

# Guuber

## Requirements Document

Lunwen He: lunwenh, lunwenh@andrew.cmu.edu

Yanning Liu: yanningl, yanningl@andrew.cmu.edu

Ziming Wang: zimingw, zimingw@andrew.cmu.edu

## 1 Introduction

### 1.1 Purpose of this document

### 1.2 Background

### 1.3 Goals

### 1.4 System overview

## 2 User Story

### 2.1 Driver

### 2.2 Passenger

## 3 Use Case

### 3.1 Sign up

### 3.2 Sign in

### 3.3 Update profile

### 3.4 Find driver

### 3.5 Find passenger

### 3.6 Send message

### 3.7 Start service

### 3.8 End service

### 3.9 View transaction history

## 4 Page flow diagram

## 5 Wireframes

### 5.1 Driver

### 5.2 Passenger

## 6 Project management

### 6.1 Team Organization

### 6.2 Methodologies

### 6.3 Tools

### 6.4 Schedule

# 1 Introduction

## 1.1 Purpose of this document

This document is the introduction and requirement document for Guuber: a car-hailing android application. In the first section, we give some brief description about the background and goals of Guuber. Then we dive into detailed design issues of Guuber. In the second section, two user stories are showed from different kind of user(driver and passenger) perspectives to demonstrate how Guuber can be used in real life. In the third section, more detailed use cases are provided to describe different features of Guuber. In the next two sections, Guuber's page flow diagrams and wireframes are shown separately. In the final section, we talk about our team management, like introductions of our team members, tools we will use during this project, and our planned schedule for this project.

## 1.2 Background

Nowadays, more and more people tend to thumb up a ride when they don't have a car instead of waiting for buses or taxis. Then Uber comes into our life. With Uber application, users can easily call Uber drivers nearby and start a journey as quickly as possible. However, "Uber supply" doesn't meet the demands: after all, Uber drivers are limited because only small portion of drivers are professional Uber drivers. For most drivers, in most cases they just want to pick up someone for a ride on their way home or work, and don't want to spend most of their time on road as Uber drivers do. Under this situation, we decide to develop a car-hailing application called Guuber. The name Guuber comes from the combination of Google and Uber since most of our technologies are based on Google API and our motivation and concepts are inspired by Uber.

## 1.3 Goals

With the help of Guuber, drivers and passengers can input their routes they are planning to go. Based on the drivers' routes and the passengers' routes, Guuber will recommend the most suitable drivers list to passengers and the most suitable passengers list to drivers (two way choice). Then they can choose to chat to someone they want to pick or take via this application, and start the journey.

There are mainly two significant goals of Guuber. One is that it provides a free way for drivers to find passengers as they want, either for making some money or just providing convenience to others if they want. Another goal is that it provides more choices for people to find a thumb-up ride.

## 1.4 System overview

We decide to use Android platform to develop Guuber, as android phones still domain the smartphone domain. We need these following Android hardware and software features to develop Guuber:

### 1. GPS

Both drivers and passengers need GPS to pinpoint locations. Also Guuber needs users' locations to recommend driver/passenger list.

### 2. Connectivity

Since the application requires the support of Google Map and Navigation, we need Android based connectivity, like Wi-Fi or 4G-LTE, to connect Internet.

### 3. Storage

We need to store chat history in client-sid. So we need a light-weight database, which is SQLite in Android.

We will utilize lessons learned from Mini1 to develop Guuber. For instance, the exception handling when an exception occurs and serialization/de-serialization of infos to persist storage. We also use OOD Design and CS Mode in our designment and implementation.

## 2 User Story

### 2.1 Driver

Bob works at a big company which is far from his home. Everyday he drives to his company in the morning and drives back to home after work. One day he found Guuber at Google Play and installed it on his Android phone and signed up as a driver. Before leaving his company, he opened Guuber and tried to find a passenger who could take his car back to home with the similar route as him. When he opened Guuber, there were several passengers in Guuber waiting and finding drivers. Bob chose one of them who was going back to home with a similar route with him. He clicked that passenger and sent a hello message to him and asked if he was looking for a car. After a while, the passenger replied his message with yes. Then, they talked about the price of that lift and came to an agreement with that transaction. Following Google Map embed in Guuber, Bob found that passenger and let him get in his car. Then, Bob pressed the Start button in Gubber and started driving to home and had a nice conversation with his passenger during that time. When he got to the destination, he pressed the End button in Gubber and received the money from that passenger. With Guuber, he not only found someone to talk with him when driving, but also made some money!

