SRS (Software Requirement Specifications)

Transport-Al Project Reports

Version 1.0

TRANSPORT-AL - AN ANDROID APP/ADMIN PANEL/ARDUINO- APPLICATION

by

Group 19

Cork Institute of Technology, 2018

A REPORT

submitted in fulfilment of the requirements for the Project

Software Development 3rd Year-1st Semester

Department of Computing

CORK INSTITUTE OF TECHNOLOGY

Bishoptown, Cork

2018

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

We take this opportunity to express my sincere gratitude and thanks to Mr. Seamus Lankford, our Professor for his constant guidance and support through his suggestions and ideas throughout our group project. You played a significant role in the successful completion of our project. Mr. Lankford, it has been an honor to work with you.

I would like to express my appreciation to my group members, Mr. Rudy Murer, Mr. Conor Morgan, Mr. Nicholas Horgan.

INTRODUCTION

Transport is an integral part of our social living. The modern society cannot run without transport facilities. There are many companies who give transport services to the individual and corporate clients. In the current system, the client first contacts with the transport company for getting transport service. The company then books the vehicle for him on the requested date and time and then sends the vehicle to his place at the time. The Transport-Al system is the online service which will automate the process of booking a taxi/cab and will facilitate both the client and the company with reduced time and efforts. First the company will register his vehicles and the vehicles to the system. Then the client will request for booking a vehicle, providing all necessary information. The fare will be calculated, and client should confirm it. Then the vehicle will serve the client. Finally, the client will have an opportunity to give a feedback for the service he got. The company can check it and take appropriate action for the future improvements.

PURPOSE

The purpose of this SRS document is to specify software requirements of the transport-Al Online Taxi Booking through Android Mobile App as well as to track the vehicle real time through Arduino device and Admin panel to manage the backend of the Application. It is intended to be a complete specification of what functionality the system provides. The main purpose of the system is to automate the process of booking a taxi online. Specific design and implementation details will be specified in a future document.

PROJECT SCOPE

This project's aim is to automate the system, booking the cab, calculating the fare, collecting fare, paying bill through app collecting all necessary information of the client and then serve the client. The data used by the system is stored in a google firebase database that will be the centre of all information held clients and vehicle. This enables things to be simplified and considerably quickened, making the jobs of the people involved easier. It supports the current process but centralizes it and makes it possible for decisions to be made earlier and easier way.

GOALS

The main goal of the system is to automate the process carried out in the organization with improved performance and realize the vision of online booking. Some of the goals of the system are listed below:

- Manage large number of client details.
- Manage all details of clients who registered and requested for getting the service.
- Create client accounts and maintain the data's effectively.
- View all the details of the clients and vehicle.
- Showing available vehicles to book for the client.
- Calculating and showing the fare to client before booking.
- Create the statistical reports to facilitate the finance department work.
- Getting the feedback from the client to facilitate future improvement.
- Tracking in real time vehicle position.

OBJECTIVES OF THE PROPOSED SYSTEM:

The aim of the proposed system is to address the limitations of the current system. The requirements for the system have been gathered from the defects recorded in the past and also based on the feedback from users of previous metrics tools. Following are the objectives of the proposed system:

- **Reach to geographically scattered clients.** One of the important objectives of the online booking system is communicate with all the clients scattered geographically.

- **Automate the process of booking.** The system will reduce the time and effort of the clients and employees and automate the process of booking.

- **Centralized data handling**. Transfer the data smoothly to all the departments involved and handle the data centralized way.

- **Reduced manpower**. Reduce the manpower needed to perform the booking and serving clients.

- **Cost cutting**. Reduce the cost involved in the booking process.

- **Operational efficiency**. Improve the operational efficiency by improving the quality of the process.

BENEFITS OF THE SYSTEM

As with most real-world activities, there are numerous benefits to using a software system taxi booking like Transport-Al. The most apparent to this project is the unification of the entire process. Another benefit of a software system is the use of a central database. This database is the basis for all actions in the system and can be trivially updated and used to aid in all of the system's processes, meaning all of the required information is stored in one central location and thus is easily accessible. This is a far more reasonable storage method than a paper-based file system, where the time of traveling to and physically searching the records for the required information could be a burden. Human error could also be a factor in that mistakes could be made in the filing process which would not occur in a well written database system and mistakes or changes on physical records can be messy to correct. Software systems are also much faster at performing certain tasks than humans, meaning that time can be saved performing processes such as sending communication emails, creating recommendations and the comparison of applications. This also means that these tasks can be done solely by the system, freeing up those involved to perform more important tasks.

