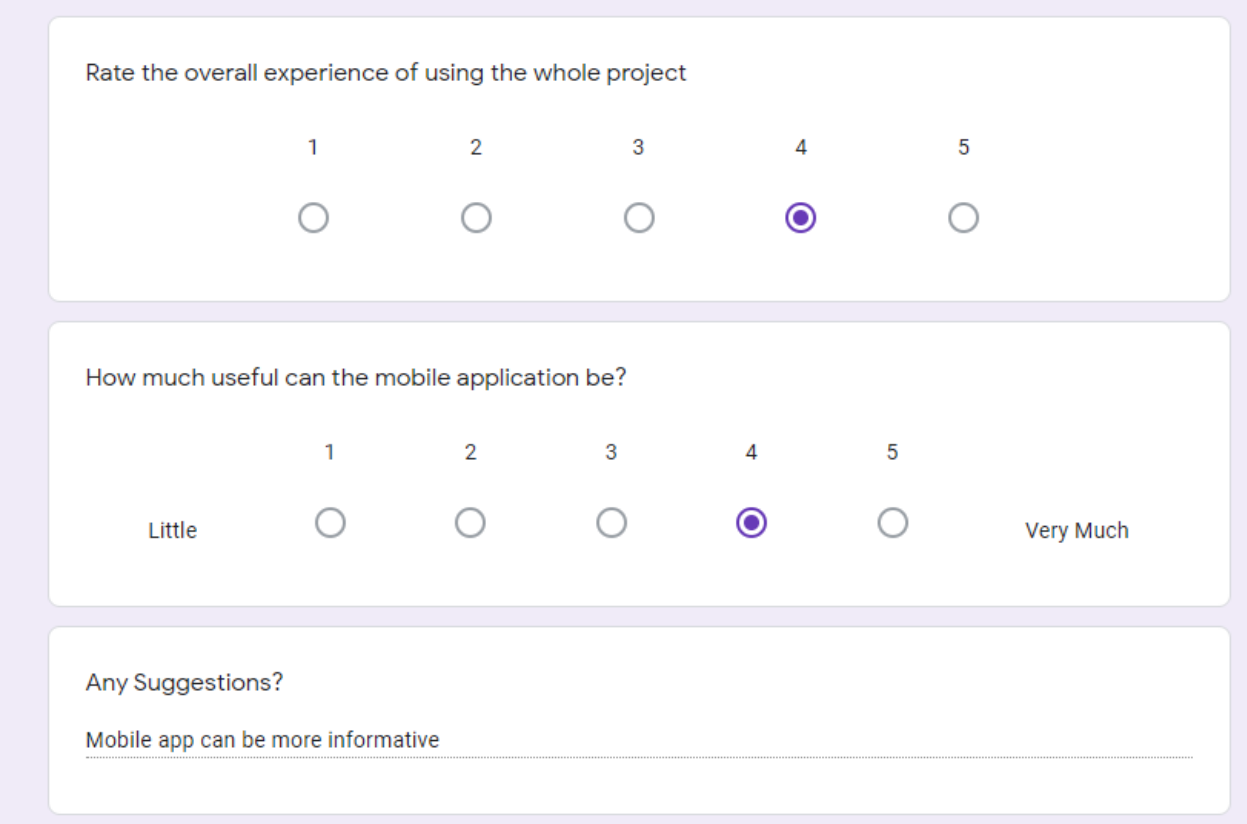


Figure 171 Post Survey Sample (2)

8.3 APPENDIX C: CODES

8.3.1 CODE OF THE UI

```
<!-- Logout Modal-->
<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"
    aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                <button class="close" type="button" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">x</span>
                </button>
            </div>
            <div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>
            <div class="modal-footer">
                <a class="btn btn-primary" href="{% url 'Taxi_Manager:Logout' %}">Logout</a>
                <button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>
            </div>
        </div>
    </div>
</div>
```

Figure 172 Logout.html code

```
    </ul>
    <!-- End of Sidebar -->

  <!-- Content Wrapper -->
  <div id="content-wrapper" class="d-flex flex-column">

    <!-- Main Content -->
    <div id="content">

      <!-- Topbar -->
      <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">

        <!-- Sidebar Toggle (Topbar) -->
        <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-3">
          <i class="fa fa-bars"></i>
        </button>

        <!-- Topbar Navbar -->
        <ul class="navbar-nav ml-auto">

          <div class="topbar-divider d-none d-sm-block"></div>

          <!-- Nav Item - User Information -->
          <li class="nav-item dropdown no-arrow">
            <a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <span class="mr-2 d-none d-lg-inline text-gray-600 small">{{ user.username|default:'Guest' }}</span>
              <img class="img-profile rounded-circle" src="https://source.unsplash.com/QAB-WJcbgJk/60x60">
            </a>
            <!-- Dropdown - User Information -->
            <div class="dropdown-menu dropdown-menu-right shadow animated--grow-in" aria-labelledby="userDropdown">
              <div class="dropdown-divider"></div>
              <a class="dropdown-item" href="{% url 'Taxi_Manager:Logout' %}" data-toggle="modal" data-target="#logoutModal">
                <i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                Change Password
              </a>
              <a class="dropdown-item" href="{% url 'Taxi_Manager:Logout' %}" data-toggle="modal" data-target="#logoutModal">
                <i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                Logout
              </a>
            </div>
          </li>

        </ul>

      </nav>
      <!-- End of Topbar -->
```

Figure 173 Password Change (1)

```
    </div>
    <!-- /.container-fluid -->
    <div class="container-fluid">
      <div class="card mb-12">
        <div class="card shadow mb-4">
          <div class="card-header">
            <h4 class="m-0 font-weight-bold text-primary" style="text-align: center">Change Password</h4>
          </div>

          <div class="card-body">

            <div class="form-group">
              <form method="post">
                {% csrf_token %}
                <div class="form-group">
                  {% render_field form.old_password class="form-control form-control-user" placeholder="Old Password" %}
                </div>
                <div class="form-group">
                  {% render_field form.new_password1 class="form-control form-control-user" placeholder="New Password" %}
                </div>
                <div class="form-group">
                  {% render_field form.new_password2 class="form-control form-control-user" placeholder="New Password Confirmation" %}
                </div>
                <input type="submit" class="btn btn-primary" value="Change">
                <a href="{% url 'Taxi_Manager:drowsiness' %}" class="btn btn-secondary">Cancel</a>
              </form>
            </div>

          </div>
        </div>
      </div>
    </div>

  </div>
  <!-- End of Main Content -->

  <!-- Footer -->
  <footer class="sticky-footer bg-white">
    <div class="container my-auto">
      <div class="copyright text-center my-auto">
        <span>Copyright &copy; Meru Sangroula 2020</span>
      </div>
    </div>
  </footer>
  <!-- End of Footer -->
```

Figure 174 Password Change (2)

```python
class PasswordChange(auth_views.PasswordChangeView):
    template_name = 'registration/password_change.html'
    success_url = reverse_lazy('Taxi_Manager:password-change-complete')


@login_required(login_url="/")
def password_change_complete(request):
    return render(request, 'registration/password_change_complete.html')
```

Figure 175 password change view

```
{% load static %}

{% load widget_tweaks %}

<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>{% block title %}{% endblock %}</title>

    <!-- Custom fonts for this template-->
    <link href="{% static 'vendor/fontawesome-free/css/all.min.css' %}" rel="stylesheet" type="text/css">
    <link
        href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i"
        rel="stylesheet">

    <!-- Custom styles for this template-->
    <link href="{% static 'css/sb-admin-2.min.css' %}" rel="stylesheet">

    <!-- Custom styles for this page -->
    <link href="{% static 'vendor/datatables/dataTables.bootstrap4.min.css' %}" rel="stylesheet">

    <!-- CSS Links for FULL CALENDAR -->
    <link
        href="https://cdn.jsdelivr.net/combine/npm/fullcalendar@5.3.2/main.min.css,npm/fullcalendar@5.3.2/main.min.css"
        rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/fullcalendar@5.3.2/main.min.css" rel="stylesheet">

    <!-- jQuery -->
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <!--external JS-->
    <script src="{% static 'js/forCharts.js'%}"></script>
    <!-- Custom styles for this template-->
    <link href="{% static 'css/sb-admin-2.min.css' %}" rel="stylesheet">

    {# Include FontAwesome; required for icon display #}
    <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.css">

    {# Include Bootstrap 4 and jQuery #}
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
        integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
```

Figure 176 base.html (1)

```
<li class="nav-item">
    <a class="nav-link" href="{% url 'Taxi_Manager:home' %}">
        <i class="fas fa-fw fa-tachometer-alt"></i>
        <span>Dashboard</span></a>
</li>

<!-- Divider -->
<hr class="sidebar-divider">

<!-- Nav Item - Register - Collapse Menu -->
<li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseRegister"
        aria-expanded="true" aria-controls="collapseRegister">
        <i class="fas fa-fw fa-folder"></i>
        <span>Register</span>
    </a>
    <div id="collapseRegister" class="collapse" aria-labelledby="headingRegister"
        data-parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
            <h6 class="collapse-header">Registration Menu:</h6>
            <a class="collapse-item" href="{% url 'Taxi_Manager:register-taxi' %}">Taxi</a>
            <a class="collapse-item" href="{% url 'Taxi_Manager:register-driver' %}">Driver</a>
        </div>
    </div>
</li>

<!-- Nav Item - Profile - Collapse Menu -->
<li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseView"
        aria-expanded="true" aria-controls="collapseProfile">
        <i class="fas fa-fw fa-table"></i>
        <span>View</span>
    </a>
    <div id="collapseView" class="collapse" aria-labelledby="headingProfile"
        data-parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
            <h6 class="collapse-header">List Profiles:</h6>
            <a class="collapse-item" href="{% url 'Taxi_Manager:view-taxi' %}">Taxi</a>
            <a class="collapse-item" href="{% url 'Taxi_Manager:view-driver' %}">Driver</a>
        </div>
    </div>
</li>


<!-- Nav Item - Income -->
<li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseProfile"
        aria-expanded="true" aria-controls="collapseProfile">
        <i class="fas fa-fw fa-table"></i>
        <span>Income</span>
```

Figure 177 base.html (2)

```
<!-- End of Topbar -->

<!-- Begin Page Content -->
<div class="container-fluid">

  <!-- Page Heading -->
  <h1 class="h3 mb-2 text-gray-800"><center>Drowsiness Information</center></h1>

  <!-- DataTales Example -->
  <div class="card shadow mb-4">
    <div class="card-header py-3">
      <h4 class="m-0 font-weight-bold text-primary" style="text-align: center">Drowsiness</h4>
    </div>
    <div class="card-body">
      <div class="table-responsive">
        <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
          <thead>
            <tr>
              <th>ID</th>
              <th>Time of Drowsiness</th>
            </thead>
            <tbody>

              {% for drow in drows %}
                <tr>
                  <td>{{ drow.id }}</td>
                  <td>{{ drow.drowsiness }}</td>
                </tr>
              {% endfor %}
            </tbody>
          </table>
        </div>
      </div>
    </div>

  </div>
  <!-- /.container-fluid -->

</div>
<!-- End of Main Content -->

<!-- Footer -->
<footer class="sticky-footer bg-white">
  <div class="container my-auto">
    <div class="copyright text-center my-auto">
      <span>Copyright &copy; Meru Sangroula</span>
    </div>
```

Figure 178 Drowsiness.html (1)

```
<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Sidebar -->
    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar">

      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-center"
          href="{% url 'Taxi_Manager:drowsiness' %}">
          <div class="sidebar-brand-icon rotate-n-15">
              <i class="fas fa-taxi"></i>
          </div>
          <div class="sidebar-brand-text mx-3">Hello have a nice day!</div>
      </a>

      <!-- Divider -->
      <hr class="sidebar-divider my-0">


      <!-- Divider -->
      <hr class="sidebar-divider">


      <!-- Nav Item - Income -->
      <li class="nav-item active">
        <a class="nav-link" href="{% url 'Taxi_Manager:drowsiness' %}">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Drowsiness Data</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
          <button class="rounded-circle border-0" id="sidebarToggle"></button>
      </div>

  </ul>
    <!-- End of Sidebar -->

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
```

