CANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report
Version 2

CENG 408
Innovative System Design and Development II

An SaaS Platform for Ridesharing, Taxi Cab, Food Delivery and Shipping

---

Onur Dündar YALDIR
201411066
Onur Ata Sarıtaş
201511049
Alper Odaman
201511042

An SaaS Platform for Ridesharing, Taxi Cab, Food Delivery and Shipping 1
Abstract 3
1. Introduction 4
1.1 Problem and Solution 4
1.2 Motivations 4
2. Literature Review 4
2.1 Background 4
2.2 Related Literature 5
3. Summary 6
3.1 Final Thoughts 6
4.Software Requirement Specifications 6
List of Figures 6
4.2 Overall Description 6
4.2.1 System Environment 6
4.2.2 Users 7
4.2.3 Operating Environment 7
4.3 External Interfaces 7
4.3.1 User Interfaces 7
4.3.2 Hardware Interfaces 7
4.3.3 Software Interfaces 7
4.4 System Features 8
4.4.1 Functional Requirements 8
4.4.1.1 Starting Menu 8
4.4.1.2 Main Menu 10
4.4.1.2.2 Flow Description 11
4.4.1.3 Driver Menu 12
1.1. Service Preferences 13
1.2. Set Service Radius 13
4.4.2 Performance Requirements 14
4.4.3 Security Requirements 14
4.4.4 Software Quality Attributes 14
4.4.4.1 Adaptability 14
4.4.4.2 Portability 14

4.3.3 Usability 15
5. Software Design Description 15
List of Figures 15
5.1. Introduction 15
5.1.1 Purpose 15
5.1.2 Scope of this Project 15
5.1.3 Glossary 15
5.2 Architecture Design 16
5.2.1 Class Diagram 16
5.2.2 Interface for Passengers 16
5.2.3 Interface for Drivers 17
5.2.4 Activity Diagram 19
5.3 Use Case Realizations 20
5.3.1 Components of Mobile Application 20
5.3.1.1 GUI Design 20
5.3.1.2 Client Design 20
5.3.1.3 System API Design 21
6. Test Plan 22
7. Conclusions 39
Appendix 40
References 46

Abstract

With the widespread popularity of smartphones, several applications regarding commuting inside cities and traveling has been introduced. In this paper, we have reviewed several ideas regarding ride sharing applications, its effects on environment, how it'd work with recent vehicle routing algorithms, its ease of use with mobile and online payment systems and its reliability as an SaaS platform.
KEY WORDS: Peer-to-peer ridesharing, software-as-a-service platform, mobile application development.

1. Introduction

1.1 Problem and Solution

Peer-to-peer ridesharing is a growing area of transportation that has gained popularity with the widespread usage of smartphones and online payment systems. After World War II, ridesharing idea started as "car-sharing clubs" in America, with government supports regarding workplace and regulations, alternative transportation and commuting methods surfaced. In early 1990's these alternative methods such as ridesharing didn't gain much popularity, caused by drivers' and passengers' communication problems, drivers' unsafe driving habits, and issues regarding payment. Recent technological advancements in

IT fields made these issues irrelevant, smartphones and online platforms are tools most people use almost constantly. Several companies has made use of this vacancy in transportation methods, and peer-to-peer ridesharing applications grew in popularity. These applications removes communication issues between drivers' and passengers', using cutting-edge algorithms for vehicle routing and passenger stops they provide optimized routes and solves disputes with their feedback systems and mobile payment options.

1.2 Motivations

As discussed in Literature Review, ridesharing is a growing area of transportation that has gained popularity with the widespread usage of smartphones and online payment systems. Several companies has made use of this vacancy in transportation methods. These applications removes communication issues between drivers' and passengers', using cutting-edge algorithms for vehicle routing and passenger stops they provide optimized routes and solves disputes with their feedback systems and mobile payment options.

2. Literature Review

2.1 Background

Ridesharing typically involves carpooling, which is gathering several travellers into private automobile. Most automobiles can carry at least four passengers, however the rate of car occupancy continues to decline [1]. Most recent data for average number of passengers per car for the countries sampled in a research done by European Environment Agency in 2016 is approximately 1.45 passengers per vehicle [2]. There are many benefits in ridesharing since it reduces the amount of vehicles needed by travellers such as reduced energy consumption, greenhouse gas emissions, parking space needed and traffic congestion. According to a case study done by Fox Networks Group for The Rideshare Company, ridesharing has been used to avoid 1,770,585 vehicle trips, 18,254,291 lbs. of CO2 emissions and 924,184 gallons of gas ended up being saved [3]. The sheer magnitude of these numbers show that ridesharing is an activity that can be used as an efficient transportation alternative. The strictness of common ridesharing engagements, creates a difficulty for people to engage in these activities. However advancements in information technologies allows passengers and drivers to gather round with no serious commitments and little premeditation. Lately ridesharing also involves more unique forms like peer-to-peer ridesharing or sometimes called dynamic ridesharing. Dynamic ridesharing presents people convenience and flexibility. Travelers can request rides minutes before their desired departure times while advances in GPS technology assists drivers in finding routes, which in return increase the convenience of both parties. However, bringing together ridesharing activities such as Food Delivery, Shipping and Taxi Cab in a single platform is a tall task, traffic congestion and general transportation methods should be considered. Traffic congestion manifests itself in cities mostly during and after morning commutes and rush hours in business areas. Increased traffic density causes escalated commute times, extra oil consumption, environmental pollution and many more unfavourable effects. Ridesharing is a means to address these risk factors, and is considered as an alternative to public transportation to traffic congestion[4][5]. More recently, with technological advances in GPS technology and innovational ride sharing initiatives such as Uber, Lyft and Carma; ridesharing market is growing steadily.

2.2 Related Literature

The online delivery of software, sometimes called Software As A Service; is a software distribution model where the hosting and management of an application is handled by a third-party provider. This distribution model removes the need for organizations to install and run applications on their own. Software As A Service (abv. SaaS) has become a common delivery model for many business applications, such as office software, management software, development software, accounting etc. The advantage for using third-party providers for these services, is reducing development times since there's no reinventing the wheel, developers use already existing solutions to their problems and implement them into the project. This process creates more reliable products, troubleshooting bugs and errors is less troublesome compared to traditional development methods. This model of delivery is being used for ridesharing applications such as Uber[6][7], using Google Maps API and Mapbox API for geolocation and payment options such as PayPal, Google Wallet and Apple Pay. Online payment is a controversial topic since most people aren't very trusting about electronic payment schemes. However, today, online payment is the fastest and most reliable way of paying for services and products since the transaction happens virtually, there is less chance of fraud, realistically there is no human error involved. SaaS platforms uses third-party providers for transactions, making the entire platform secure by design. These transactions' fast nature is generally combined with mobile payments. Payments in mobile platforms can be done with smartphones, tablets, or generally Near Field Communication (abv. NFC) devices. These mobile devices is used to authorize and exchange money for services on the go, making them essentially the best choice for ridesharing applications. The wide usage of smartphones and mobile payment gave peer-to-peer ridesharing popularity a boost. SaaS platforms offers mobile applications quicker deployment times for new features and easy updates for existing ones. There are substantial amount of benefits using SaaS platforms. For developers, using SaaS platforms reduces risks in security; third party components can provide reliable security for developers not well-versed in handling sensitive data. Third party components can also assist teams in staying up-to-date.

One of the most important systems in mobile applications is the feedback system. In traditional web and mobile applications, feedback systems can be grouped into 2 main groups. These are direct and indirect feedback systems. Direct feedback systems ask the user directly about the application with their opinions and experiences with question surveys or pop-ups. Indirect feedback systems aim at creating meaningful feedback by processing the user's history of use on the system, the frequency of clicks in the system. The aim of feedback systems is to identify product holders, designers and developers with problems, shortcomings and situations in which the end user is not satisfied. The feedback system allows the system or application to be a more successful product with feedback from users.

3. Summary

3.1 Final Thoughts

Nowadays, the use of some software services has become popular in order to facilitate people's daily lives or to gain extra economic benefits. One of them, the ridesharing system, provides safe and more economical journeys without time constraints. Considering the problems of some of ridesharing schemes used in the 1990s, the modern-day versions of ridesharing has been developed from the ground up by using innovative technologies. However, there are some technologies that people hesitate to use, like mobile payment.

However, after realizing the reliability of these technologies, there is no doubt that ridesharing will become much more advanced than where it is now and that it will be used not only in developed countries but in many places around the world. One of the factors that make ridesharing safe is the evaluation of passengers and drivers with feedbacks and rating system. In this way, both drivers and passengers can make their choices by seeing the average score.

4.Software Requirement Specifications

List of Figures

- Figure 1: Menu Use Case Diagram
- Figure 2: Main Menu Use Case Diagram
- Figure 3: Driver Use Case Diagram

4.1 Introduction
4.1.1 Purpose

The purpose of this document is to describe the project called An Saas Platform for Ridesharing Taxi Cab, Food Delivery and Shipping. The project aims to reduce travel time and traffic problem via mobile platform. The mobile platform uses the feedback system to rank passengers and drivers, reduce risk factors and provide a safer and cheaper way to travel. This document contains detailed information about the requirements of the project. Reflect the defined restrictions and recommended software functions. The SRS document also describes how drivers and passengers interact.