## 2.2 Passenger

Alex just graduated from his college and found a job at a company which is a little far from where he lives in. Everyday, he has to catch Caltrain or VTA to get to his company or get back to his home. Usually it takes a long time and makes inconvenience to him. Sometimes he misses the public transportation and gets late for his work. He really hates that. One day he heard about Guuber from his friends and installed it on his Android phone. He signed up as a passenger. When he was having breakfast, he opened Guuber to see if he could find a car to get a lift to his company. After he posted his request on Guuber by signing in as a passenger and pressed the Find a Driver button, he found that there were several drivers near his house. He sent messages to some of them and got responses from some of them. Then, he chose the nearest driver and talked about the price of his lift and came to an agreement with that driver. Just a few minutes later, the driver arrived at his house. He got in that car and got a lift to his company. When arriving at his company, he paid his lift and arrived at his company within just 20 minutes!

## 3 Use Case

### 3.1 Sign up

<b>Use Case ID:</b> A	<b>Use Case Name:</b> Sign Up
<b>Primary Actor(s):</b>	Users (Drivers and Passengers)
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows user to sign up a new account in Guuber as driver or passenger.
<b>Preconditions:</b>	User chooses 'sign up' from the Sign In/ Sign Up page
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. Enter a valid(pass Guuber's name policy) username</li> <li>2. Enter a valid(pass Guuber's password policy) password</li> <li>3. Retype same password as password in step 2</li> <li>4. Select an user type (driver/passenger)</li> <li>5. Select gender</li> <li>6. Enter email address</li> <li>7. Enter car license number (if select driver as user type in step 4)</li> <li>8. Press the Submit button</li> </ol>
<b>Postconditions:</b>	After submitting, user registers successfully and enters the home page

<b>Frequency of Use:</b>	Medium
<b>Alternative Flows:</b>	During any time of this use case, user can go back to the Sign In/ Sign Up page to sign in with an existing account or restart a new sign up process. In any steps of this use case, if the information input is invalid, Guuber will provide alert message and user has to input again.
<b>Exceptions:</b>	<ol style="list-style-type: none"> <li>1. Entered information is incomplete</li> <li>2. Username already exists</li> </ol>
<b>Assumptions:</b>	UI is running. Connection to back-end server has setup. User can provide all required and valid information.
<b>Issues:</b>	Connection to back-end server fails
<b>Associated Requirements:</b>	UI should be user friendly. User information should persist in the database and can be queried successfully.

### 3.2 Sign in

<b>Use Case ID: B</b>	<b>Use Case Name:</b> Sign In
<b>Primary Actor(s):</b>	Users (Drivers and Passengers)
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows users to sign in
<b>Preconditions:</b>	User chooses 'sign in' from the Sign In/ Sign Up page
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. Enter the username</li> <li>2. Enter the password</li> <li>3. Press the Log-In button</li> </ol>
<b>Postconditions:</b>	After submitting, user enters the home page
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	During anytime of this use case, user can go back to the Sign In/ Sign Up page to sign up for a new account. If username and password do not match any entries stored in back-end system, Guuber will provide alert message and user has to input again.
<b>Exceptions:</b>	User didn't fill in both username and password
<b>Assumptions:</b>	UI is running. Connection to back-end server has setup. The user has

	already signed up a Guuber account.
<b>Issues:</b>	Connection to back-end server fails
<b>Associated Requirements:</b>	UI should be user friendly. User information should persist in the database and can be queried successfully.

### 3.3 Update profile

<b>Use Case ID: C</b>	<b>Use Case Name:</b> Update Profile
<b>Primary Actor(s):</b>	Users (Drivers and Passengers)
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows users to update their profile
<b>Preconditions:</b>	In the home page after logging in successfully, the user selects Update Profile
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. Enter a valid(pass Guuber's name policy) username</li> <li>2. Enter a valid(pass Guuber's password policy) new password</li> <li>3. Retype same password as password in step 2</li> <li>4. Select Gender</li> <li>5. Enter an Email address</li> <li>6. Enter car license number (if the user is a driver)</li> <li>7. Press the Save button</li> </ol>
<b>Postconditions:</b>	After saving, the user goes back to the home page
<b>Frequency of Use:</b>	Low
<b>Alternative Flows:</b>	During any time of this use case, user can choose to go back to homepage or skip editing any part of user profile and press Save button.
<b>Exceptions:</b>	<ol style="list-style-type: none"> <li>1. Username already exists</li> <li>2. Password has a low security level</li> </ol>
<b>Assumptions:</b>	UI is running. Connection to back-end server has setup. The user has a Guuber account and has logged in successfully.
<b>Issues:</b>	Connection to back-end server fails
<b>Associated Requirements:</b>	UI should be user friendly. Data inside back-end database should be updated.