Figure 179 Drowsiness.html (2)

8.4 APPENDIX D: DESIGNS

8.4.1 GANTT CHART

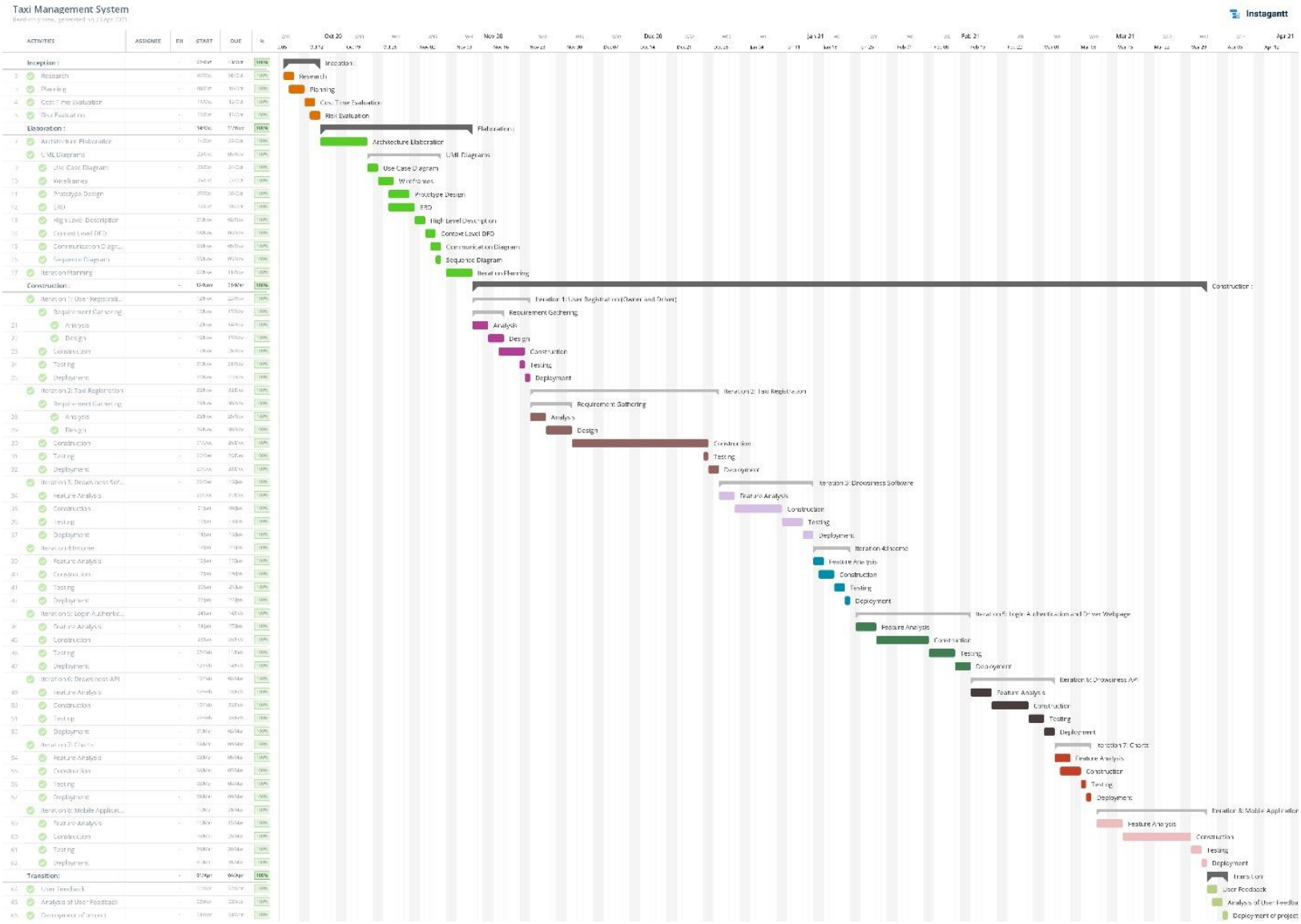


Figure 180 Gantt Chart

8.4.2 WORK BREAKDOWN STRUCTURE

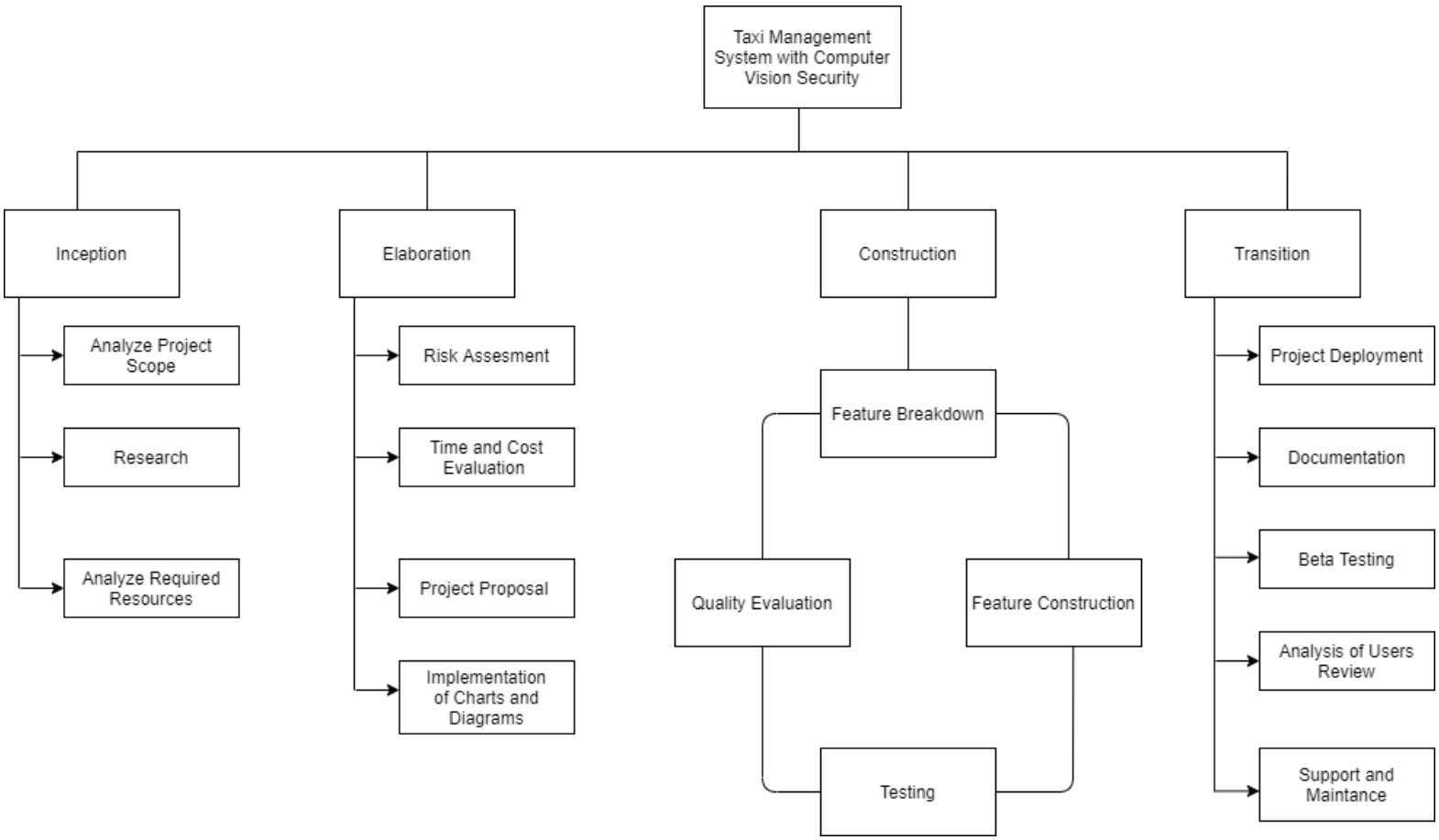


Figure 181 Work Breakdown Structure

8.4.3 MILESTIONE

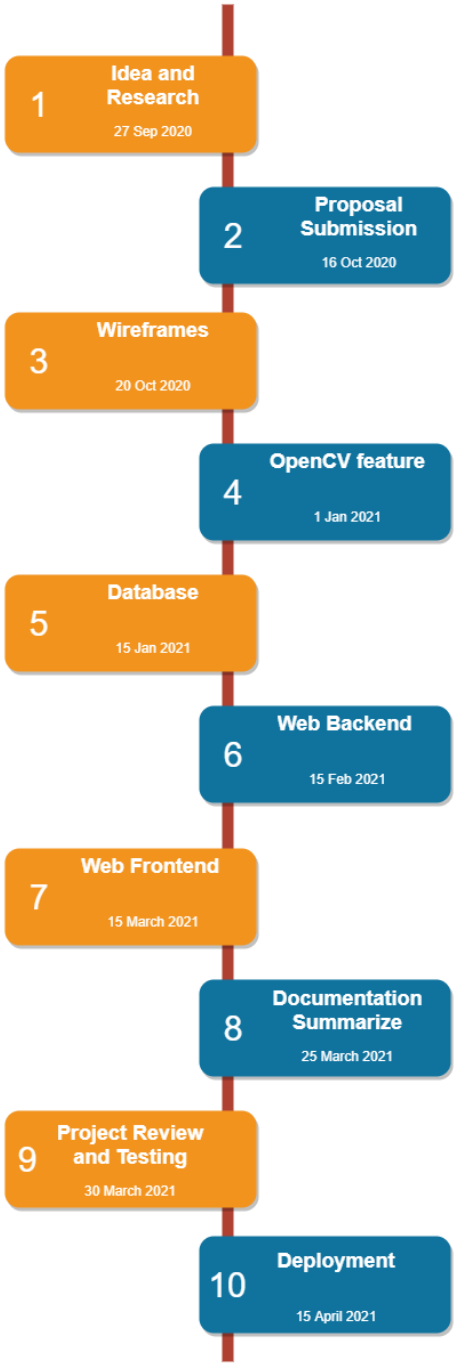


Figure 182 Milestone

8.4.4 HIGH-LEVEL USECASE

• **Name:** Owner Login
• **Actor(s):** Owner (Primary)
• **Description:** Owner first needs to login into the web app before using all the features of the web app. Owner will be asked their respective username and password. Wrong username or wrong password results in login failure.

• **Name:** Register Taxi
• **Actor(s):** Owner (Primary)
• **Description:** Owner can register taxi using the registration form which will appear in the web app where owner needs to enter every details of the taxi.

• **Name:** View Taxi
• **Actor(s):** Owner (Primary)
• **Description:** After the registration, owner can view all the registered taxi. The owner can also view the detail of individual taxi and can update or delate the taxi as well.

• **Name:** Register Driver
• **Actor(s):** Owner (Primary)
• **Description:** Owner can register taxi using the registration form which will appear in the web app where owner needs to enter every details of the driver and also the taxi number which the driver is going to use.

• **Name:** View Driver
• **Actor(s):** Owner (Primary)

• **Description:** After the registration, owner can view all the registered driver. The owner can also view the detail of individual driver and can update or delate the driver as well.

• **Name:** Driver Login
• **Actor(s):** Driver (Primary)
• **Description:** Driver first needs to login to the web app for using different features.

• **Name:** Daily Income
• **Actor(s):** Owner (Primary)
• **Description:** Owner can update the daily income of the drivers earned the whole day. Only the owner can view the total income generated in the web app.

• **Name:** View Charts
• **Actor(s):** Owner (Primary)
• **Description:** Owner can view different data in the form of charts. Like the age of total driver represented in the form of charts, or gender distinguishing in charts.

• **Name:** View Drowsiness Data
• **Actor(s):** Driver (Primary)
• **Description:** Drowsiness data means the driver can view the date and time when he/she got the alarm of drowsiness while in work. This can help the driver to track the exact time for his/her microsleep while driving so that the driver can be careful about the next time.