4.2 Overall Description
4.2.1 System Environment

This application is a dynamic ridesharing and carpooling application. The user which can be passenger, driver or both. Passenger calls a driver which passenger can select and driver accept passenger for a drive. Users can give feedback to each other users. For drivers the system should confirm the driver’s licence.

4.2.2 Users

4.3.2.1 Passengers

- Passengers must have a smartphone.

- Passengers must know how to use applications.
- Passengers must know how to use wallet systems.
- Passengers must know at English at least initial level.

4.2.2.2 Drivers

- Drivers must have a driver license.
- Drivers must have a smartphone.
- Drivers must know how to use applications.
- Drivers must know how to use wallet systems.
- Drivers must be never get a traffic ticket for drink and drive.
- Drivers must never suspended from system before.

4.2.3 Operating Environment

This system will be a web application. So that it can be used every browser. Only internet connection and an updated (for performance) browser needed.On the database side, a database system will be used to design and implement the necessary entities, tables and relations. . Web applications are flexible and has proven performance. Only with a front-end development it can be a mobile application. End of the project system will be a mobile application that is based on web application.

4.3 External Interfaces

4.3.1 User Interfaces

- A smart phone or a browser include menus like call drive, my wallet, login, logout, sign up also drivers can navigate these menus but drivers will have extra operations

4.3.2 Hardware Interfaces

- This project requires a server for server-side operations, databases etc

4.3.3 Software Interfaces

- This project requires a internet connection for APIs' and external map APIs, and a database connection for read/write data.

4.4 System Features

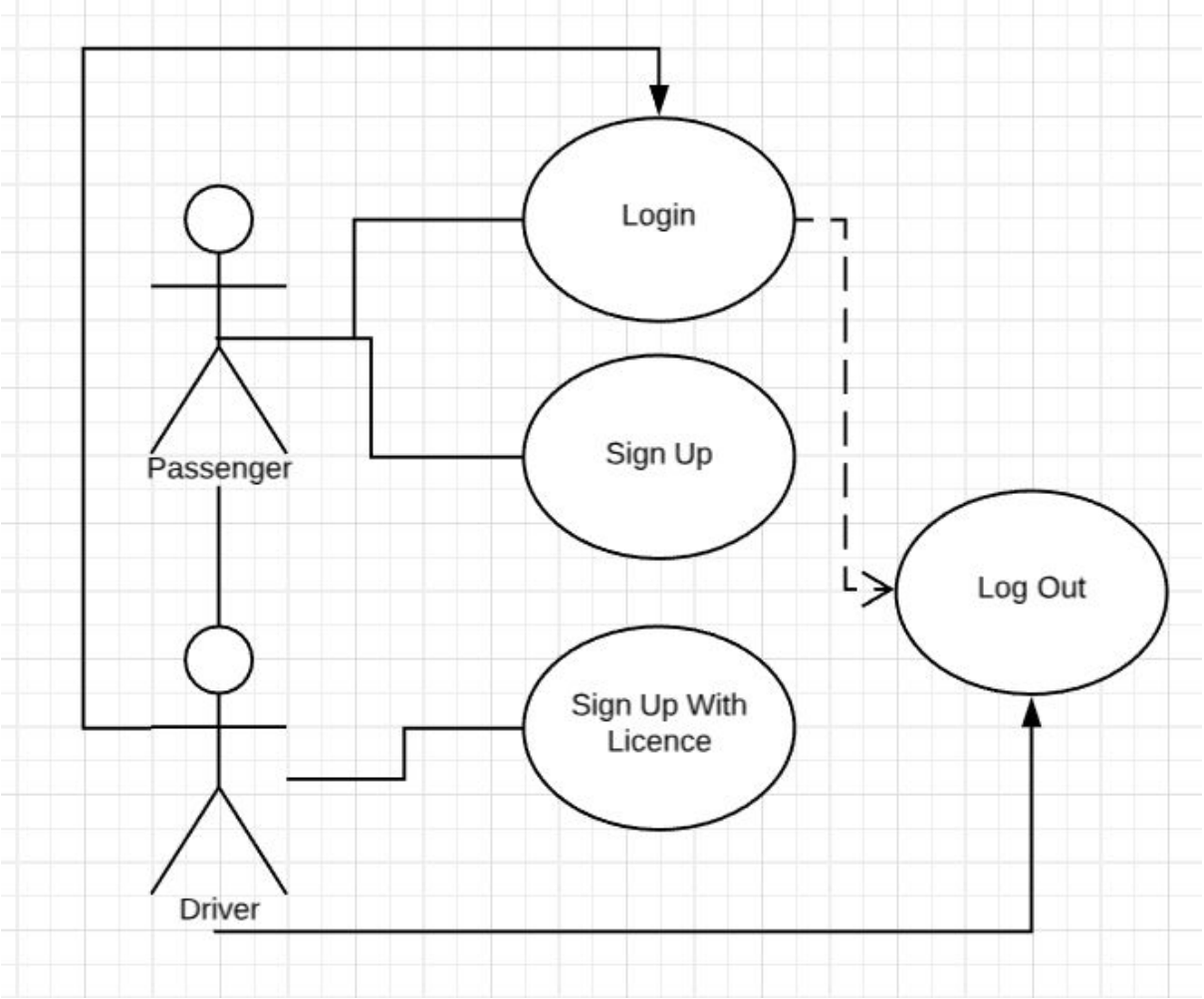
4.4.1 Functional Requirements

4.4.1.1 Starting Menu

- Login
- Sign Up
- Sign Up with License
- Log Out

4.1.1.1.1 Description

- Sign Up : User must sign up before login
- Sign Up With Licence : Users must Sign up With Licence for being a driver.
- Login : User must login before perform any operations.
- Log out : User must log out after no need to use this application.



4.1.1.1.2 Flow Description

1. Sign Up

1.1. User must sign Up with valid informations

1.1.1. If informations are invalid, error message is displayed.

2. Sign Up With Licence

2.1. User must sign up with Licence and valid informations if want to be a driver.

2.1.1. If informations are invalid, error message is displayed

2.2. The confirmation waiting from system for being a driver.

3. Login

3.1. After sign up user login the system using credentials which created in sign up step

3.1.1. If credentials are invalid, error message is displayed.

3.2. User redirected to main menu.

4. Log out

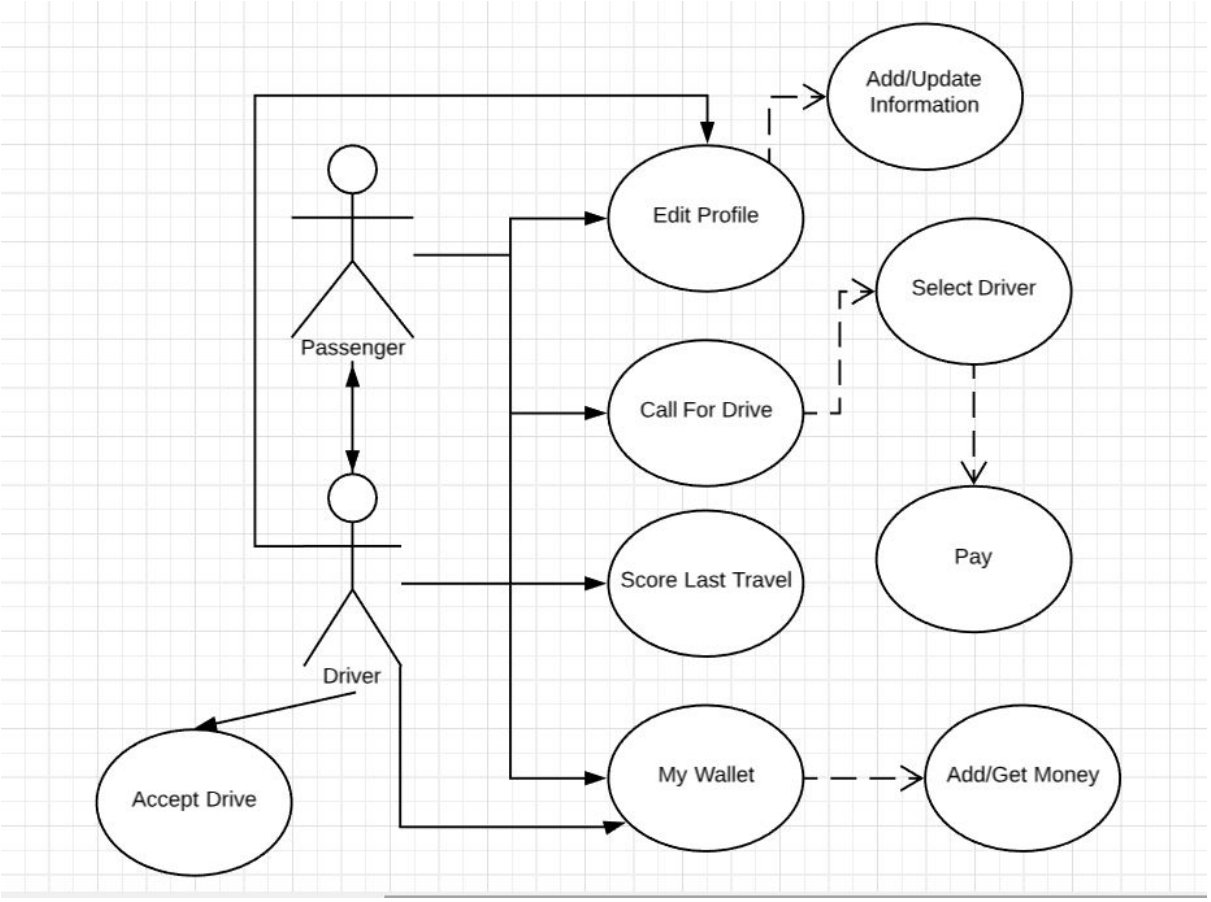
4.1. After using application. User log out from system