### 3.4 Find driver

<b>Use Case ID:</b> D	<b>Use Case Name:</b> Find driver
<b>Primary Actor(s):</b>	Passenger
<b>Secondary Actor(s):</b>	Driver
<b>Description:</b>	This use case allows passenger to find a car to get a lift
<b>Preconditions:</b>	The passenger has signed into Guuber
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The passenger selects Find Driver</li> <li>2. The passenger selects his/her destination</li> <li>3. The passenger selects some available drivers and sends messages to them</li> <li>4. If the passenger receives a message from other drivers, he/she can choose to reply that message or ignore that message</li> </ol>
<b>Postconditions:</b>	After reaching an agreement with a driver, the passenger gets a lift and waits for the driver to pick him/her up.
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	The passenger can choose to cancel this use case during this use case and return to homepage
<b>Exceptions:</b>	<ol style="list-style-type: none"> <li>1. There isn't any available drivers right now.</li> <li>2. No driver replies his message</li> </ol>
<b>Assumptions:</b>	Available drivers are showed in the map. The passenger can only send messages to available drivers.
<b>Issues:</b>	Internet connection fails
<b>Associated Requirements:</b>	UI should be user friendly. Google Map should be running smoothly.

### 3.5 Find passenger

<b>Use Case ID:</b> E	<b>Use Case Name:</b> Find passenger
<b>Primary Actor(s):</b>	Driver



<b>Secondary Actor(s):</b>	Passenger
<b>Description:</b>	This use case allows driver to find a passenger
<b>Preconditions:</b>	The driver has signed into Guuber
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The driver selects find passenger</li> <li>2. The driver selects his/her destination</li> <li>3. The driver selects some waiting passengers and send messages to them</li> <li>4. If the driver get messages or responses from other passengers, he/she can choose to view and reply messages or just ignore them</li> </ol>
<b>Postconditions:</b>	The driver has came to an agreement with one passenger and start driving to that passenger to pick him/her up
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	The driver can choose to cancel this use case during this use case and return to homepage
<b>Exceptions:</b>	<ol style="list-style-type: none"> <li>1. There isn't any available passengers right now.</li> <li>3. No passenger replies his message</li> </ol>
<b>Assumptions:</b>	Available passengers are showed in the map. The driver can only send messages to available passengers.
<b>Issues:</b>	Connection to Internet fails
<b>Associated Requirements:</b>	UI should be user friendly. Google Map should be running smoothly.

### 3.6 Send message

<b>Use Case ID:</b> F	<b>Use Case Name:</b> Send message
<b>Primary Actor(s):</b>	Users (Drivers and Passengers)
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows drivers to send messages to passengers and passengers to send messages to drivers.
<b>Preconditions:</b>	The driver or passenger has signed into Guuber. If his/her user type is

	driver, he/she has decided to find a passenger, pressed the Find Passenger button at homepage and there are passengers waiting. If his/her user type is passenger, he/she has decided to find a driver, pressed the Find Driver button at homepage and there are available drivers.
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. User(driver/passenger) selects one available user(passenger/driver) showed in Google map and opens the chat room.</li> <li>2. User types message and presses Send button.</li> </ol>
<b>Postconditions:</b>	The message is sent to the destination. Another user can see the notification of new message and select to view the new message or ignore it.
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	User can choose to cancel this use case during this use case and return to homepage, or select to chat with other user.
<b>Exceptions:</b>	<ol style="list-style-type: none"> <li>1. The content of the message is empty.</li> <li>2. The destination of the messages becomes unavailable(like the passenger has got a lift or the driver has started a service).</li> </ol>
<b>Assumptions:</b>	If the user is a driver, there are passengers waiting and looking for a driver. If the user is a passenger, there are available drivers.
<b>Issues:</b>	Connection to back-end server fails
<b>Associated Requirements:</b>	UI should be user friendly. Google Map should be running smoothly.