8.4.5 WIREFRAME

8.4.5.1 For Web Application

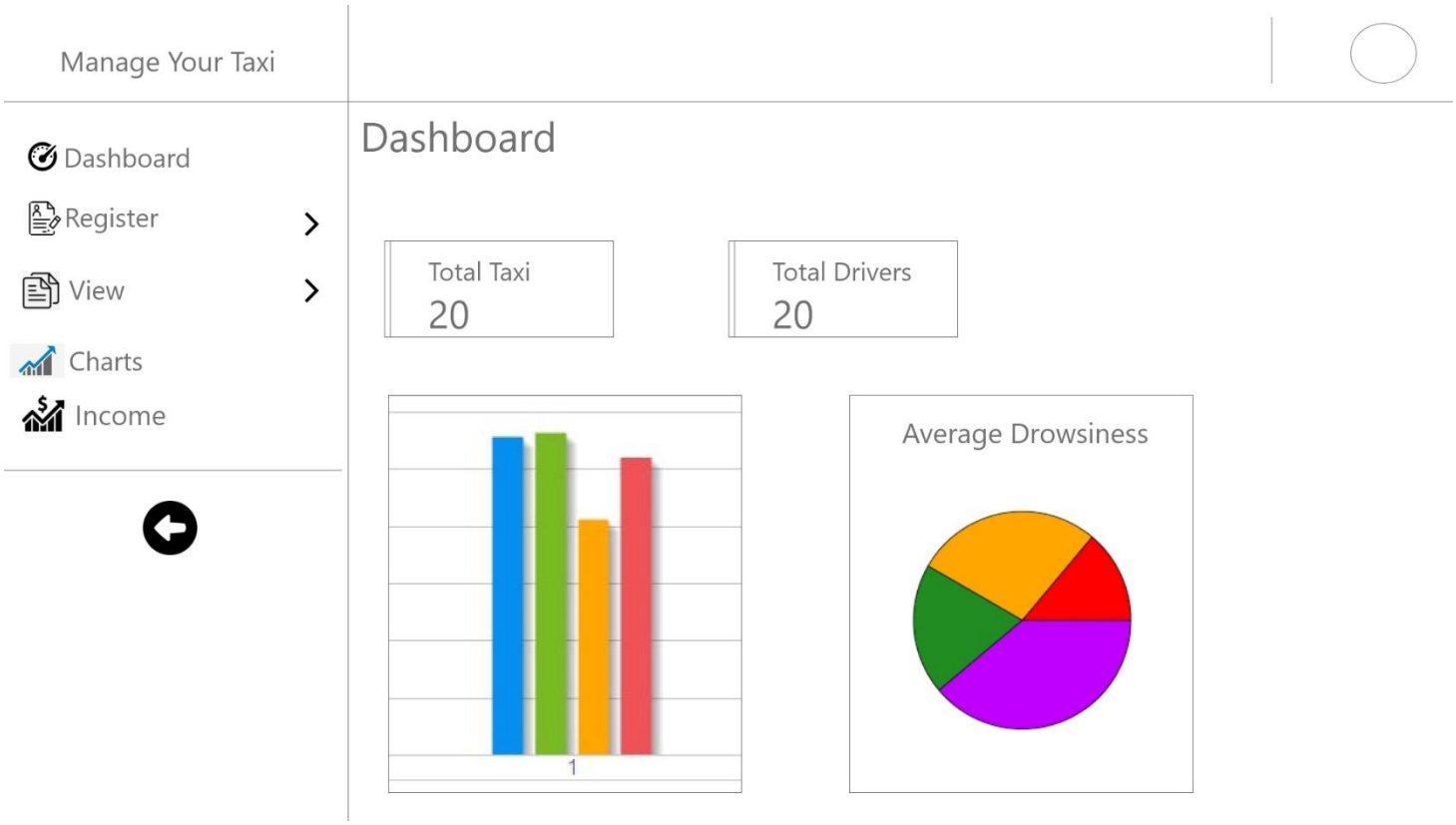


Figure 183 Wireframe 1 Dashboard

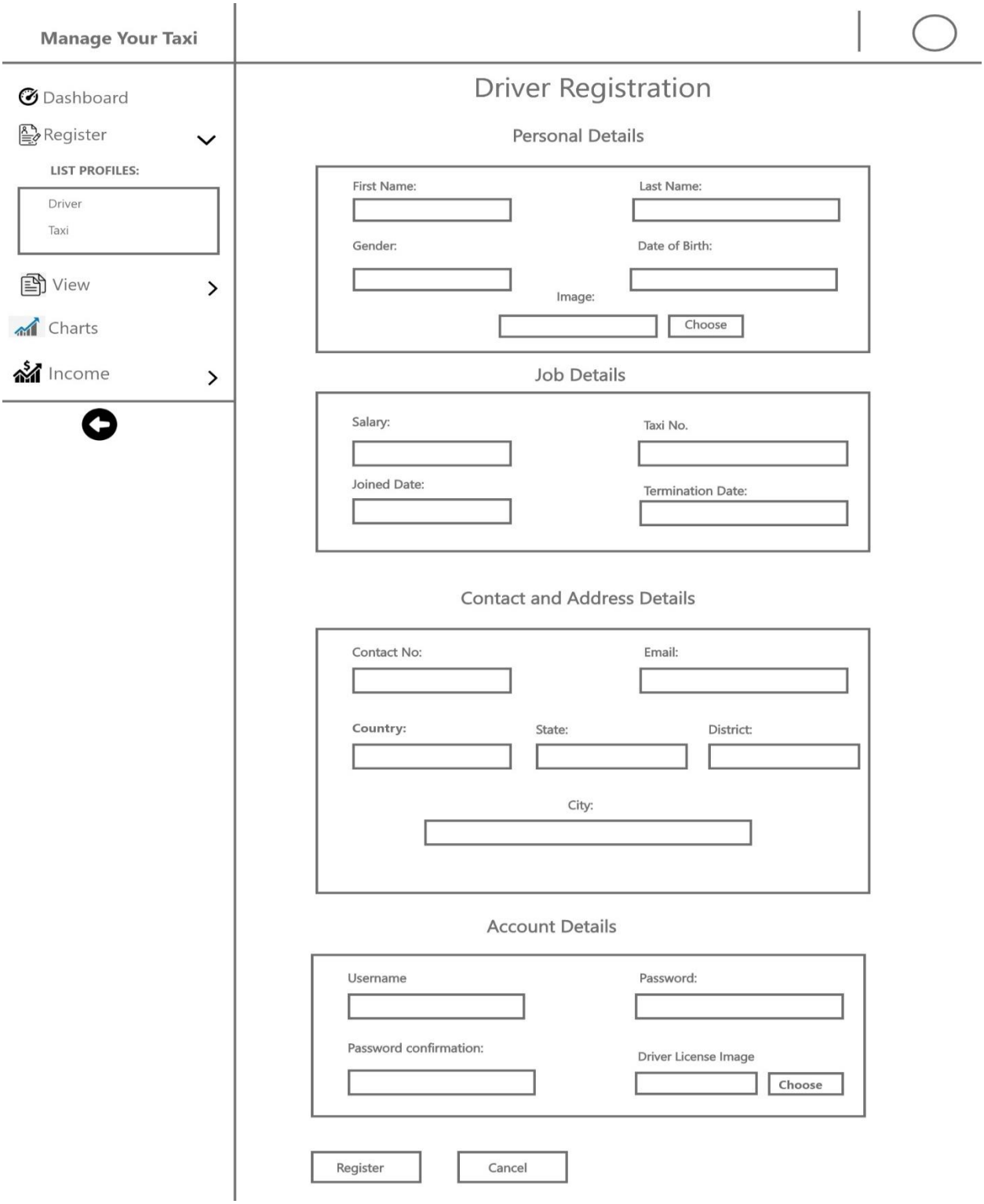


Figure 184 Wireframe 2 Driver Registration

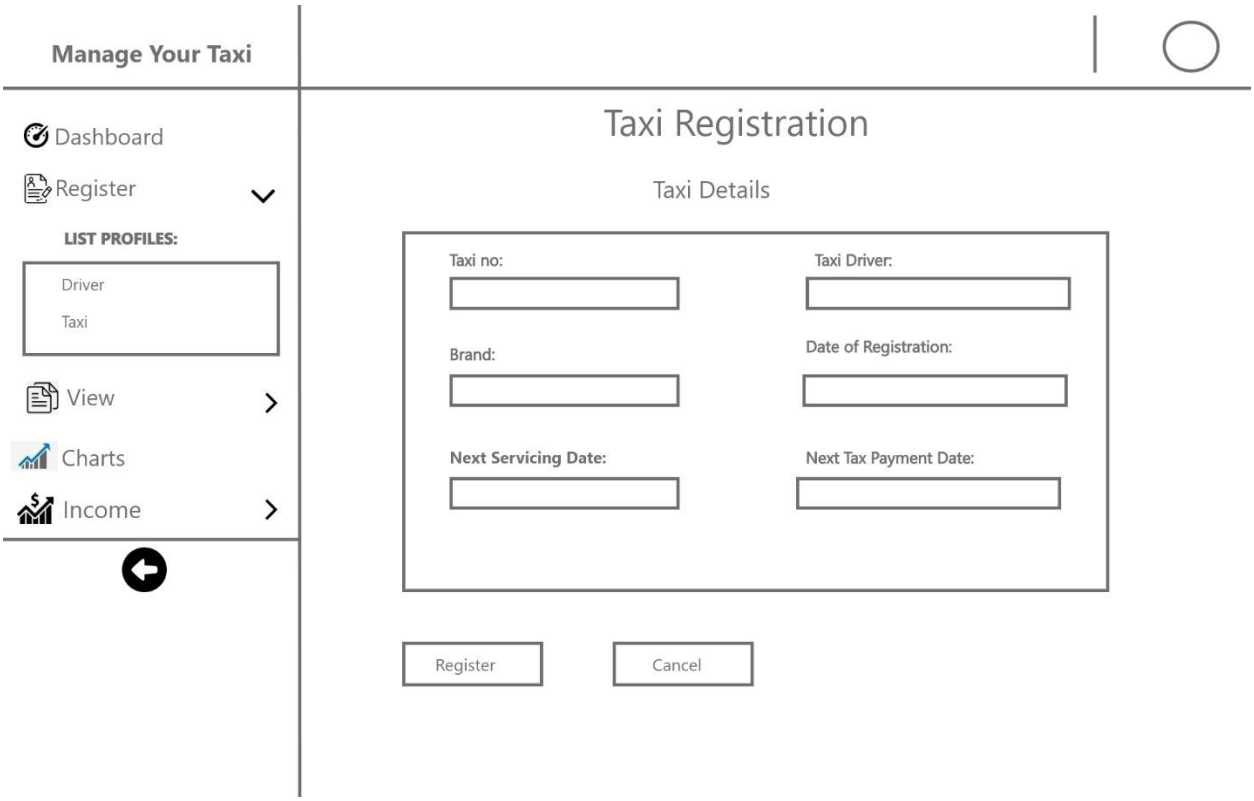


Figure 185 Wireframe 3 Taxi Registration

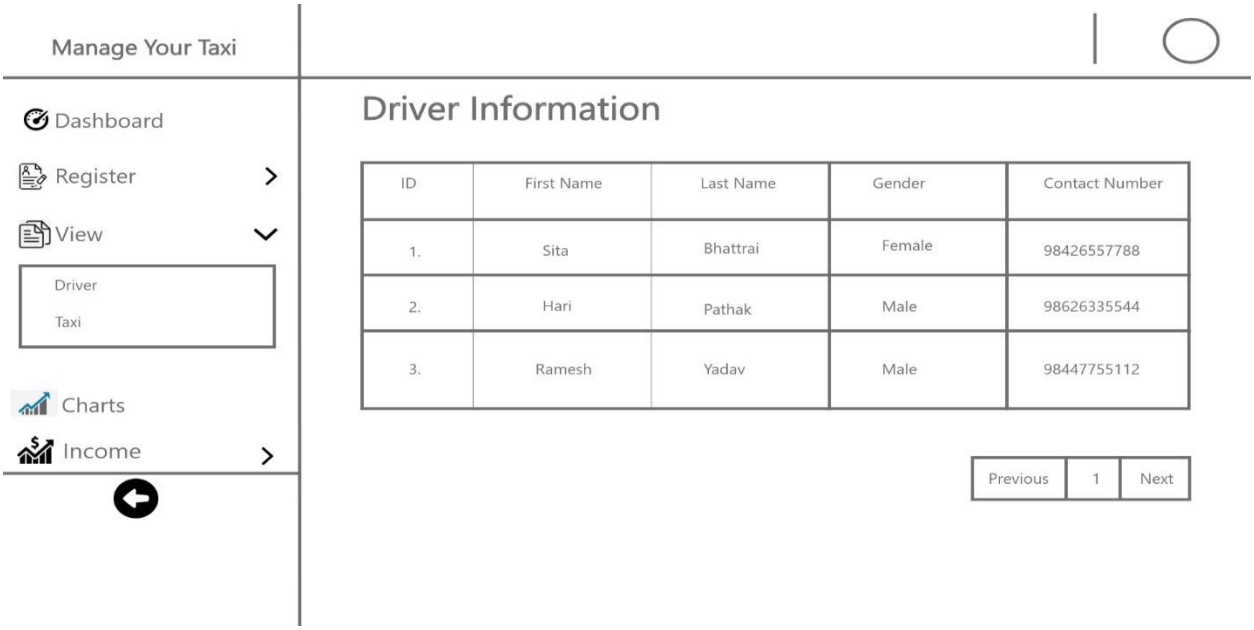


Figure 186 Wireframe 4 Driver Information

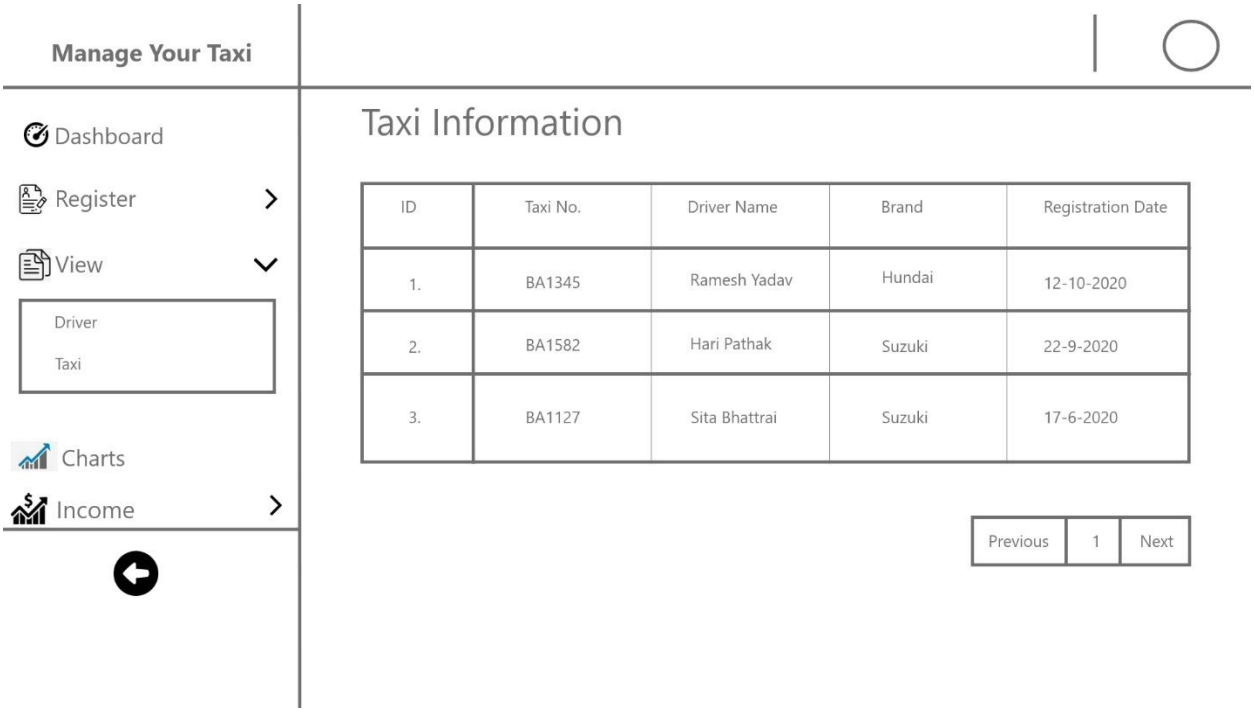


Figure 187 Wireframe 5 Taxi Information

Figure 188 Wireframe 6 Update Driver Registration

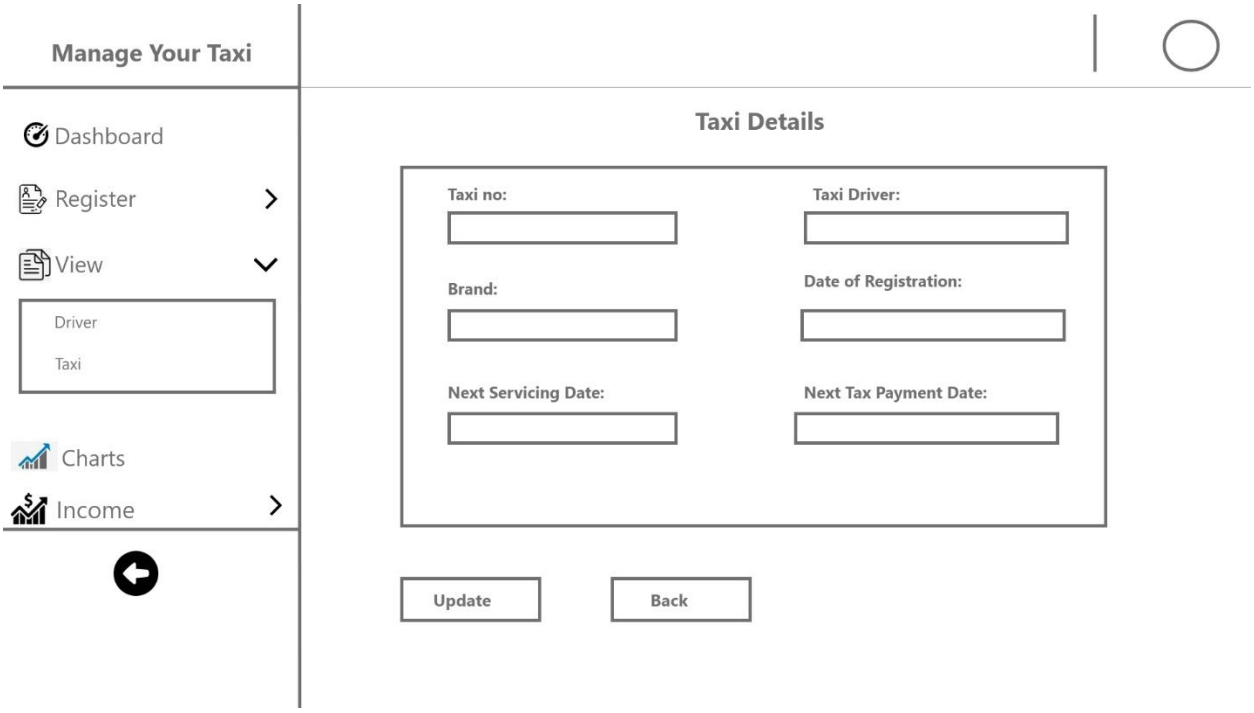


Figure 189 Wireframe 7 Update Taxi Details

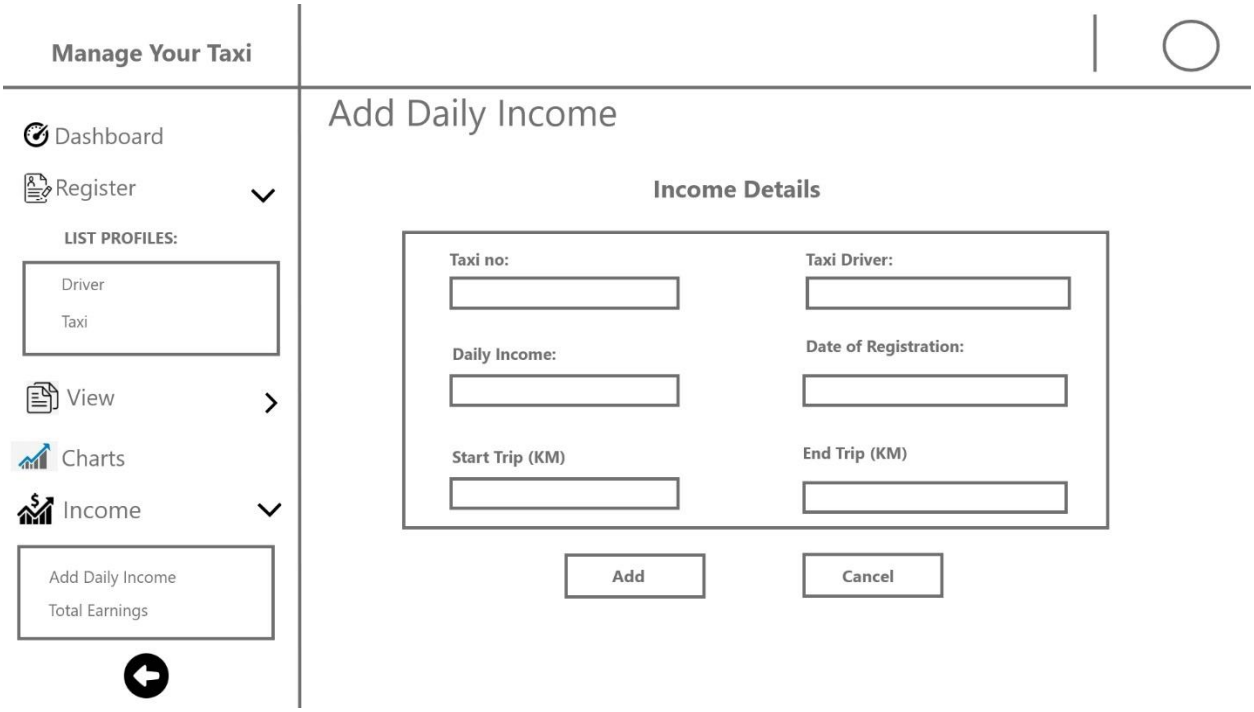


Figure 190 Wireframe 8 Add Daily Income