4.4.1.2 Main Menu

- Edit Profile
- Call for Drive
- Score Last Travel
- My Wallet
- Accept Drive

4.4.1.2.1 Description

- Edit Profile: Both passengers and drivers should be able to edit their profile.
- Call For Driver: Passengers call drive and select their drivers after that,they are making payment.
- Score Last Travel: Both passengers and drivers should be able to give feedback for their last travel.
- My wallet: Both passengers and drivers should be able to manage their wallet.
- Accept Drive: Only confirmed drivers can accept drives.



4.4.1.2.2 Flow Description

4.4.1.2.2.1 Passenger Side

1. My wallet

1.1. Passenger goes the My wallet section and add money first.

1.1.1. If there is an error. Message is displayed.

2. Edit Profile

2.1. Passenger goes Edit Profile Menu and Add Informations.

2.1.1. If there is an error. Message is displayed.

3. Call For a Drive

3.1. Passenger calls for a drive.

3.1.1. If there is an error. Message is displayed.

3.1.2. Passenger select the driver.

3.1.2.1. If there is an error. Message is displayed.

3.1.3. Passenger pay the payment.

3.1.3.1. If there is an error. Message is displayed.

4. Score Last Travel

4.1. Passenger goes score last travel section and gives feedback for last travel.

4.1.1. If there is an error. Message is displayed.

4.4.1.2.2.2 Driver Side

1. Edit Profile

1.1. Driver adds informations to profile.

1.1.1. If there is an error. Message is displayed.

2. Accept Drive

2.1. Driver accepts the drive

2.1.1. If there is an error. Message is displayed.

2.2. Driver provides the service.

3. My Wallet

3.1. Driver goes my wallet section and check money.

3.1.1. If there is an error. Message is displayed.

4.4.1.3 Driver Menu

- Preferences
- Service Preferences
- Drive History
- Give Feedback
- Current Travel
- Notifications

4.4.1.3.1 Description

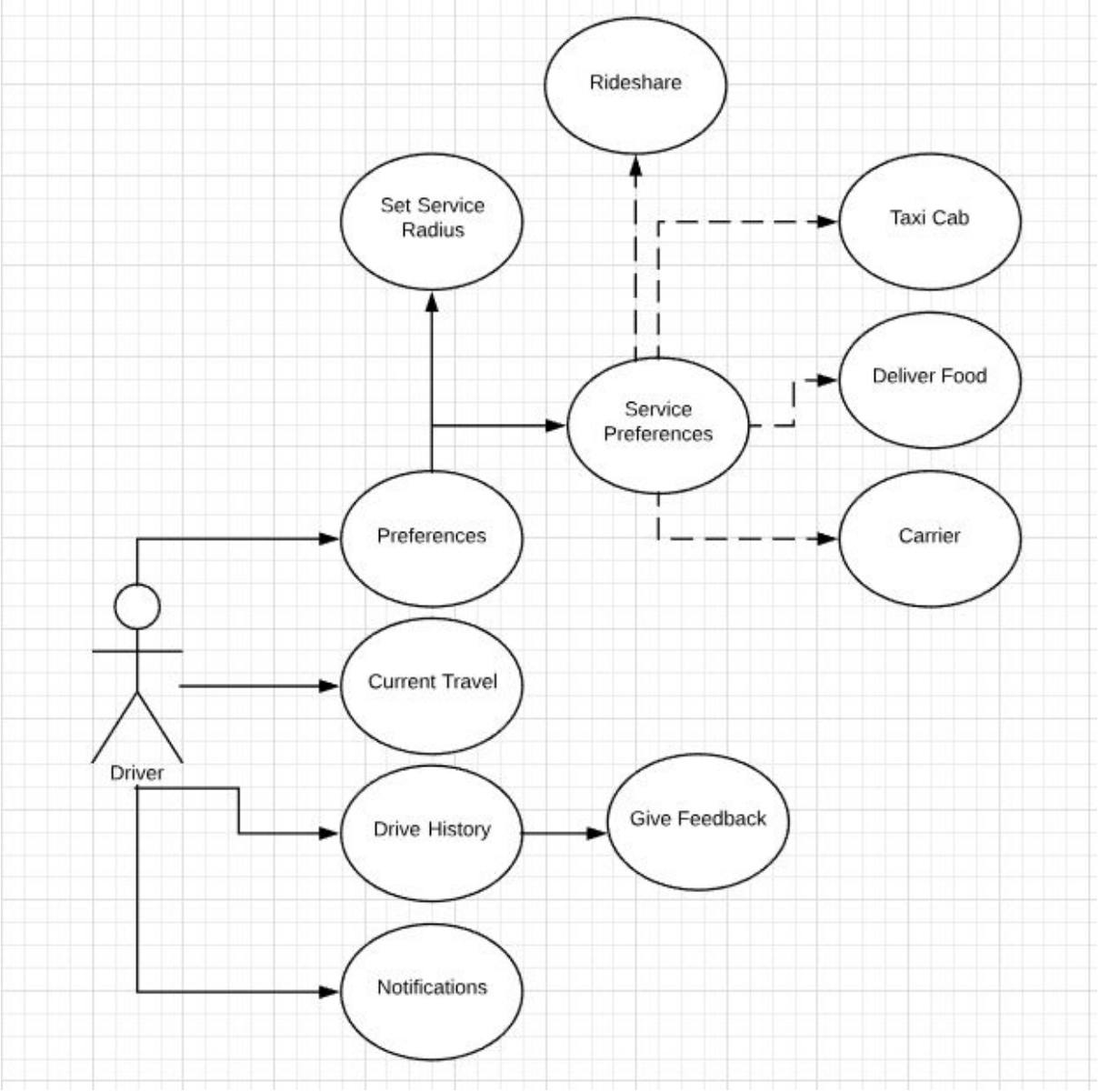
Preferences: Drivers can be able to select in this screen, which services they will provide.

Drive History: Drivers can be able to list old drives from this screen and give feedback

Current Travel: Drivers can be able to access information about current travel like map, billing, navigation etc

Notifications: Drivers can be able to access old and new notifications like calling from any

passenger, feedback notifications etc…



4.4.1.3.2 Flow Description

1. Preferences

1.1. Service Preferences

1.1.1. Set which services will be provide

1.1.1.1. If there is an error, message is displayed.

1.2. Set Service Radius

1.2.1. If there is an error, message is displayed.

2. Notifications

2.1. Check notifications regularly for if new passenger.

2.1.1. If there is an error, message is displayed.

2.2. Accept Passenger.

2.2.1. If there is an error, message is displayed.

3. Current Travel

3.1. Get the passenger,food or goods to transport.

3.1.1. If there is an error, message is displayed.

4. Drive History

4.1. List drives historically and get the information about.

4.1.1. If there is an error, message is displayed.

4.2. If feedbacks are missing, give feedback.

4.2.1. If there is an error, message is displayed.

4.4.2 Performance Requirements

The performance of connection between client and server must have high quality without any latency to avoid bad calls for drivers or passengers. Also API's response speed which are finding routes, meeting drivers and passengers has importance. Transition between menus must be smooth and fast. Authentication module must be fast and reliable.

4.4.3 Security Requirements

The feedback & scoring system must be processed attentively for both passengers and drivers security. Also payment system must be a reliable and proven payment system.

4.4.4 Software Quality Attributes

4.4.4.1 Adaptability

- This project is also Software as a Service. Everyone can use this platform with specified roles.

4.4.4.2 Portability

- This project aims at smart phones.

- The project will be a web app, so it can be used every browser not just smartphones.

4.3.3 Usability

- There are 2 roles in this project.

1. Passengers
2. Drivers.

5. Software Design Description

List of Figures

Figure 1: Main Page Interface for Passenger Figure 2: Class Diagram Figure 3: Main Interface Flowchart for Drivers Figure 4: Activity Diagram for the Application Figure 5: Use Case Realizations of the System

5.1. Introduction

5.1.1 Purpose

The purpose of this document is to describe the implementation of the “Riders on the Storm: An SaaS Platform for Ridesharing, Taxi Cab, Food Delivery and Shipping” application described in the Project Proposal Document. The application is designed to be used in ridesharing, food delivery, shipping and taxi cab activities.