### 3.7 Start service

<b>Use Case ID:</b> G	<b>Use Case Name:</b> Start service
<b>Primary Actor(s):</b>	Driver
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows driver to start a service
<b>Preconditions:</b>	The driver has came to an agreement with that passenger.
<b>Normal Flow of</b>	<ol style="list-style-type: none"> <li>1. The driver presses the Start Service button and follows the</li> </ol>

<b>Events:</b>	navigation of Google Map
<b>Postconditions:</b>	<ol style="list-style-type: none"> <li>1. The state of the driver will changes to unavailable.</li> <li>2. The driver will no longer receive messages from other passengers, he also cannot send messages to other passengers before the ending of this service.</li> </ol>
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	N/A
<b>Exceptions:</b>	N/A
<b>Assumptions:</b>	The drive has came to an agreement with his passenger.
<b>Issues:</b>	The passenger quits the service after starting.
<b>Associated Requirements:</b>	UI should be user friendly. Google Map should be running smoothly.

### 3.8 End service

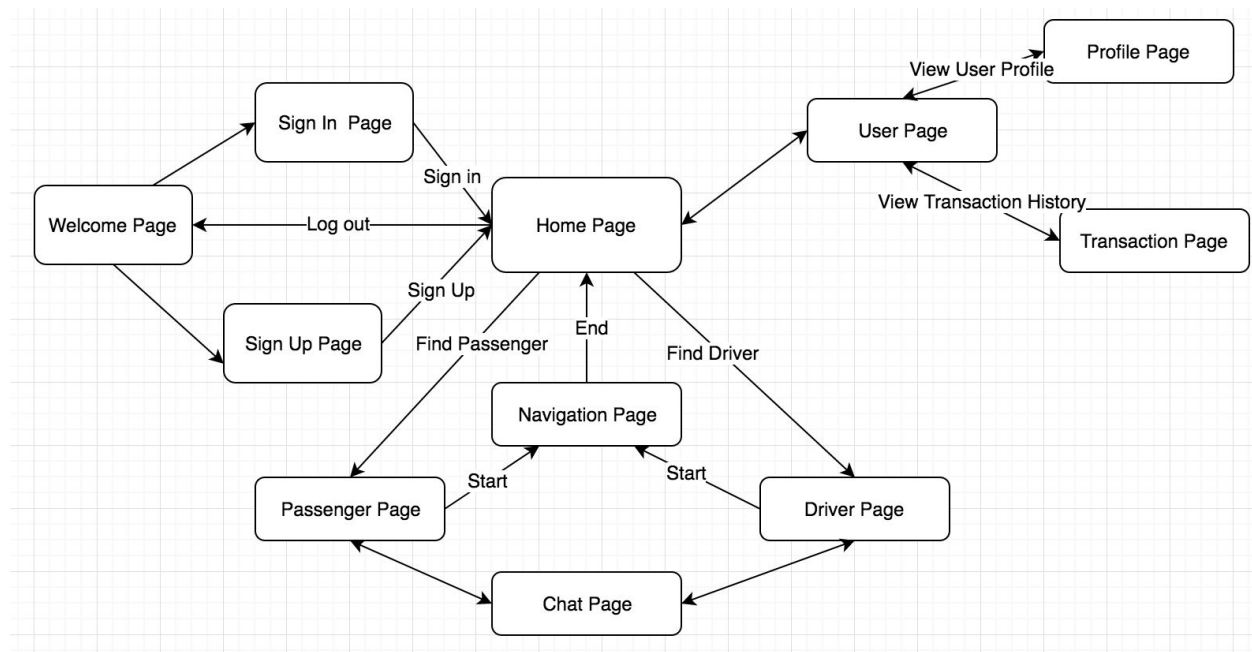
<b>Use Case ID:</b> H	<b>Use Case Name:</b> End service
<b>Primary Actor(s):</b>	Driver
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows driver to finish the service
<b>Preconditions:</b>	The driver has started the service.
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The driver presses the End Service button.</li> </ol>
<b>Postconditions:</b>	The service is finished and return to homepage
<b>Frequency of Use:</b>	High
<b>Alternative Flows:</b>	N/A
<b>Exceptions:</b>	N/A
<b>Assumptions:</b>	The driver and passenger have arrived at the destination.
<b>Issues:</b>	Connection to back-end server fails

<b>Associated Requirements:</b>	UI should be user friendly. Google Map should be running smoothly.
---------------------------------	--

### 3.9 View transaction history

<b>Use Case ID: I</b>	<b>Use Case Name:</b> View Transaction History
<b>Primary Actor(s):</b>	Users (Drivers and Passengers)
<b>Secondary Actor(s):</b>	N/A
<b>Description:</b>	This use case allows users to view their transactions history
<b>Preconditions:</b>	User has signed in.
<b>Normal Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. User selects one transaction and views it's details</li> <li>2. Press the BACK button to the home page</li> </ol>
<b>Postconditions:</b>	The user goes back to the home page
<b>Frequency of Use:</b>	Medium
<b>Alternative Flows:</b>	During anytime of this use case, user can choose to go back to homepage
<b>Exceptions:</b>	There isn't any available transaction history.
<b>Assumptions:</b>	UI is running. Connection to back-end server has setup. The user has a Guuber account and has logged in successfully.
<b>Issues:</b>	Connection to back-end server fails
<b>Associated Requirements:</b>	UI should be user friendly. Database should persist data and can be queried successfully.