Figure 191 Wireframe 9 View Total Earning

8.4.5.2 For Mobile Application

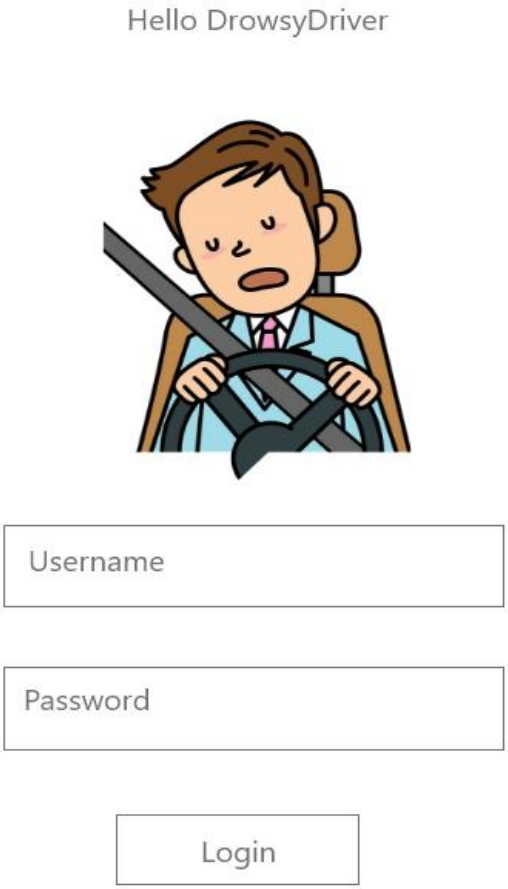


Figure 192 Wireframe 10 Mobile login

Figure 193 Wireframe 11 View Drowsiness Data

8.5 APPENDIX E: SOFTWARE REQUIREMENTS SPECIFICATION **(SRS)** DOCUMENT

8.5.1 Introduction

8.5.1.1 Purpose

The project which I will be working on is Taxi Management System with Computer Vision Security. This result of this project will be a web application to keep track of taxi business and a prototype working python script which will work as an eye tracking computer vision technology. What I am trying to do is track the eye of the taxi driver and protect the driver from accidents caused by drowsiness.