5.1.2 Scope of this Project

This document describes the implementation details of the Riders on the Storm application. The application will consist of 2 major parts. First part will be the application interface for mobile devices, this interface will consist of two parts, one for passengers, one for drivers. Second part is the server API for these interfaces mentioned above to communicate and coupling. This diagram also consists of Flow Chart Diagram, Class Diagram and Use Case Diagrams. Every figure serves the purpose of explaining design details.

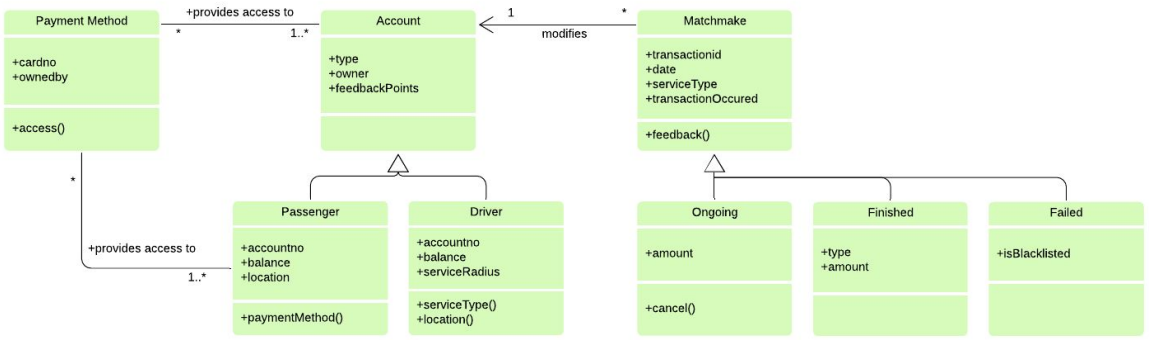
5.1.3 Glossary

Terms	Definitions
API	A set of subroutine definitions, communication protocols, and tools for building software. [19]
SaaS (Software-as-a-Service) Platform	A software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. [20]

Peer-to-peer Ridesharing	A service that arranges one-time shared rides on very short notice. [21]
Mobile Application	A computer program or software application designed to run on a mobile device such as a phone/tablet or watch. [22]
React Framework	A JavaScript library for building user interfaces.
React Native	A framework for building native apps using React Framework.

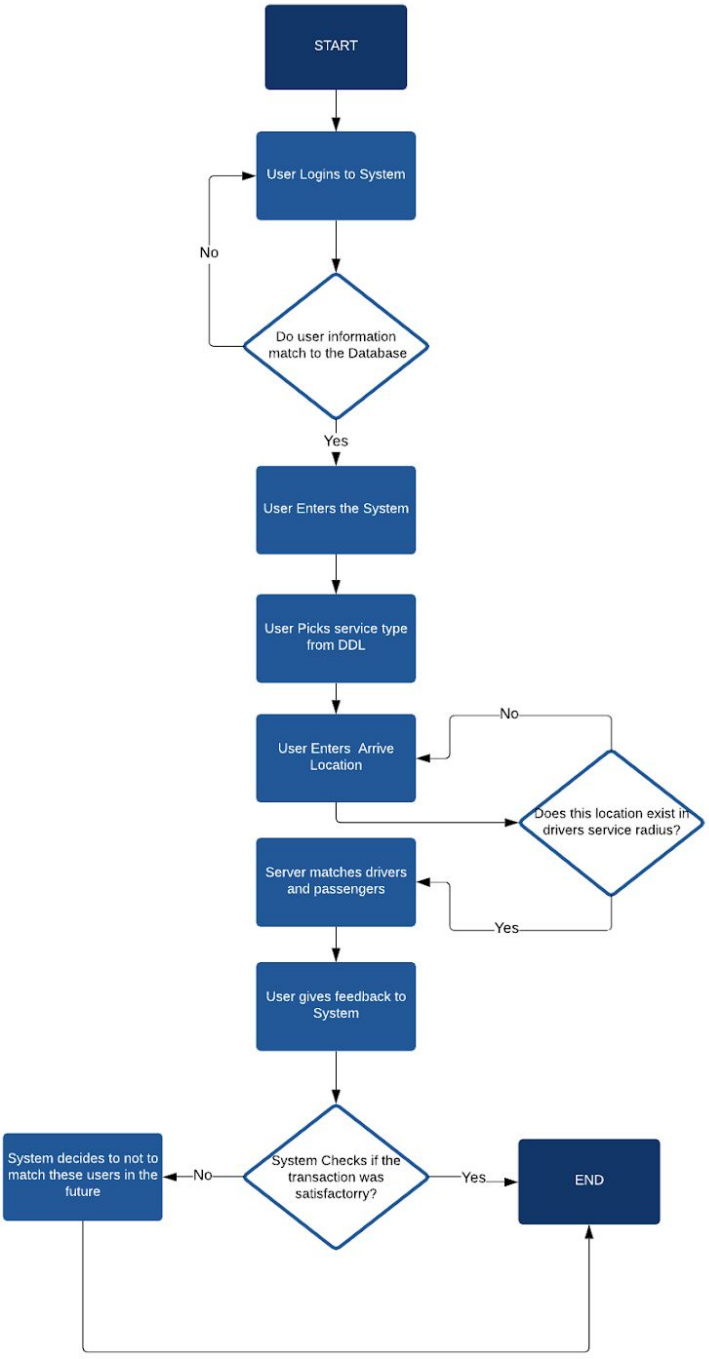
5.2 Architecture Design

5.2.1 Class Diagram



5.2.2 Interface for Passengers

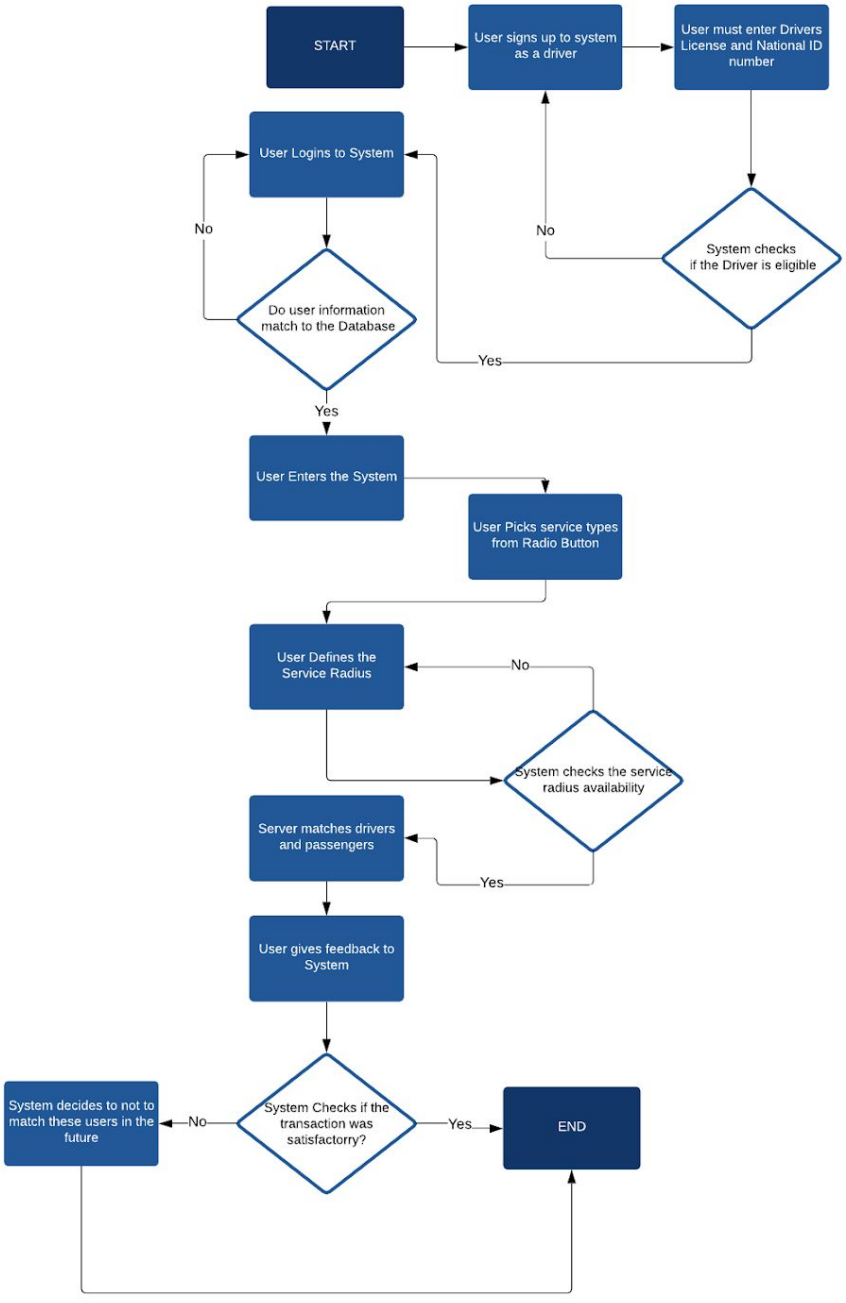
Description: This main page is the logged in passenger's home page. There is a local map, a button and there are two hyperlinks in this page. One of the hyperlinks connects menu page which has four hyperlinks and personal average feedback score which is given by drivers. First hyperlink brings passenger's past travels. Second hyperlink connects to the help page. Third hyperlinks connects to passenger's payment page. And the last hyperlink brings the settings page. The second hyperlink of main page connects to WHERE page. This page has one label and one hyperlink. The label brings other saved addresses. The hyperlink is to redirect client side and passenger select a location from local map. In the client side if passenger slip the local map a button comes up. This button shows the passenger's current location.