TECHNOLOGIES

- Google drive
- Android Studio
- Google Firebase Server and Database
- Atom IDE and Node.js used for PayPal payment implementation.
- For internal communication with client email is be used.
- Trello, Slack, WhatsApp, GitHub: Used for the development and maintenance of the group project.
- Install shield: Package to be used to simplify the installation process of the software.
- JavaScript used for Admin part developed.
- Arduino Uno
- 1Sheeld + App

OVERVIEW

SRS will include two sections.

Overall Description will describe major components of the system, interconnection, and external interfaces.

Specific Requirements will describe the functions of actors, their role in the system and constraints.

OVERALL DESCRIPTION

App Perspective

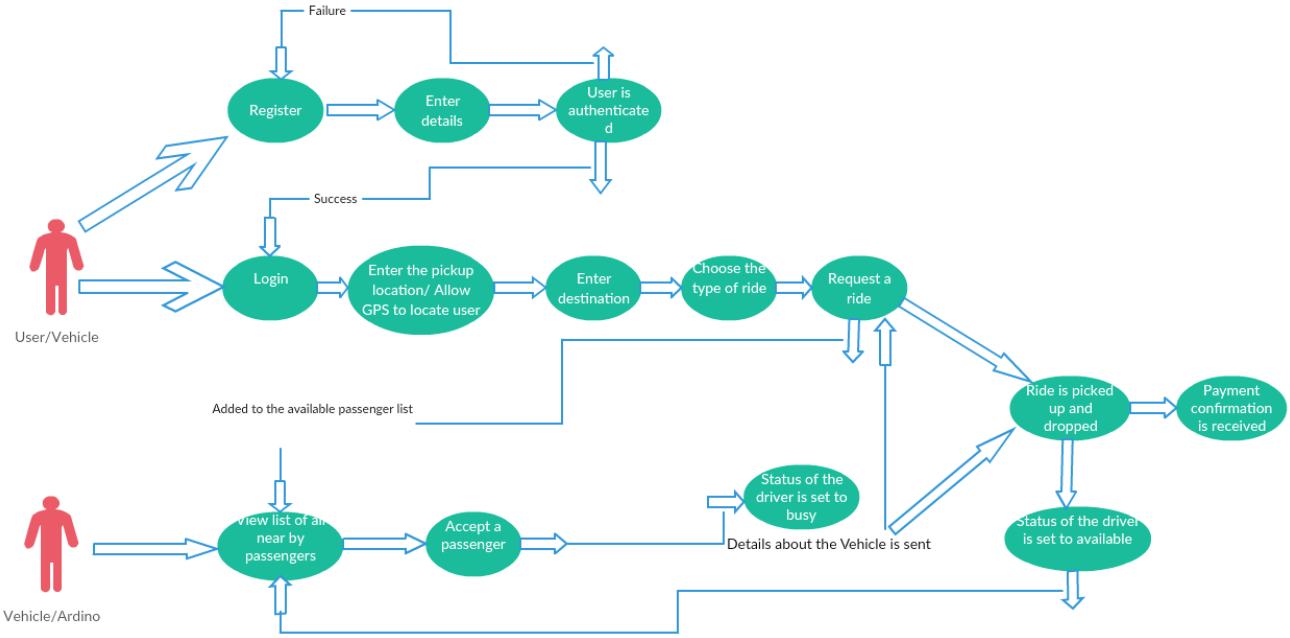


Figure 1: Model of the System

APP FEATURES

Some of the features are identified for the software. They are listed below:

- **View Available Vehicles:** The client must able to see all details about the available vehicles without any constraints.
- **Log in and log out from the App**
- **Calculate Fare:** The client must be available to check the fare they should pay for the vehicles.
- **Rating:** The administrator can able to see the feedback given by each client so that he can take appropriate actions for future improvement.
- **Payment:** User can pay his/her total bill through App
- **Register:** Client can register with the App
- **Tracking the location:** User can see and tracking the location of the vehicle

USER CLASSES AND CHARACTERISTICS

USER CHARACTERISTICS

The client should have the basic idea to operate (use) the system and he already has the experience to work in the app (android). Default Language is English.

SYSTEM REQUIREMENTS AND ANALYSIS:

The following sections will introduce the numerous requirements of the system from the point of view of different users and will introduce a number of decisions that have been made regarding implementation. These sections also attempt to somewhat describe the role of each user group in the system, discussing their individual roles through the functions they can perform.

USER INTERFACE

The user interface for this system will have to be simple and clear. Most importantly, the ages must be easy to read, easy to understand and accessible. The color scheme should be appropriate to provide familiarity with the Company and there should be no contrast issues.

CLIENT VIEW FUNCTIONALITY:

Registration and Login System: Clients will carry out their own registration, providing the system with a way to associate a user to their request(s). This will enable the system to display personalized information when the user logs in and certain information, such as name and address, to be added to each booking request automatically.

Booking System: The booking process will be as straightforward as possible, using an intuitive form layout, with the necessary information being completed in stages.