To keep track of your own business can be very stressful. Let me give an example of a Taxi business owner who must keep track of his/her 30 Taxi. There is a very less chance of remembering all of his/her taxis number and the detail of the driver as well. Many new drivers could also join in the middle. So, focusing on this problem I have decided to make a Web app for taxi business owners that can ease this workload from them.

8.5.1.2 Indented Audiences and Reading Suggestions

My main audiences are owner of small or large taxi company. Drivers are also the intended audience of my project. To prevent from road accident caused through microsleep, Drowsiness Detection Software will be installed on the taxi which is operated by the same company who buys the project. This will be the main feature of the project which will attract the end user even more.

8.5.1.3 Project Scope

According to my own research and survey, I have found that majority of taxi drivers do not buy taxi of their own instead they do contract with small taxi running company which is owned by a single person. Basically, one person buys 5-10 taxis and run a taxi business. The main client of this project are those business owners.

This project has a future because these small taxi businesses do not use any software and people face a lot difficulty once the taxi business extends. As a solution to this problem, my project will help them run their business more easily in less time. It will keep track of income, taxis and driver and will also provide security to the drivers.

8.5.2 Overall Description

8.5.2.1 System Perspective

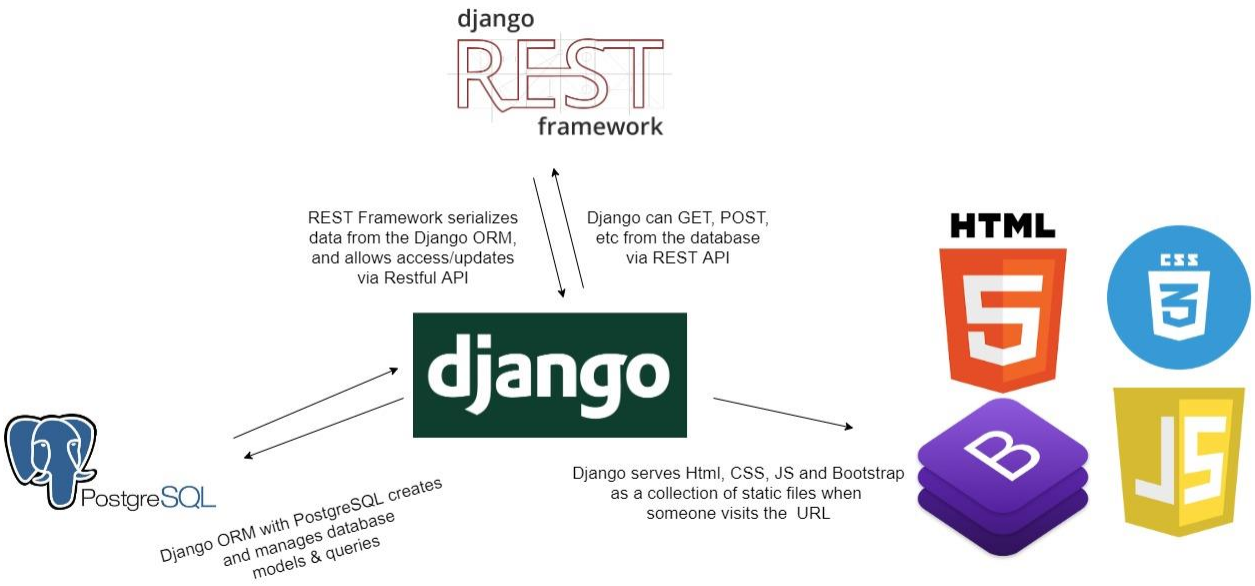


Figure 194 System Architecture

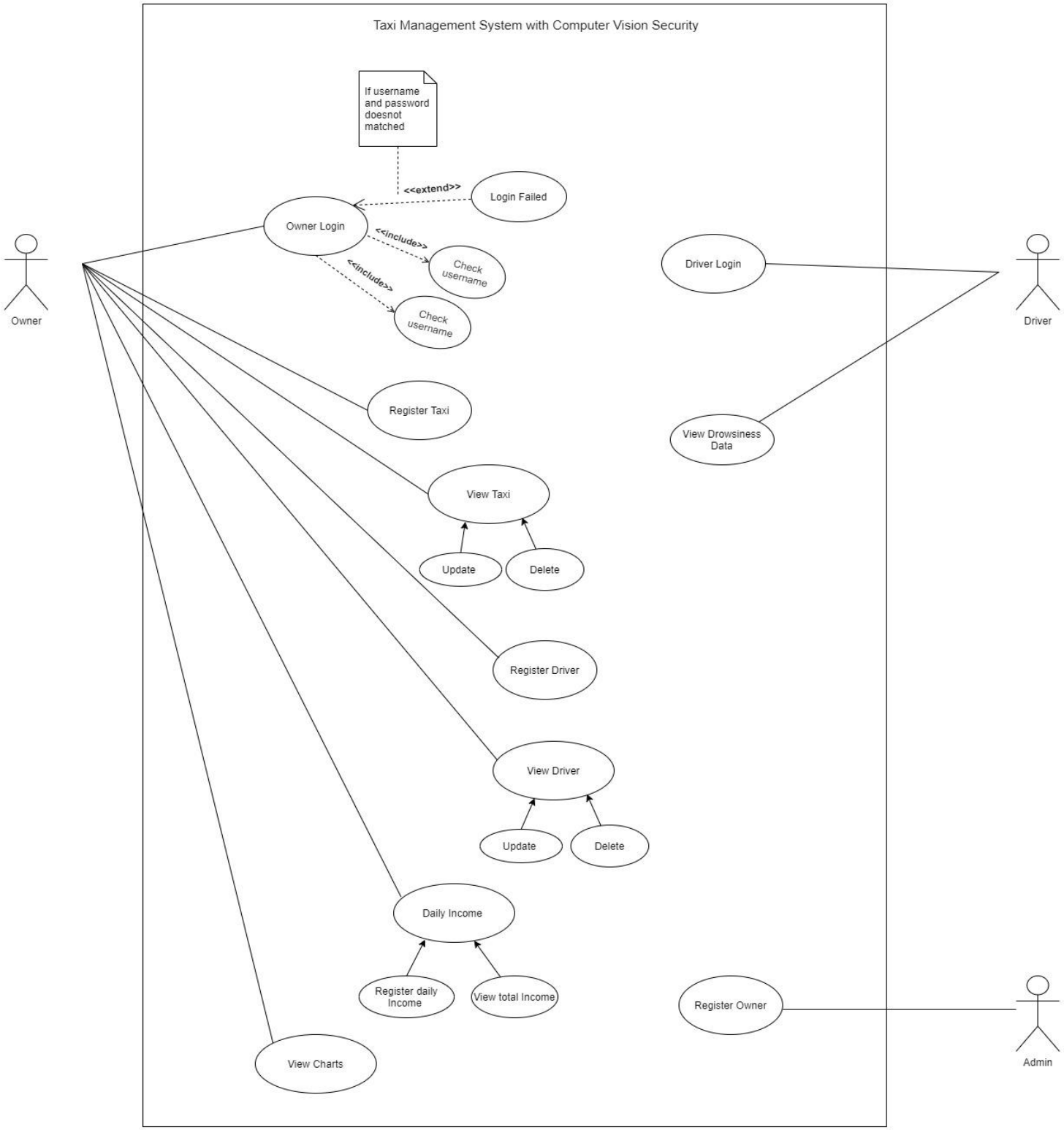


Figure 195 Use Case

8.5.2.2 System Features

The following are the primary features of the project:

- Login Authentication
  Both Owner and Driver has their own pages, and both needs login verification. Driver has no access in Owner page and same goes to driver.
- Owner Login and CRUD Operation
  Owner can register, view, delete and update taxi, and driver. Further, owner can add daily income and view the total income and profits gained by the business.
- Drowsiness Detection
  This feature basically alerts the driver when the driver feels drowsy. It tracks the eyes of the driver while driving and when the driver's eyes are closed then it fires the alarm and records the date and time of the alarm.
- Driver Login and View the drowsiness data
  This project also contains driver login where driver can see all the drowsiness data recorded and can be self-aware.

8.5.2.3 User class and characteristics

The following are the classes of users who will have specific roles of operation with Application and its reports:

- Normal User (Business Owner)
  This project basically gives a dashboard to the business owner. The owner will have the main role to control all the activities like registration and CRUD operations. The owner takes the responsibility to update the dashboard every day. Whenever an old driver quits the job or new driver is hired, it's the responsibility of the owner to make changes in the dashboard.

- Driver (Who works on that business)
  This project also has driver as a user as he/she can login and see their own drowsiness stats. While riding the taxi during work, the AI will track the eyes of the

driver to see whether he/she is sleepy or not which will not only prevent the life of driver but will also prevent the life of other passenger and the damage of taxi.

8.5.2.4 Operating Environment

- In the present context this project is focused on only one platform which is web. In future other platform like drowsiness data of driver showing in an app shall be taken into consideration.
- For now, the application is made able to operate only through browser.

8.5.2.5 Design and Implementation Constraints

- The owner cannot register himself; the system admin will have to register the owner.
- The driver can only login and see his drowsiness data.
- The camera (Laptop Camera) is not enough to catch the eye ratio in dark, instead night vision or lidar technology can be connected to the device.

8.5.2.6 Assumptions and Dependencies

- The application is targeted for taxi business to keep their records and run their business smoothly.
- Since the distraction software is a prototype software, so the code must run in the computer which will be kept next to driver seat. This software does not fulfill the real-world implications.
- The owner can use the project daily and can implement changes daily.
- The drowsiness feature is assumed to be for the driver of that particular company who will use the web app. It cannot be separated and use separately.

8.5.3 Functional Requirements

8.5.3.1 Workouts Feature

<table>
<tr><th>Req.ID</th><th>Requirement Description</th><th>Priority</th><th>Complexity</th></tr>
<tr><td>1.</td><td>User (Owner) login and can use all the features
<table>
<tr><td colspan="2">System Requirement</td></tr>
<tr><td>1.1</td><td>Owner has to login using verified username and password.</td></tr>
<tr><td>1.2</td><td>The owner will be redirected to dashboard where the owner can see all the charts related to the database.</td></tr>
<tr><td>1.3</td><td>The owner can register the taxi or driver, or the income generated by filling the forms given in the web app.</td></tr>
</table></td><td>Most</td><td>High</td></tr>
<tr><td>2.</td><td>User (Driver) login
<table>
<tr><td colspan="2">System Requirement</td></tr>
<tr><td>1.1</td><td>Driver has to login using verified username and password.</td></tr>
<tr><td>1.2</td><td>The driver will be redirected to home page where driver can see his/her drowsiness data.</td></tr>
</table></td><td>Most</td><td>High</td></tr>
</table>

Table 31 Workout Feature

8.5.3.2 Database Connection

Noting really needs to be done for database connection. Since this project will be using Django so while building the project the admin will connect the project to the database itself and owner can only use the web application without any database problem. The database used is PostgreSQL.