5.2.3 Interface for Drivers

Description: This main page is logged in driver's home page. There are 4 hyperlinks for notifications for new passengers to accept and feedback notifications, a preferences hyperlink redirects a 2 new hyperlinks that redirects service preferences and set service status.Other hyperlinks in mainpage are drive history which is list old drives chronological

and Current Travel hyperlink shows information about live travel session.

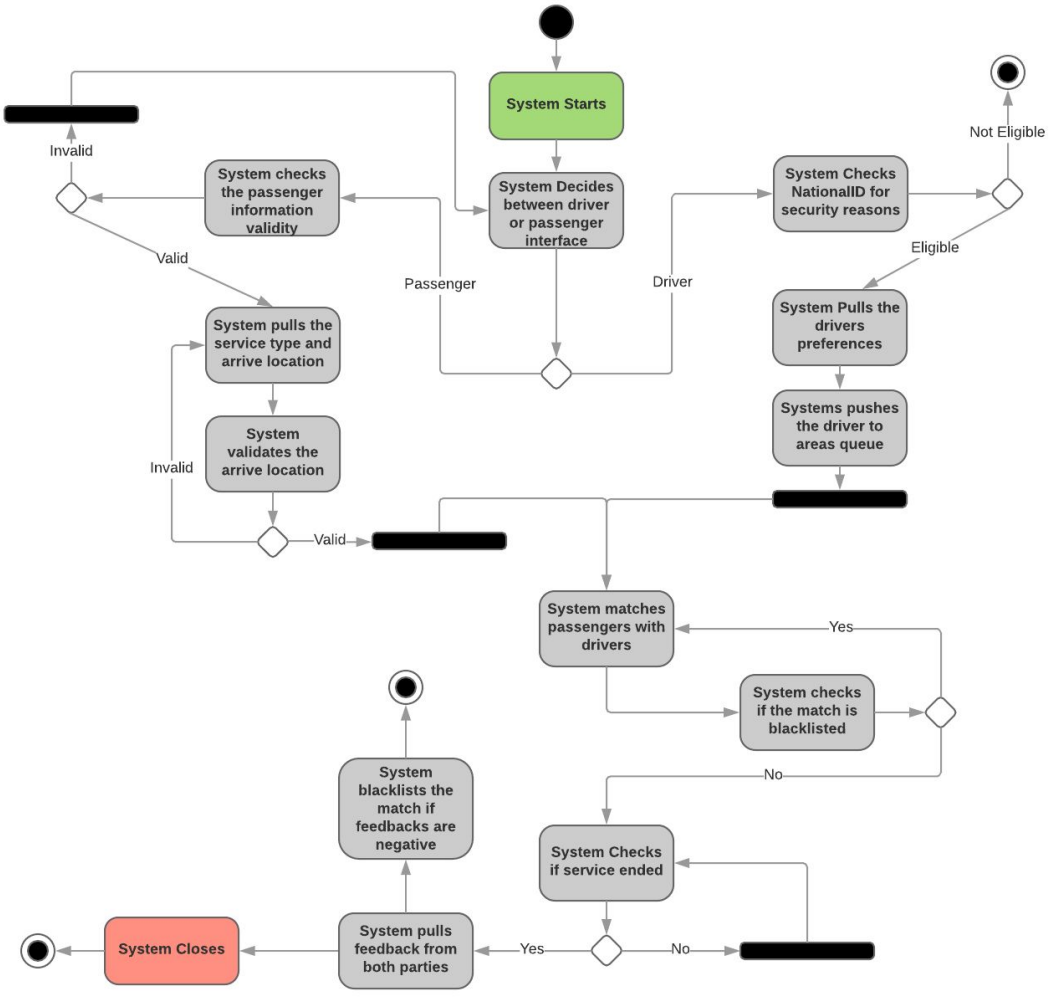


5.2.4 Activity Diagram

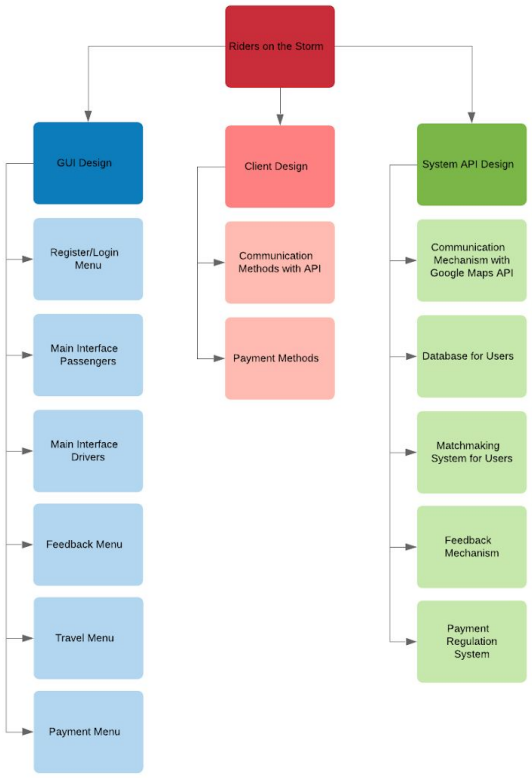
Activity Diagram describes the dynamic aspects of the system. First, Users should register to the system, giving their personal information. There are two options for registering, they can be drivers or passengers. After passenger is selected, the system should check the passenger information validity, if they are signed up before or not. After driver is selected, the system requires National Identification Number to check for traffic tickets for drinking and driving. After the security check the system pulls up the Main Interface respectively.

In Main Interface, Drivers should specify their preferences for their requested service types and service radius. The system pushes the driver to a queue and the driver is ready to go.

In Passenger side, Users picks their service type and arrive locations. After the system validates the arrive location for errors, the System matches drivers and passengers for the current process. Checks if the match is blacklisted, if not, match occurs. After the transaction happens, the system takes feedback from both parties, and decide to blacklist the match or not.



5.3 Use Case Realizations



5.3.1 Components of Mobile Application

In the above Figure, the components of the application is demonstrated. The system consist of three major components, GUI Design, Client Design and System API Design.

5.3.1.1 GUI Design

The GUI Design focuses on making an interface with a focus on looks. It provides an interaction between the user and application. It is composed of Register/Login Menu, Interfaces for botw Passengers and Drivers, Feedback Menu, Travel Menu and Payment Menu. The User will start the application in Register/Login Menu, and continue onwards to the Main interfaces, and after transactions, Travel, Feedback and Payment Menus.

5.3.1.2 Client Design

The Client Design comprises of Communication and Payment methods. The

Application should be able to communicate with the system's API, to handle location, match and feedback methods. Payment methods can be handled by third parties such as PayPal or Google Wallet.

5.3.1.3 System API Design

The System API Design is the core of the application. The system will handle all location and matchmaking systems, feedbacks and payment regulations. Also the system should handle the Users information in a database.

6.Test Plan

6.1 Introduction

6.1.1. Overview

In this part, components of application, which can be divided into 2 part; server and api
will be tested.

6.2.2 Scope

This document includes test plans, cases and test results and specifications about test development.

6.2.3 Terminology

SERVER	Server Side
GUI	User Interface Side

6.2 Test Cases

6.2.1. GUI Test Cases

S1: GUI

1. Test Cases

1.1. Register

ID	Title
C1	Register Button
C2	Register Function
C5	Cancel Button

1.4. Ride

ID	Title
C16	Passenger Ride Button
C17	Driver Ride Button
C18	Passenger Ride Function
C19	Driver Ride Function
C20	Map Location Reliability

1.5. Profile

ID	Title
C21	Profile Button
C22	Profile Function
C23	Logout Button
C24	Logout Function

6.2.2 Server Test Cases

S2: Server

1. API

ID	Title
C8	API Valid Register Call
C13	API Invalid Register Call
C15	API Duplicate Register Call
C9	API Valid Login Call
C14	API Invalid Login Call
C10	API Request Ride Call
C11	API Passenger Info Call
C12	API Delete User Call
C25	API Finish Ride Call
C26	API Find Driver Call
C27	API Get Profile Call

6.3 Test Cases Descriptions

6.3.1. GUI Test Cases Descriptions

1. Test Cases

1.1. Register

C1: Register Button

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

The application should be installed to mobile device

Steps

	Step	Expected Result
1	Clicked Application	Application started succesfully
2	Clicked Register button	Redirected to Register succesfully

C2: Register Function

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

GUI.C1

Steps

	Step	Expected Result
1	Filled all Blanks and Clicked Register Button	Registered with no errors and redirected Login page

C5: Cancel Button

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Clicked Cancel Button at Register Screen	Returned Login screen
2	Clicked Register button at Login screen	Texboxes reset

1.2. Login

C4: Login Function

Type	Priority	Estimate	References
Functional	Critical	None	None
Automation Type			
None			

Preconditions

GUI.C2 if needed

Steps

	Step	Expected Result
1	Registered succesfully if needed	
2	Filled blanks with valid info	Logined Succesfully

1.3. Comment

C6: Comment Function

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Filled Comment Textbox	Placeholder dissapeared
2	Clicked Send Button	Comment Saved and linked related rider

1.4. Ride

C16: Passenger Ride Button

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Ride button Clicked	Button Clicked and Called Function

C17: Driver Ride Button

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Click Ride button	Ride button Clicked and called Function

C18: Passenger Ride Function

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Preconditions

C16

Steps

	Step	Expected Result
1	Function called with button	The Ride Created

C19: Driver Ride Function

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Clicked Ride button and called Function	Ride Accepted

C20: Map Location Reliability

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Opened Map Screen	The location of device was same with Map's pointer location.