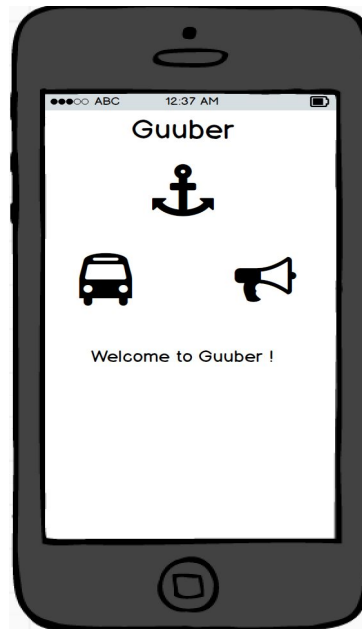
## 4 Page flow diagram



## 5 Wireframes

### 5.1 Driver

1. Welcome



## 2. Sign-up





A smartphone screen displaying the 'Guuber' registration form. The status bar at the top shows 'ABC' and '10:05 AM'. The form fields are: 'UserName' with 'Bob', 'Password' with '\*\*\*\*\*', 'RePassword' with '\*\*\*\*\*', 'UserType' with a dropdown menu showing 'Driver', 'Gender' with a dropdown menu showing 'M', 'Email' with 'bob@gmail.com', and 'Car-ID(\*)' with '7262'. A 'Submit' button is at the bottom.

Guuber

UserName Bob

Password \*\*\*\*\*

RePassword \*\*\*\*\*

UserType Driver ▼

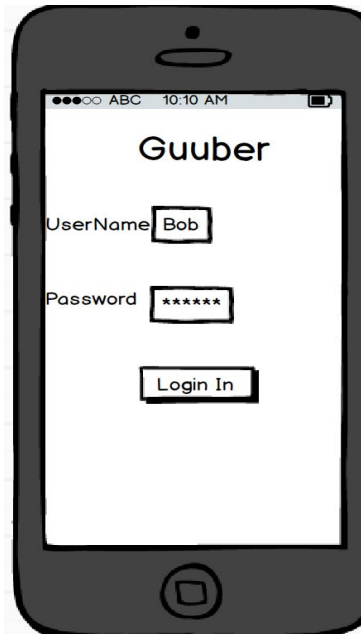
Gender M ▼

Email bob@gmail.com

Car-ID(\*) 7262

Submit

### 3. Sign-in



A smartphone screen displaying the 'Guuber' sign-in form. The status bar at the top shows 'ABC' and '10:10 AM'. The form fields are: 'UserName' with 'Bob' and 'Password' with '\*\*\*\*\*'. A 'Login In' button is at the bottom.

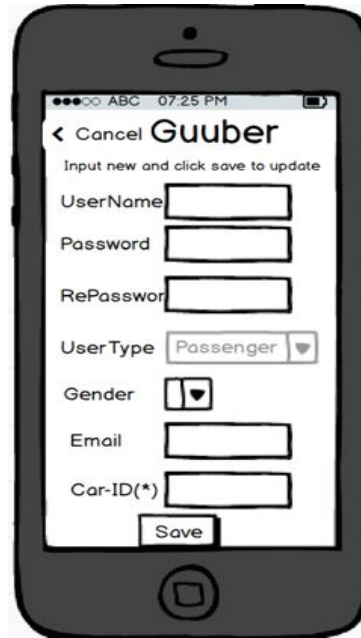
Guuber

UserName Bob

Password \*\*\*\*\*

Login In

### 4. Update Profile



A hand-drawn sketch of a smartphone screen displaying the 'Guuber' registration form. The status bar at the top shows 'ABC' and '07:25 PM'. The app title 'Guuber' is at the top left with a back arrow. Below it is the instruction 'Input new and click save to update'. The form contains the following fields: 'UserName' (text input), 'Password' (text input), 'RePassword' (text input), 'UserType' (dropdown menu with 'Passenger' selected), 'Gender' (radio button with a heart icon), 'Email' (text input), and 'Car-ID(\*)' (text input). A 'Save' button is at the bottom right.