8.5.4 External Interfaces Requirements

8.5.4.1 User Interfaces

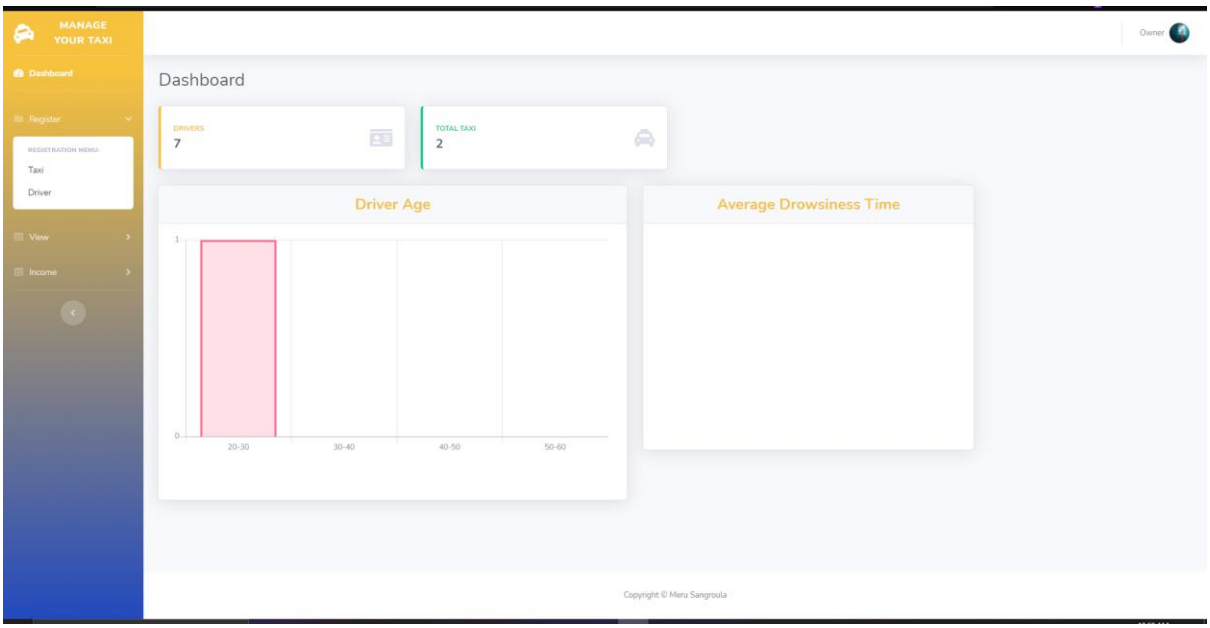


Figure 196 Dashboard

This is not the final UI of the dashboard, but it is just an example of how the dashboard could possibly be. Here after the Owner login he/she will get accessed to all the features.

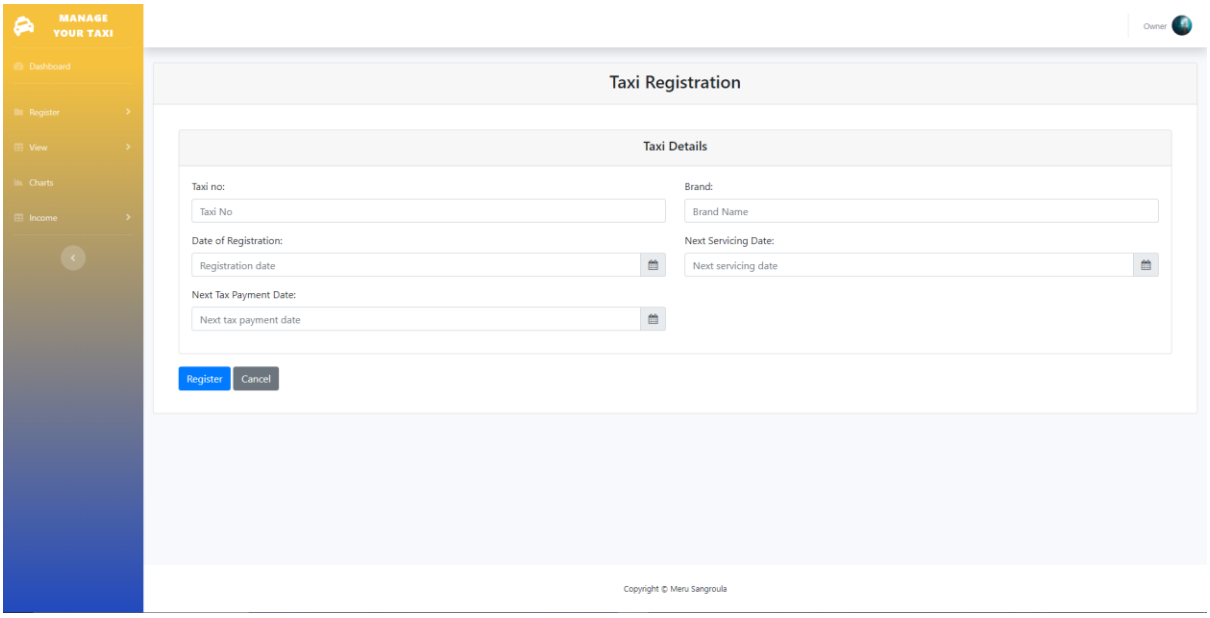


Figure 197 Taxi Registration

This is the registration form example where the owner will possibly register the new taxi or the driver.

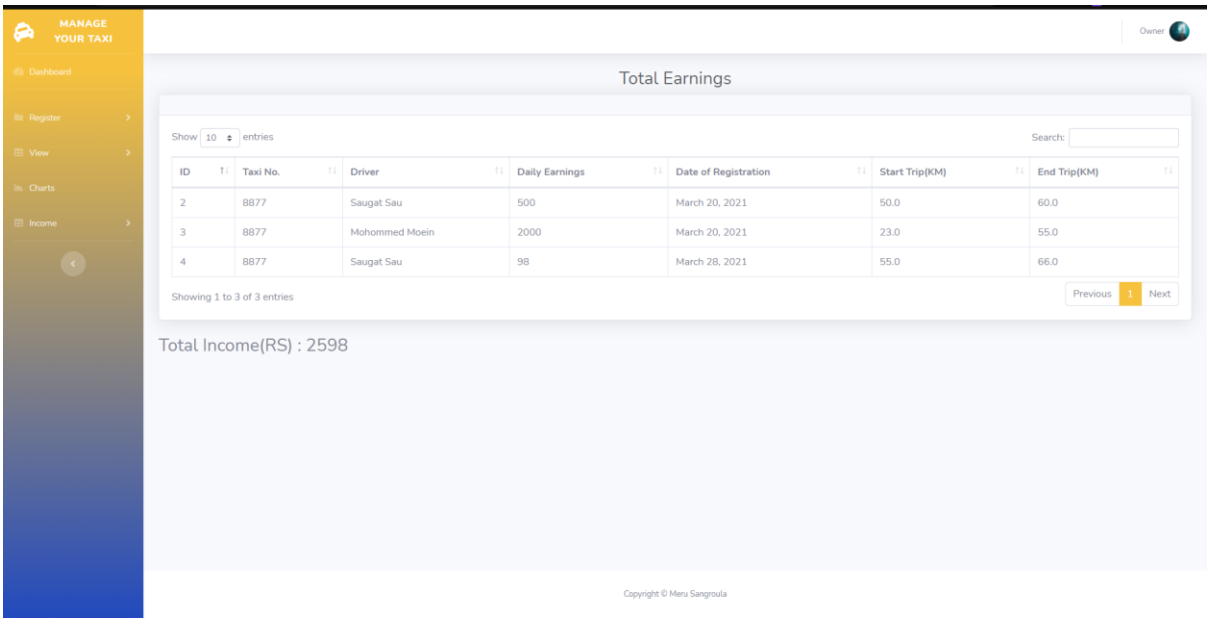


Figure 198 Total Earning

In this page, the user will see all the total incomes and earnings and other information.

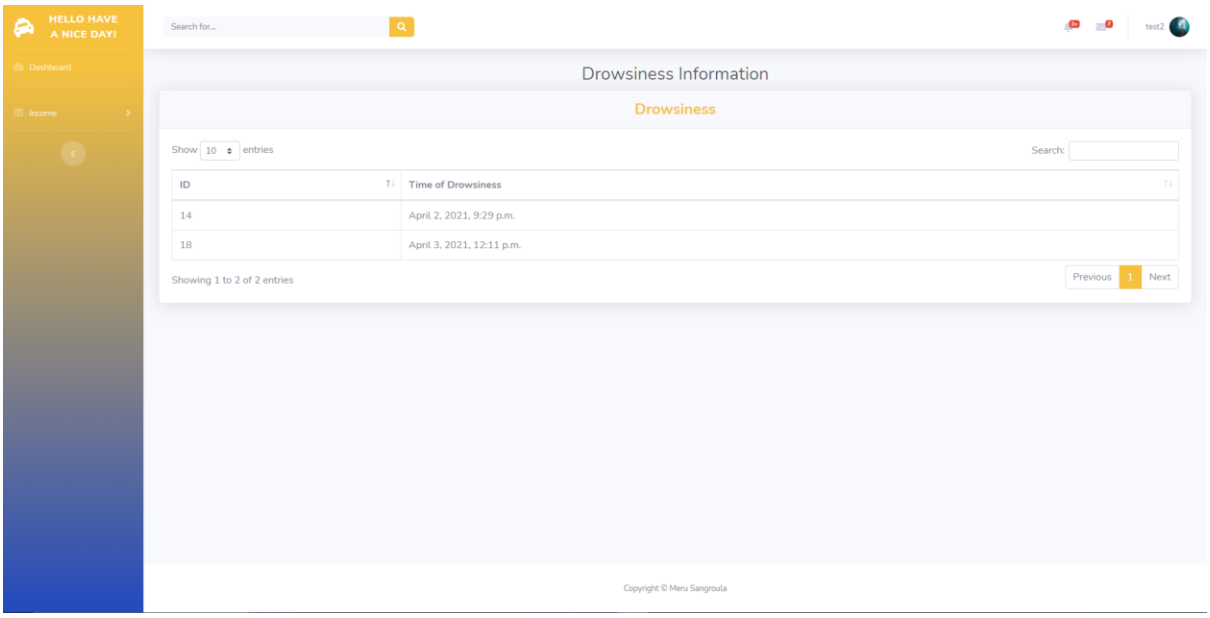


Figure 199 Driver Drowsiness

This project also has driver login where the driver will see his/her drowsiness data and get information of his/her sleep.

8.5.4.2 Hardware Interfaces

- Laptop

  Since my drowsiness feature is a working prototype, laptop is needed to run OpenCV code for driver distraction software. I will also use the built-in camera to detect the face of the driver.

- External Camera (Optional)

  Drowsiness feature need camera to detect face so, I need external camera which will be connected to my laptop or I can also detect face directly from the **built-in camera of laptop**.

8.5.4.3 Software Interfaces

- Django

  Django is a python-based opensource web framework. It follows MTV(Model-Template-View) pattern. A framework is nothing more than the series of modules that facilitates development. I am using Django for making taxi management web application and I choose Django because It is a very powerful web framework containing lots of built-in awesome features which will boost the production speed. Django offers authentication which can be used to build a fully functional authentication system by running a simple command. Along with that Django offers a lot of built-in features.

- OpenCV

  OpenCV is an open source computer vision and machine learning software. OpenCV is a library which has more than 2500 optimized algorithms that helps to detect, recognize faces. It can also track movements and many more. I am using OpenCV library for making Driver Distraction Detection software. I am using OpenCV specifically because it already has many face trained modules and will not complicate my project. Now-a-days many face recognized software are made using OpenCV.