1.5. Profile

C21: Profile Button

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Click Profile Button	Button clicked with no error

C22: Profile Function

Type	Priority	Estimate	References
Functional	Low	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Clicked Profile Button and redirected Profile screen	Active user's profile shown at page

C23: Logout Button

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Clicked Logout Button	Button Clicked succesfully

C24: Logout Function

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Steps

	Step	Expected Result
1	Click Logout Button	Active user Logouted successfully

6.3.2 Server Test Cases Descriptions

S2: Server

1. API

C8: API Valid Register Call

Type	Priority	Estimate	References
Functional	Critical	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	Opened Postman and Send a Post to [host]/users with a validjson data. Example:
	{
	"userType":"passenger",
	"name":"onurr",
	"surname":"yaldiir",
	"city":"ankara",
	"username":"clariononurr",
	"password":"1234567890",
	"phone":"0504586023",
	"email":"onrdy916@gmail.com"
	}
		Status 201 returned with a json response included a token. Example
		
		{
		"user": {
		"_id": "5ceffa657ef85529d41e5c41",
		"userType": "passenger",
		"name": "onurr",
		"surname": "yaldiir",
		"city": "ankara",
		"phone": "0504586023",
		"email": "onrdy916@gmail.com",
		"__v": 1
		},
		"token":
		"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1Y2VmZmE2NTdlZjg1NTI5ZDQxZTVjNDEiLCJpYXQiOjE1NTkyMzEwNzd9.VYsdJD1q25Emsr4sseeDgFiPJpRo7j5LJvRydjNTfSU"
		}

C13: API Invalid Register Call

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	Opened Postman and Send a Post to [host]/users with a invalid json data. Example:
	{
	"userType":"passenger",
	"name":"onurr",
	"surname":"yaldiir",
	"city":"ankara",
	"username":"clariononurr",
	"password":"1234567890",
	"phone":"0504586023",
	"email":"onrdy9"
	}
		Invalid mail error returned. Example
		errors": {
		"email": {
		"message": "Email is invalid",
		"name": "ValidatorError",
		"properties": {
		"message": "Email is invalid",
		"type": "user defined",
		"path": "email",
		"value": "onrdy9",
		"reason": {}
		},

C15: API Duplicate Register Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Server.Passenger.C8

Steps

	Step	Expected Result

1	Trying to Register with same json	Returned a mongodb error json. Example
		
		{
		"driver": true,
		"name": "MongoError",
		"index": 0,
		"code": 11000,
		"errmsg": "E11000 duplicate key error collection: api-database.users index: phone_1 dup key: { : \"0504586023\" }"
		}

C9: API Valid Login Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	Opened Postman and Send a Post to [host]/users/login with a validjson data. Example:
	{
	"password":"1234567890",
	"email":"onrdy96@gmail.com"
	}
		Returned user info and a token in a json. Example
		{
		"user": {
		"_id": "5ce6cb120bce3342e45ed995",
		"userType": "passenger",
		"name": "onur",
		"surname": "yaldir",
		"city": "ankara",
		"phone": "0504586022",
		"email": "onrdy96@gmail.com",
		"__v": 9
		},
		"token":
		"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1Y2U2Y2IxMjBiY2UzMzQyZTQ1ZWQ5OTUiLCJpYXQiOjE1NTkyMzE2OTF9.8hWFo6W5H4aqNdGltrQRNalYkPpIHL_bVGaIAgu2DC4"
		}

C14: API Invalid Login Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	Opened Postman and Send a Post to [host]/users/login with a invalid json data. Example:
	{
	"password":"12345678901",
	"email":"onrdy96@gmail.com"
	}
		Response Status 400 Bad Request

C10: API Request Ride Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	Called [host]l/users/ride_finish Post mehod with a json.
	Example :
	{
	"origin": {
	"lat": 39.921980,
	"long": 32.839100
	},
	"destination": {
	"lat": 39.909111,
	"long": 32.810600
	},
	"ongoing": "false",
	"finished": "false"
	}
		Result is Http 201.

C11: API Passenger Info Call

Type	Priority	Estimate	References
Functional	Medium	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result
1	Go Authanticaton tab at Postman, add Bearer token whit got at login	Token added

2	Send [host]/users/me a get Call	a json data about user. Example
		{
		"_id": "5ce6cb120bce3342e45ed995",
		"userType": "passenger",
		"name": "onur",
		"surname": "yaldir",
		"city": "ankara",
		"phone": "0504586022",
		"email": "onrdy96@gmail.com",
		"__v": 11
		}

C12: API Delete User Call

Type	Priority	Estimate	References
Functional	Low	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	delete call [host]/users/me with token	Response is :
		{
		"_id": "5ce6cb120bce3342e45ed995",
		"userType": "passenger",
		"name": "onur",
		"surname": "yaldir",
		"city": "ankara",
		"phone": "0504586022",
		"email": "onrdy96@gmail.com",
		"__v": 11
		}

2	get call [host]/users/me with same token	Response is :
		{
		"error": "Authentication error"
		}

C25: API Finish Ride Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result

1	call [host]/users/ride_finish with a json and token .
	Example:
	{
	"feedback": 7
	}
		Http response was 201

C26: API Find Driver Call

Type	Priority	Estimate	References
Functional	High	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result
1	Call a post method [host]/users/find	Http response is 200.

C27: API Get Profile Call

Type	Priority	Estimate	References
Functional	Low	None	None
Automation Type			
None			

Preconditions

Postman Tool

Steps

	Step	Expected Result
1	Get token from login json	Token succesfully copied

2	Paste token to authorization section and call [host]/users/me "get" method with token	The json data of profile returned. Example
		
		{
		"_id": "5cf3ae3387aba91704691c19",
		"name": "Onur",
		"surname": "Saritas",
		"email": "saritas@mail.com",
		"city": "ankara",
		"phone": "+905055053050",
		"userType": "driver",
		"__v": 2
		}

7. Test Results

7.1 Individual Results

7.1.1 GUI Results

R1: GUI

Created On	6/2/2019
Completed	Yes

Passed	Blocked	Untested	Retest	Failed
100% (14/14)	0% (0/14)	0% (0/14)	0% (0/14)	0% (0/14)

1. Test Cases

1.1. Register

ID	Title	Assigned To	Status
T1	Register Button		Passed
T2	Register Function		Passed
T4	Cancel Button		Passed

1.2. Login

ID	Title	Assigned To	Status
T3	Login Function		Passed

1.3. Comment

ID	Title	Assigned To	Status
T5	Comment Function		Passed

1.4. Ride

ID	Title	Assigned To	Status
T6	Passenger Ride Button		Passed
T7	Driver Ride Button		Passed
T8	Passenger Ride Function		Passed
T9	Driver Ride Function		Passed
T10	Map Location Reliability		Passed

1.5. Profile

ID	Title	Assigned To	Status
T11	Profile Button		Passed
T12	Profile Function		Passed
T13	Logout Button		Passed
T14	Logout Function		Passed

7.1.2 Server Test Results

R2: Server

Created On	6/2/2019
Completed	Yes
Completed On	6/2/2019

Passed	Blocked	Untested	Retest	Failed
100% (11/11)	0% (0/11)	0% (0/11)	0% (0/11)	0% (0/11)

1. API

ID	Title	Assigned To	Status
T16	API Valid Register Call		Passed
T21	API Invalid Register Call		Passed
T23	API Duplicate Register Call		Passed
T17	API Valid Login Call		Passed
T22	API Invalid Login Call		Passed
T18	API Request Ride Call		Passed
T19	API Passenger Info Call		Passed
T20	API Delete User Call		Passed
T24	API Finish Ride Call		Passed
T25	API Find Driver Call		Passed
T26	API Get Profile Call		Passed

7. Conclusion

This document includes broad information about our project, titled as "An SaaS Platform for Ridesharing, Taxi Cab, Food Delivery and Shipping". In our project, we aimed to provide easy-to-use, open source and up-to-date web based solutions to the widespread ridesharing activities, to improve the communication between ridesharers and increase the traffic flow in high-populated metropol areas. We use open source technologies on mobile devices and server-side applications. The purpose for selecting mobile devices as the communication tool is because the widespread usage of social media and smart devices, since smart devices are so common, we are living in an information and improvement driven era.