Update Details: clients will all have the ability to update their personal details at any time.

Payment/Bill: Client can pay his/ her bill through the app

Receive Notification from the Drive/Vehicle

Leave Rating for the Driver

Cancel the booking

DRIVER/VEHICLE VIEW FUNCTIONALITY:

1. Registration and Login System
2. Update details/Reset password-if password forgotten
3. Receive notification for new booking
4. Accept/cancel booking
5. Pickup client
6. Drive complete
7. Rating customer
8. Payout

FUNCTIONAL OR SPECIFIC REQUIREMENTS:

The system should satisfy the following requirements:

CLIENT ASPECT:

1. Registration
2. Update details/ Reset password – if password forgotten
3. Make a booking
4. Check their booking status
5. Fair calculation
6. Trip history
7. Pay bill for the ride
8. Accept/cancel booking

DRIVER/VEHICLE ASPECT:

1. Registration
2. Update details/ Reset password – if password forgotten
3. Accept/reject a booking
4. Receive notification for new booking
5. Pickup customer
6. Driver complete when reached to the destination.
7. Trip history
8. Pay out for the ride

ANALYSIS

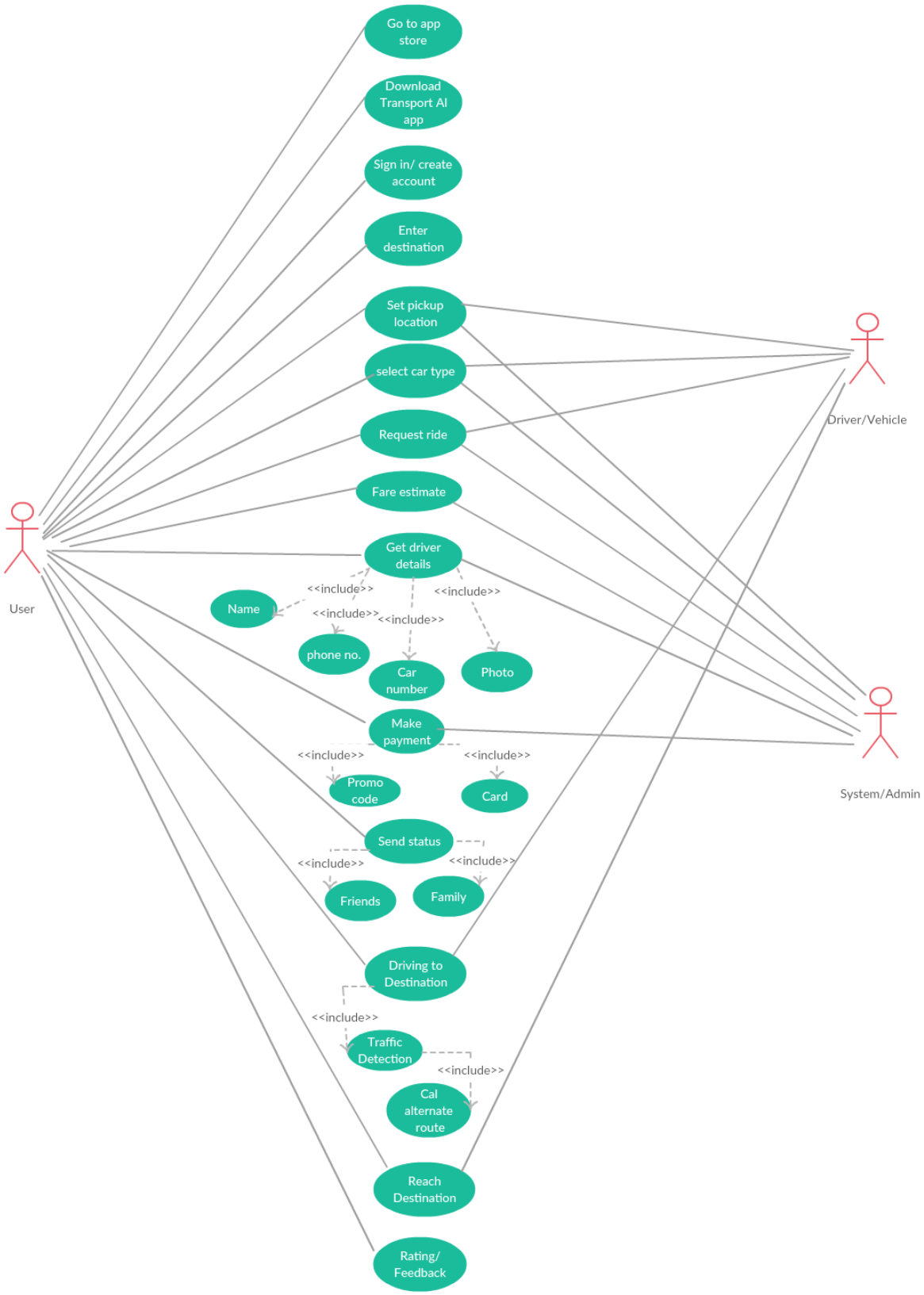
1. Authenticating users based on username and password.
2. Recording client's request for booking.
3. Checking whether the vehicle is available for booking.
4. Keeping history of courses bookings.
5. Keeping record of feedbacks received from the clients.
6. Keeping track of the payment