- PostgreSQL

  PostgreSQL is an advance, enterprise-class and open-source relational database system (postgresqltutorial, 2020). I am using PostgreSQL because of the following reasons:

  - Open Source DBMS
  - Diverse Community
  - Diversified extension features
- HTML, CSS, Bootstrap, Java Script for frontend.
- Django RestAPI

Django REST framework is a powerful and flexible toolkit for making Web APIs. I am using Django RestAPI because of following reasons:

- Serialization supports both ORM and non-ORM data source
- Easy to use with Django

8.5.4.4 Other Non-Functional Requirements

8.5.4.4.1 Performance Requirements

Req.ID	Requirement Description	Priority	Complexity
1.	While opening the web app some browsers might not support the UI but in most of the cases it supports the UI.	Should	High
2.	While running the drowsiness script, the code should run properly without any delay and should track the eye.	Should	High

Table 32 Performance Requirement

8.5.4.4.2 Safety and Security Requirements

Req.ID	Requirement Description	Priority	Complexity
1.	Only verified users should be allowed to use the webapp.	Should	High
2.	Driver should have no authority to access the dashboard page.	Should	Medium
3.	The webapp should not create any malware problems in the background.	Should	Low

Table 33 Safety and Security Requirement

8.7 APPENDIX F: USER FEEDBACK

8.7.1 USER FEEDBACK FORM

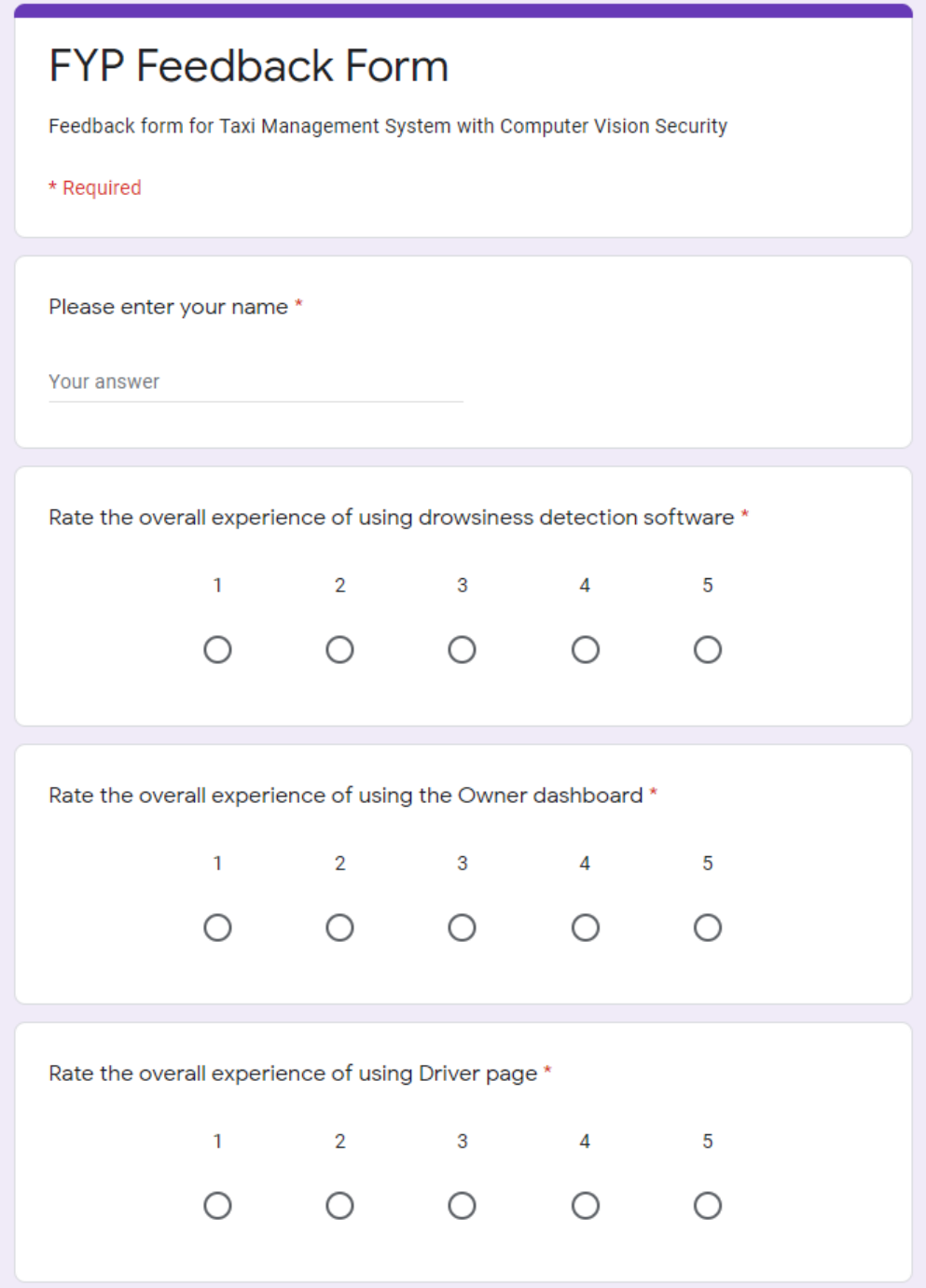


Figure 200 User Feedback form (1)

Rate the overall experience of using Mobile App

1 2 3 4 5

Rate the overall experience of using the project *

1 2 3 4 5

What do you like most about the project *

- Owner Dashboard
- Drowsiness Software
- Other: _____

If you were a taxi company owner would you use this web application? *

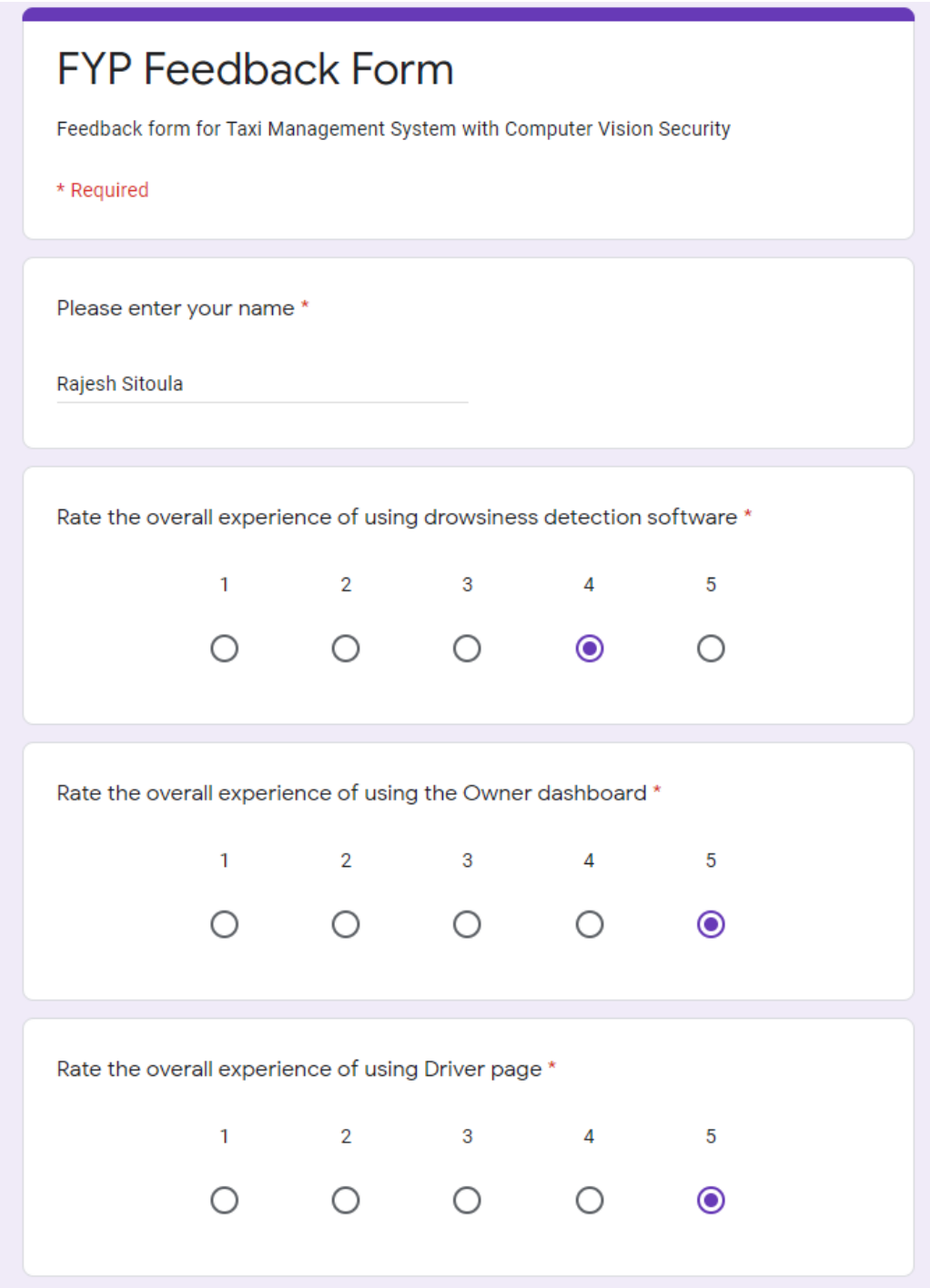
- Yes
- No

Do you have any suggestions for improving the application

Your answer

Figure 201User Feedback form (2)

8.7.2 SAMPLE OF FILLED USER FEEDBACK FORMS



FYP Feedback Form

Feedback form for Taxi Management System with Computer Vision Security

* Required

Please enter your name *

Rajesh Sitoula

Rate the overall experience of using drowsiness detection software *

1 2 3 4 5

Rate the overall experience of using the Owner dashboard *

1 2 3 4 5

Rate the overall experience of using Driver page *

1 2 3 4 5

Figure 202 Filled User Feedback form (1)

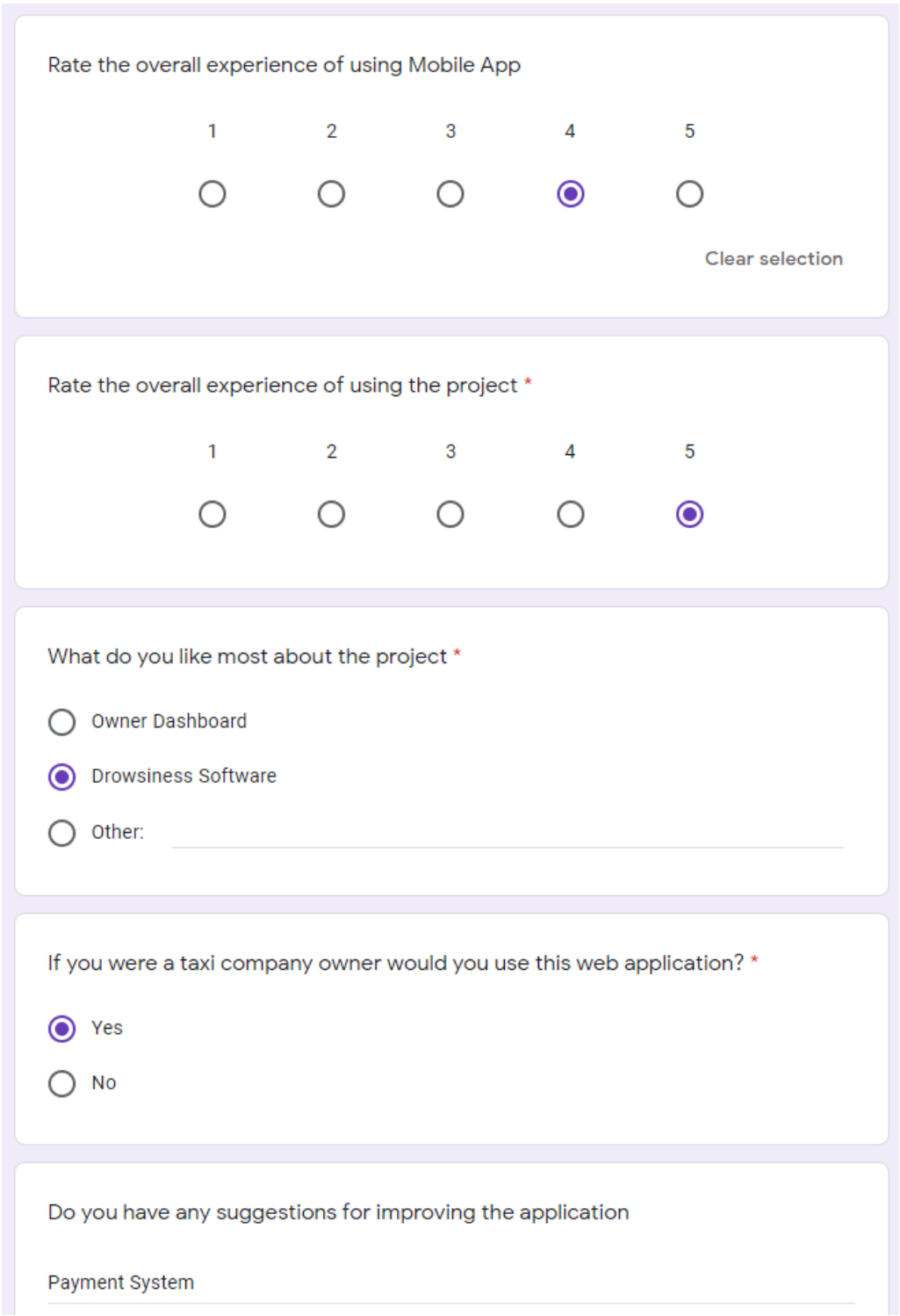


Figure 203 Filled User Feedback form (2)