Our project uses real-time communication and real time gps information. We have researched a lot of technologies which other companies used for similar projects and tried to understand how it can be more efficient. We declared mistakes from past and we tried to not make these mistakes in our project.

In the light of the developmental analysis that we have made, we have concluded that the popular node.js runtime and React Native framework are best suited our needs, after we have planned a meeting with out stakeholders, our requirements and design documents were made, we decided our chosen frameworks are suitable. The general setback that we encountered was the lack of example. Ridesharing is a relatively new activity, so designing and prototyping was done by our part. However we got help from our advisor, and done extensive research on the subject, and we have managed to conclude the project to its finish. We believe we have developed a platform that is cost-effective, upgradeable, and easy-to-use.

Appendix A: Installation Guide

Introduction

Riders is an SaaS (Software-as-a-Service) platform for ridesharing, it features a cross-platform server under node.js runtime, and a client web application written with React.js that can be installed in browsers, and recent mobile platforms.

System Requirements for Server

OS

Windows 7 SP1 or higher.

Linux.

macOS (10.10 Yosemite or higher).

Unix-like systems other than Linux are not supported but might work.

Processor

64-bit architecture

Kernel version 3.10 or higher

A modern CPU is recommended for performance reasons.

System Requirements for Client

Android 4.0 or higher.

iOS 8 or higher.

A web browser compatible with Google's JavaScript V8 engine.

64-bit architecture.

Building the Server

Download or clone the repository from GitHub.

Extract the package using 7Zip, WinRAR or similar packaging tools.

Install node.js 10.16 LTS runtime for your respective operating system from the official site. (https://nodejs.org/)

Install MongoDB from the official site. (https://www.mongodb.com)

for Windows

Run the installer for node.js and MongoDB that you've downloaded before, and follow the installer's instructions.

Start MongoDB Database from its executable (.exe).

Open Command Prompt, with `cd` command, traverse to the directory which the downloaded folder is in.

Using Command Prompt, enter the following instructions:

```
npm install
npm run start
```

for Linux and Mac OS X

Run the installer for node.js and MongoDB that you've downloaded before, and follow the installer's instructions.

Start MongoDB.

Open terminal, and install xcode's command-line tools:

```
$ xcode-select –install
```

Open terminal, with `cd` command, traverse to the directory which the downloaded folder is in.

Using terminal, enter the following instructions:

```
npm install
npm run start
```

Building the Client

Download or clone the repository from GitHub.

Extract the package using 7Zip, WinRAR or similar packaging tools.

Install node.js 10.16 LTS runtime for your respective operating system from the official site. (https://nodejs.org/)

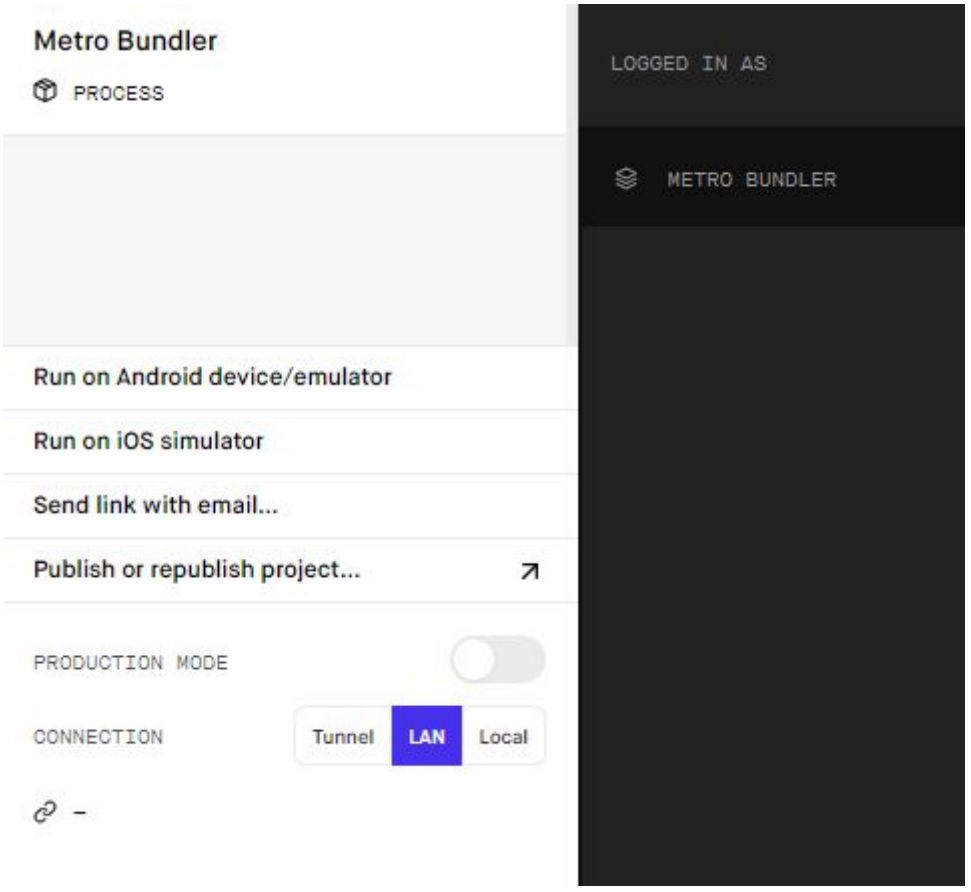
Using terminal or Command Prompt, traverse to the folder which the repository is currently in.

Install Expo Runtime with the following command:

```
npm install -g expo-cli
expo init
expo start
```

A browser page will open up. In the page, look for the following section:

Build the web application for your respective platform using the on-screen options



You can install the prebuilt binaries from the project's GitHub repo.

Appendix B: User Manual

System Requirements

for Server

Windows 7 SP1 or higher.

Linux.

macOS (10.10 Yosemite or higher).

Unix-like systems other than Linux are not supported but might work.

64-bit architecture

Kernel version 3.10 or higher

A modern CPU is recommended for performance reasons.

for Client

Android 4.0 or higher.

iOS 8 or higher.

A web browser compatible with Google's JavaScript V8 engine.

64-bit architecture.

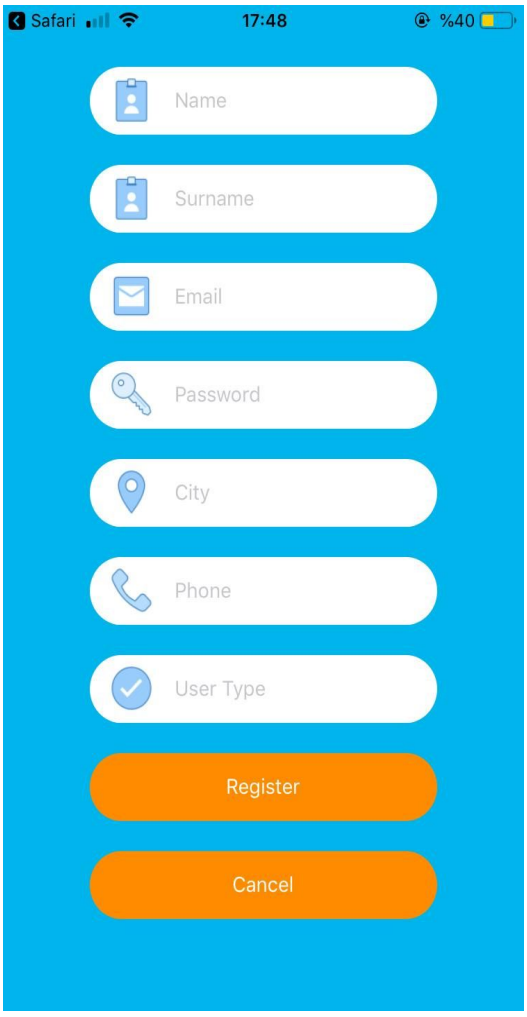
Overview of the Software

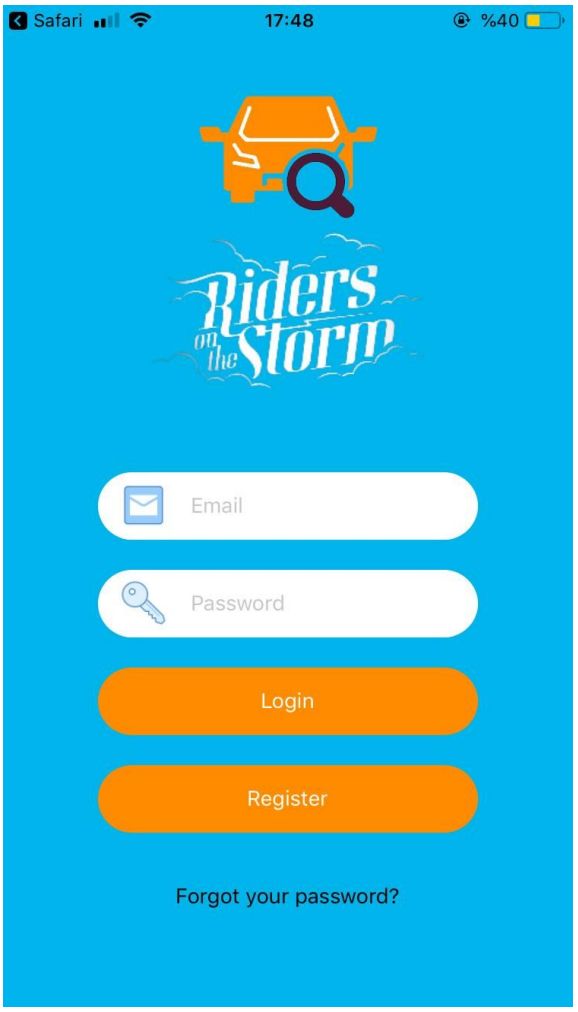
Riders, An SaaS Platform is a ridesharing platform that assists pedestrians and overall aims to get improve traffic flow in modern cities. This platform is used in daily transportation and commuting environments to increase communication between drivers and passengers to achieve efficient and safe ridesharing activities. By using modern and open source packages, the platform is upgradeable and easy-to-use.