< Cancel **Guuber**

Input new and click save to update

UserName

Password

RePassword

UserType Passenger ▼

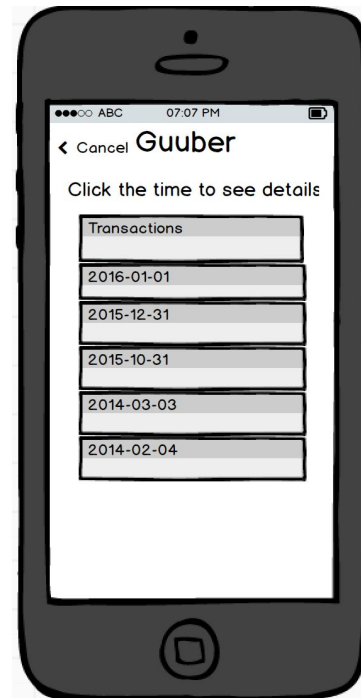
Gender ☐ ♥

Email

Car-ID(\*)

Save

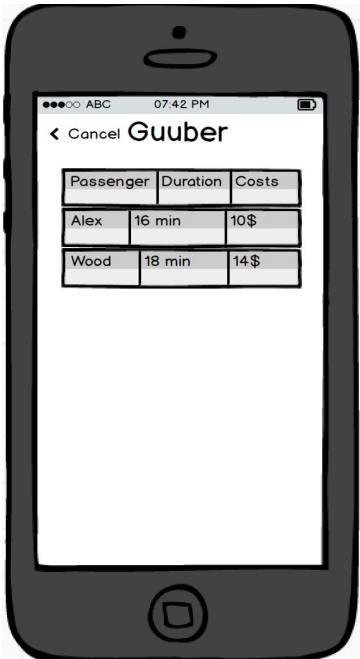
## 5. Transaction History



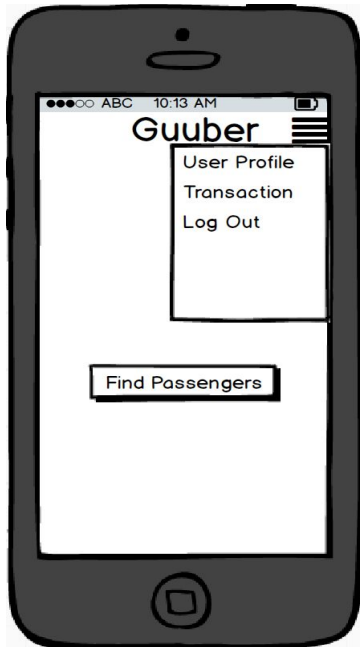
A hand-drawn sketch of a smartphone screen displaying the 'Guuber' transaction history screen. The status bar at the top shows 'ABC' and '07:07 PM'. The app title 'Guuber' is at the top left with a back arrow. Below it is the instruction 'Click the time to see details'. A table with the following data is shown:

Transactions
2016-01-01
2015-12-31
2015-10-31
2014-03-03
2014-02-04





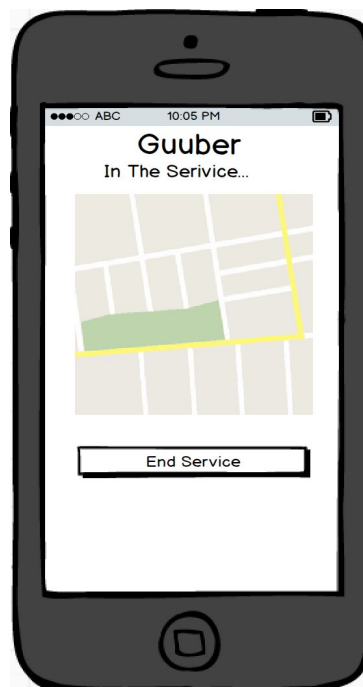
6. Find Passenger



7. Start Service



8. End Service



## 5.2 Passenger

1. Welcome



## 2. Sign-up





A smartphone screen displaying the 'Guuber' registration form. The status bar at the top shows 'ABC' and '10:05 AM'. The form fields are: 'UserName' with 'Alice', 'Password' with '\*\*\*\*\*', 'RePassword' with '\*\*\*\*\*', 'UserType' with a dropdown menu showing 'Passenger', 'Gender' with a dropdown menu showing 'F', 'Email' with 'Alice@gmail.com', and 'Car-ID(\*)' with a greyed-out field. A 'Submit' button is at the bottom.