8.7.3 WEB APPLICATION MANUAL (USER)

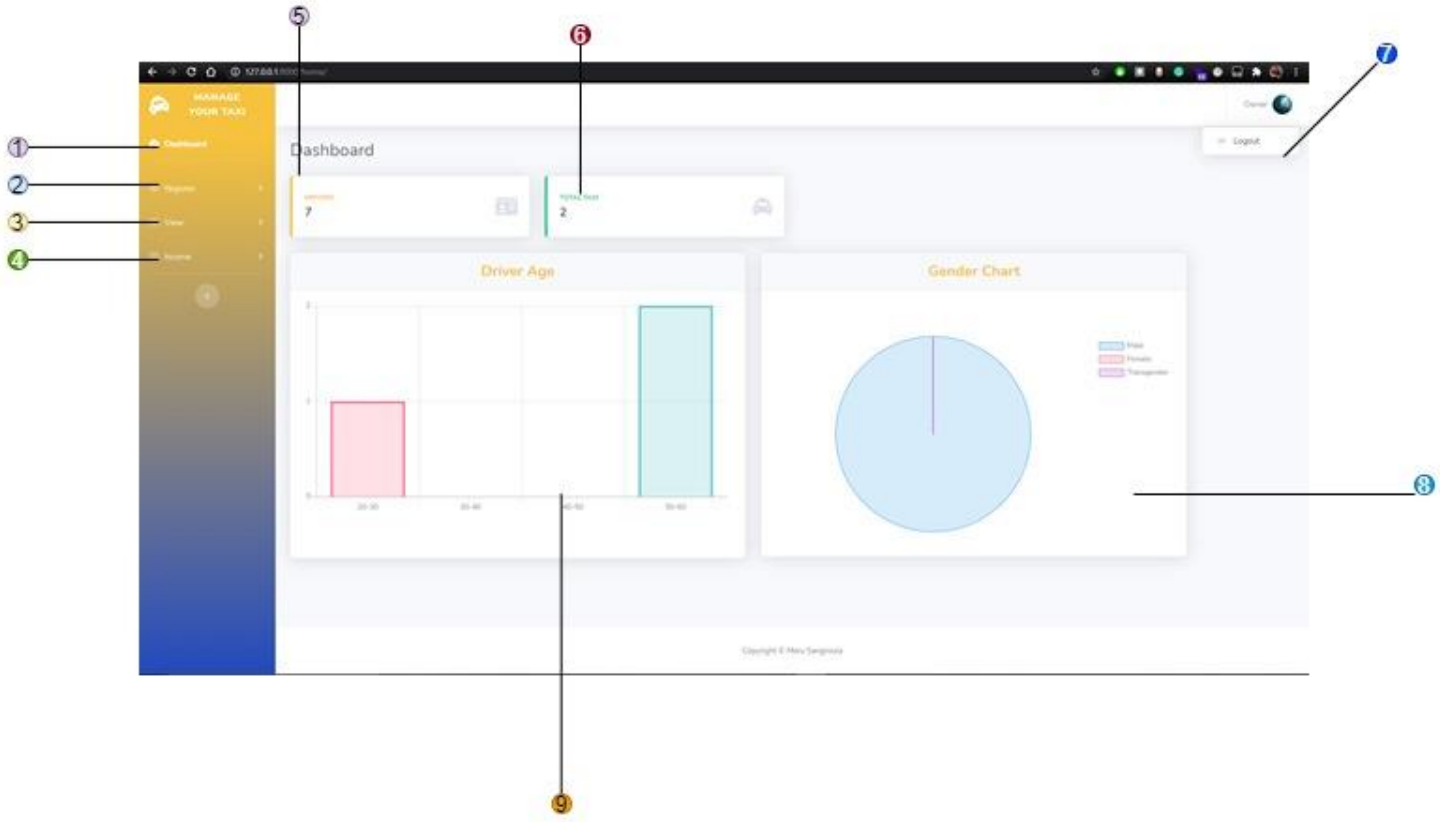


Figure 204 Web application Manual

The above figure is the user manual. This manual will give the total information about the web application. This manual will make easier of the end user to understand the design.

No. 1 is the dashboard. This page appears when authenticated user is logged in. This page gives the overall information about the business and gives user to user other features as well.

No. 2 is the register option where there are 2 registration options. One is Taxi and the other is driver. In this page, user can fill the forms to register driver and taxi as well.

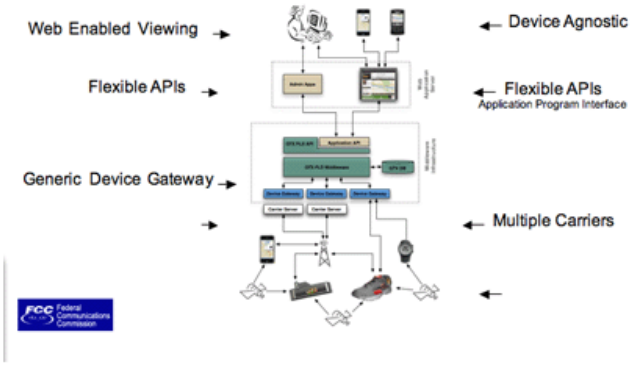
No. 3 is the view option. In this option, user can either view total taxi or driver and has access to update and delete both taxi and driver.

No. 4 is the income option. The owner here can either add daily income can view his total company incomes. User also has access to do CRUD operations for income. No 5 and No. 6 card shows the total number of driver and taxi the company has. No. 7 is the logout options where owner can logout from the application. No. 8 and No. 9 shows different types of chart. One shows the gender chart of all the drivers and the other shows the age graph of all the drivers.

8.8 APPENDIX G: FUTURE WORK

8.8.1 READING FOR THE FUTURE WORK

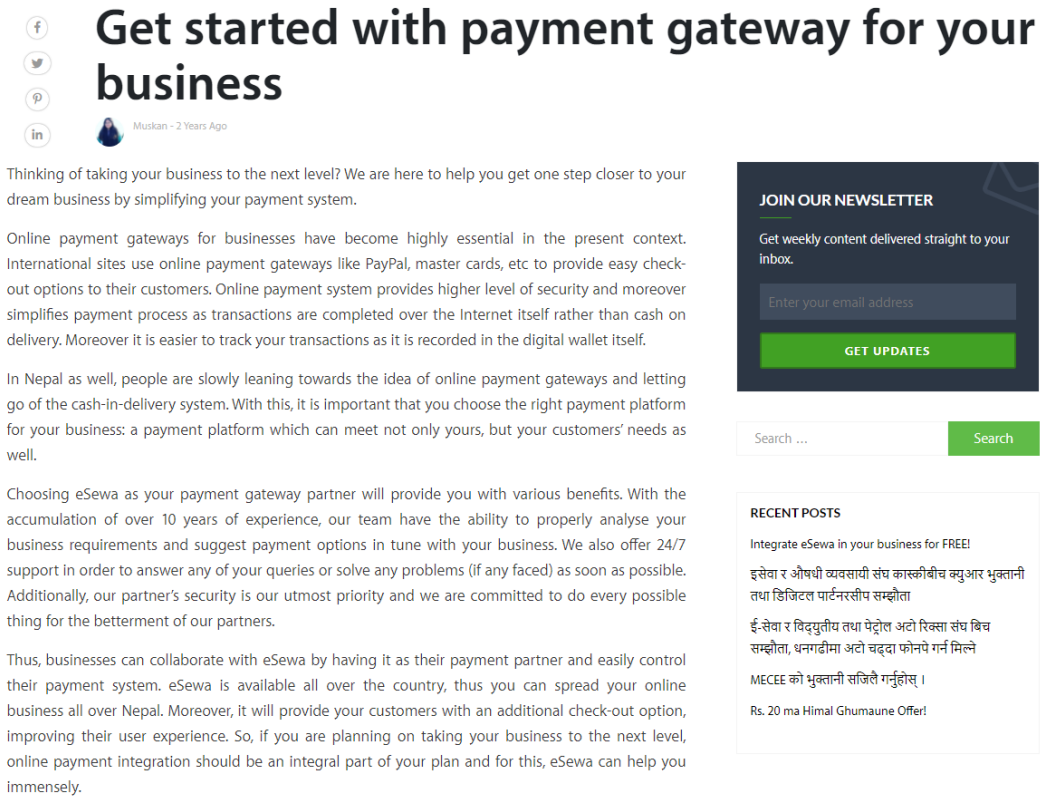
- Implementation of GPS Tracking



Real-time tracking is also particularly useful from a security perspective as it allows vehicle owners to pinpoint the exact location of a vehicle at any given time. And, the GPS tracking system in the vehicle may then be able to help police work out where the vehicle was taken to if it was stolen.

Figure 205 GPS Tracking System

- Payment Gateway



Get started with payment gateway for your business

Muskan - 2 Years Ago

Thinking of taking your business to the next level? We are here to help you get one step closer to your dream business by simplifying your payment system.

Online payment gateways for businesses have become highly essential in the present context. International sites use online payment gateways like PayPal, master cards, etc to provide easy check-out options to their customers. Online payment system provides higher level of security and moreover simplifies payment process as transactions are completed over the Internet itself rather than cash on delivery. Moreover it is easier to track your transactions as it is recorded in the digital wallet itself.

In Nepal as well, people are slowly leaning towards the idea of online payment gateways and letting go of the cash-in-delivery system. With this, it is important that you choose the right payment platform for your business: a payment platform which can meet not only yours, but your customers' needs as well.

Choosing eSewa as your payment gateway partner will provide you with various benefits. With the accumulation of over 10 years of experience, our team have the ability to properly analyse your business requirements and suggest payment options in tune with your business. We also offer 24/7 support in order to answer any of your queries or solve any problems (if any faced) as soon as possible. Additionally, our partner's security is our utmost priority and we are committed to do every possible thing for the betterment of our partners.

Thus, businesses can collaborate with eSewa by having it as their payment partner and easily control their payment system. eSewa is available all over the country, thus you can spread your online business all over Nepal. Moreover, it will provide your customers with an additional check-out option, improving their user experience. So, if you are planning on taking your business to the next level, online payment integration should be an integral part of your plan and for this, eSewa can help you immensely.

JOIN OUR NEWSLETTER

Get weekly content delivered straight to your inbox.

Enter your email address

GET UPDATES

Search ... Search

RECENT POSTS

Integrate eSewa in your business for FREE!

इसेवा र औषधी व्यवसायी संघ कास्कीबीच क्युआर भुक्तानी तथा डिजिटल पार्टनरसीप सम्झौता

ई-सेवा र विद्युतीय तथा पेट्रोल अटो रिक्सा संघ बिच सम्झौता, धनगढीमा अटो चढ्दा फोनपे गर्न मिल्ने

MECEE को भुक्तानी सजिलै गर्नुहोस् !

Rs. 20 ma Himal Ghumaune Offer!

Figure 206 Payment Gateway