The platform consists of two main parts, a server to manage transactions and match the users for ridesharing and transportation activities, and a client that runs on modern mobile devices like Android and iOS. Users should register and login to the system with their email accounts, and they can immediately start using the platform. The platform has two main interfaces, one for passengers, one for drivers. Drivers should identify themselves with their National Identification Number. Switch between these interfaces is not possible, an account can be used in only that interface.

User Walkthrough

Register and Login Screens

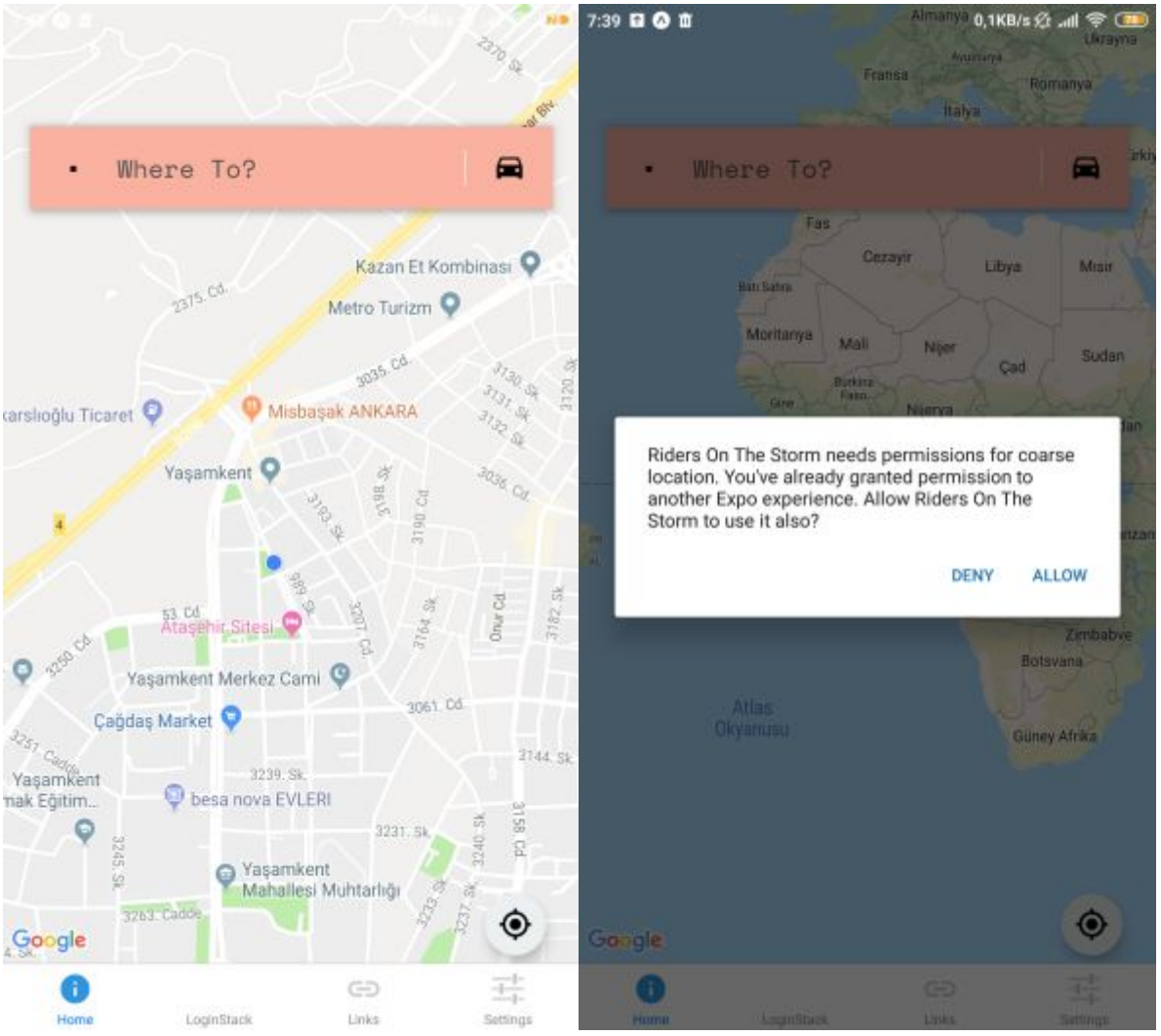




In Register Screen, users are expected to fill in the required information for their accounts, the fields are listed below, and can be seen in the figure. After the registration process, users are logged in directly.

In Login screen, existing users can log in to the system and start using the platform. Email and password are expected from user.

User Main Interface



After logging in, the users are presented with the below permission request, the current location is needed for the application to run, so the permission must be granted before continue or the application will be unstable.

Main interface of the program is as represented above. The users have a settings bar at the bottom of the screen, for different pages and requests. For the ride requests and ridesharing activities, the user must enter the destination location in the text bar in the upper middle of the screen.

References

- [1] Deakin, E., Frick, K. T., & Shively, K. (2012). Dynamic ridesharing.
- [2] Chan, N. D., & Shaheen, S. A. (2012). Ridesharing in north america: Past, present, and future. Transport Reviews, 32(1), 93-112.
- [3] Amey, A., Attanucci, J., & Mishalani, R. (2011). Real-time ridesharing: opportunities and challenges in using mobile phone technology to improve rideshare services. Transportation Research Record: Journal of the Transportation Research Board, (2217), 103-110.
- [4] Kelly, K.L., 2007. Casual carpooling-enhanced. J. Public Transp. 10, 119–130
- [5] Ferguson, E., 1997. The rise and fall of the American carpool: 1970–1990. Transportation 24, 349–376
- [6] Introducing a more flexible option to purchase the Google Maps Web Service APIs. (2018, November 01). Retrieved from https://mapsplatform.googleblog.com/2015/09/introducing-more-flexible-option-to.html
- [7] Uber Developers API (2018, November 01). Retrieved from https://developer.uber.com/docs/riders/ride-requests/tutorials/api/introduction
- [8] Qin, Z., Sun, J., Wahaballa, A., Zheng, W., Xiong, H., & Qin, Z. (2017). A secure and privacy-preserving mobile wallet with outsourced verification in cloud computing. Computer Standards & Interfaces, 54, 55-60.
- [9] Pernet-Lubrano, S. (2010). Mobile payments: moving towards a wallet in the cloud?.
- [11] Yang, Y., Liu, Y., Li, H., & Yu, B. (2015). Understanding perceived risks in mobile payment acceptance. Industrial Management & Data Systems, 115(2), 253-269.
- [12] Masoud, N., Nam, D., Yu, J., & Jayakrishnan, R. (2017). Promoting Peer-to-Peer Ridesharing Services as Transit System Feeders. Transportation Research Record: Journal of the Transportation Research Board, (2650), 74-83.
- [13] Bauer, D. (2017). Opportunities and barriers of ride-sharing in work commuting–a case study in Sweden.
- [14] Ma, R., & Zhang, H. M. (2017). The morning commute problem with ridesharing and dynamic parking charges. Transportation Research Part B: Methodological, 106, 345-374.
- [15] Masoud, N., Nam, D., Yu, J., & Jayakrishnan, R. (2017). Promoting Peer-to-Peer Ridesharing Services as Transit System Feeders. Transportation Research Record: Journal of the Transportation Research Board, (2650), 74-83.
- [16] Zickuhr, K. M. (2016). When Uber comes to town: The impact of transportation network companies on metropolitan labor markets.
- [17] Rayle, L., Shaheen, S., Chan, N., Dai, D., & Cervero, R. (2014). App-based, on-demand ride services: Comparing taxi and ridesourcing trips and user characteristics in san francisco university of california transportation center (uctc). University of California, Berkeley, United States
- [18] Hayashi, F., & Bradford, T. (2014). Mobile payments: merchants' perspectives. Economic Review, 99, 5-30.
- [19]: Clarke, Steven (2004)."Measuring API Usability". Dr. Dobb's. Retrieved 29 July 2016.
- [20]: What is Software-As-A-Service? Accessed At: https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service [Retrieved 11 December 2018]