Guuber

UserName Alice

Password \*\*\*\*\*

RePassword \*\*\*\*\*

UserType Passenger ▼

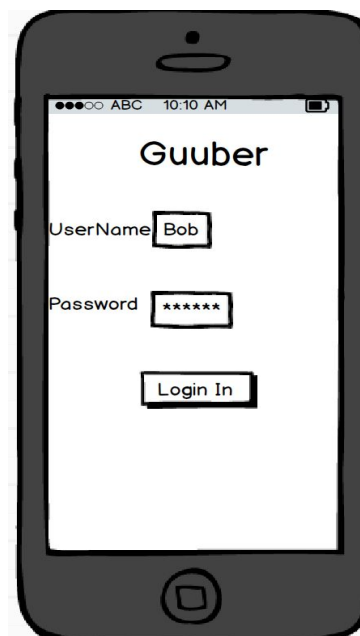
Gender F ▼

Email Alice@gmail.com

Car-ID(\*)

Submit

### 3. Sign-in



A smartphone screen displaying the 'Guuber' sign-in form. The status bar at the top shows 'ABC' and '10:10 AM'. The form fields are: 'UserName' with 'Bob' and 'Password' with '\*\*\*\*\*'. A 'Login In' button is at the bottom.

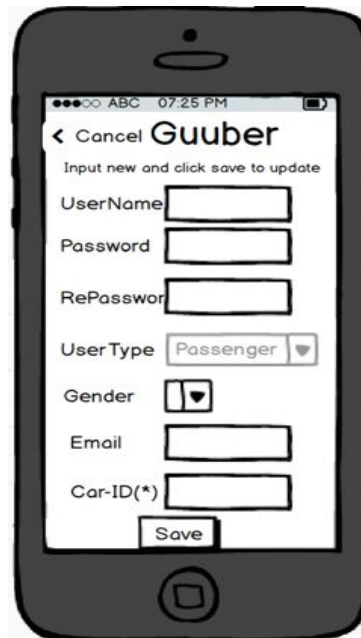
Guuber

UserName Bob

Password \*\*\*\*\*

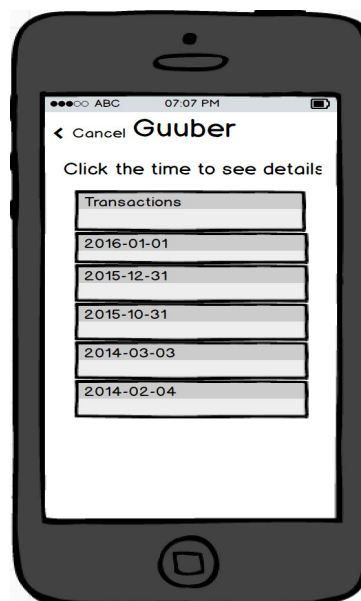
Login In

### 4. Update Profile



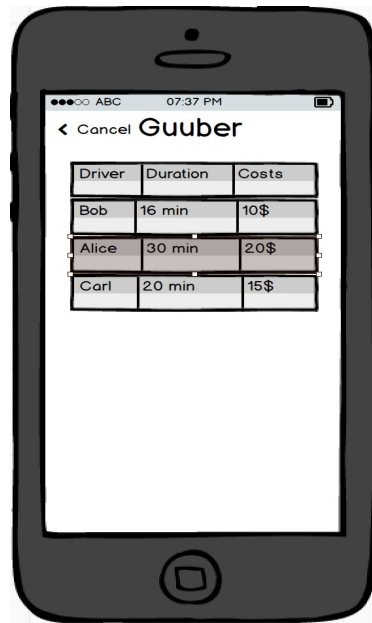
A screenshot of a mobile app interface for registration. The status bar at the top shows 'ABC' and '07:25 PM'. The app title is 'Guuber' with a back arrow. Below the title is the instruction 'Input new and click save to update'. The form contains the following fields: 'UserName' (text input), 'Password' (text input), 'RePassword' (text input), 'UserType' (dropdown menu with 'Passenger' selected), 'Gender' (radio button with a dropdown arrow), 'Email' (text input), and 'Car-ID(\*)' (text input). A 'Save' button is located at the bottom of the form.

## 5. View Transaction History

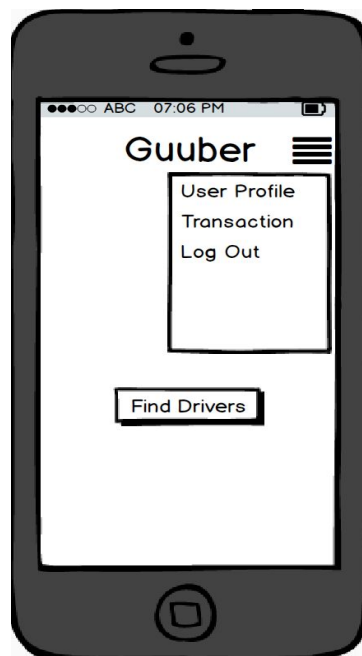


A screenshot of a mobile app interface for viewing transaction history. The status bar at the top shows 'ABC' and '07:07 PM'. The app title is 'Guuber' with a back arrow. Below the title is the instruction 'Click the time to see details'. A table titled 'Transactions' displays a list of dates. The dates are: 2016-01-01, 2015-12-31, 2015-10-31, 2014-03-03, and 2014-02-04. Each date is enclosed in a rectangular box, indicating it is a clickable element.

Transactions
2016-01-01
2015-12-31
2015-10-31
2014-03-03
2014-02-04



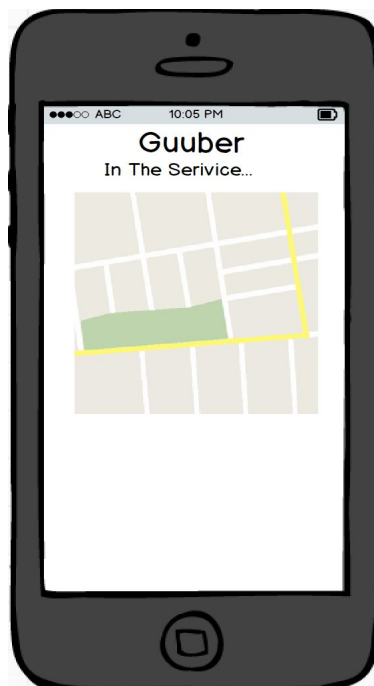
6. Find Driver



7. Start Service



## 8. End Service



## 6 Project management

### 6.1 Team Organization

Our team consists of 3 members, with somewhat equal responsibilities. Our team will try to distribute work evenly and allow for members to contribute to every part of the project as individual schedules and workloads permit.

### 6.2 Methodologies

Our team will use a combination of Kanban and Scrum software development methodologies to define tasks and move them forward from a backlog to development, testing, review, and deployment. Our team will also hold regular meetings to give updates on progress and to recognize and resolve issues early on. Trello will be used to facilitate the management of tasks, and Slack will be used to streamline communication between team members. More details of tools we will use are talked about in the next section.

### 6.3 Tools

Task	Tool	Description
Page Flow	draw.io	A free web application to draw a variety of diagrams for designment, such as flowcharts, process diagrams, class diagrams, UML and ER diagrams.
Wireframe	Balsamiq Mockups 3	A convenient wireframing tool that allows us to build UI mockups easily.
Version Control	Github	We upload code and documents into the team Github repository, so that we can work collaboratively and keep control of editing history and different versions. Link: <a href="https://github.com/helunwencser/Guuber.git">https://github.com/helunwencser/Guuber.git</a>
Communication	Slack	Besides regular face-to-face meeting, the team will also communicate ideas online with Slack.
Progress Management	Trello	Trello can be used by the team to manage tasks. It organizes tasks into Backlog, Development, Test and Done lists. It also keeps track of the schedule of each



		task as well as resources and personnel assignment.
Continuous Integration	Travis-CI	Travis-CI is used as our continuous integration platform.
Android Development	Android Studio	Guuber application will be built with the latest version of Android Studio with Android 6.0 (Marshmallow).

## 6.4 Schedule

Task Name	Start Date	End Date	Mar				Apr				May		
			Mar 6	Mar 13	Mar 20	Mar 27	Apr 3	Apr 10	Apr 17	Apr 24	May 1	May 8	May 15
			⚙️ 🔍 @										
Compile Project Requirement	03/16/16	03/28/16					Compile Project Requirement						
Refine Project Requirements	03/29/16	04/04/16					Refine Project Requirements						
Design Project	04/05/16	04/14/16								Design Project			
Construction Phase 1	04/15/16	04/25/16									Construction Phase 1		
Construction Phase 2	04/26/16	05/02/16										Construction Phase 2	
Final Presentations	05/01/16	05/06/16									Final Presentations		