PERFORMANCE REQUIREMENTS

Some Performance requirements identified is listed below:

- The database shall be able to accommodate a minimum of 10,000 records of clients.
- The software shall support use of multiple users at a time.
- Good Wi-Fi or data connectivity
- App software version 6 or above.
- There are no other specific performance requirements that will affect development.

CASE DIAGRAMS



The details are written as follow :

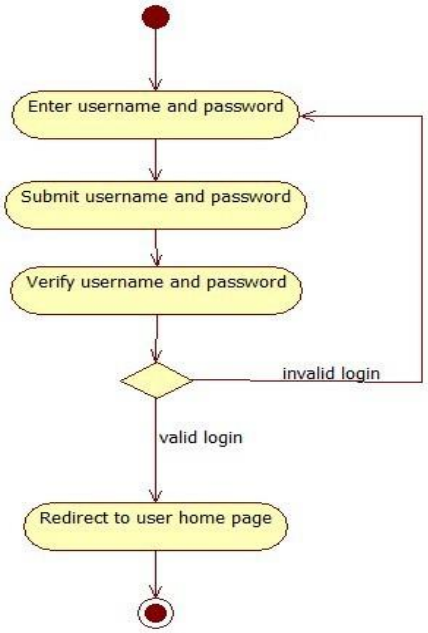
Below a use case scenario for an user requesting a ride with Standard vehicle, the company's most basic ride-sharing service.

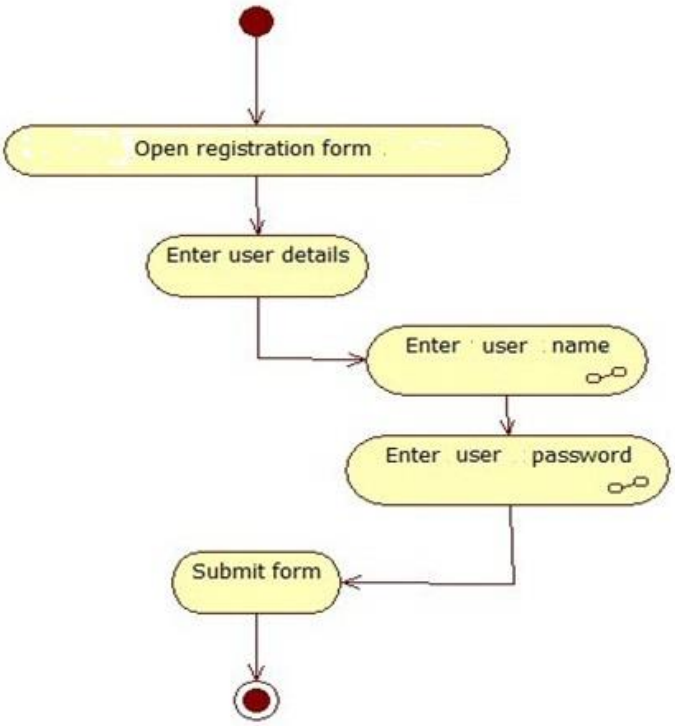
Name	Customer Requests Standard Ride
ID	001 – System Generated ID
Description	How a user will request a ride
Actors	User, Driver/Vehicle, and System
Organizational Benefits	Connect riders with Vehicles for cashless ridesharing
Frequency of Use	Each time a user requests a ride.
Triggers	User clicks "Request Standard"
Assumptions	1.The user is located within a Transport-Al market zone 2. There is at least one available driver/Vehicle within the maximum range of the user.
Preconditions	User has an authenticated Transport-Al account
Postconditions	The driver/Vehicle has dropped off the user at the desired location and the appropriate transaction occurs
	1. User selects "Set Pickup Location" at current or desired pickup location 2. User can select "Request any type of vehicle out of 3"

	3. System sends user pickup location and customer rating to available drivers 4. The closest driver/Vehicle accepts ride in app and the system routes them to user location. 5. User is notified that a driver/Vehicle is found or on the way, listing car make, model, license plate number, driver photo, and driver rating etc. 6. System notifies user via text message when driver is one minute away from the pickup location(have to implement) 7. Driver/Vehicle and user meet at pickup location 8. User or driver input drop-off location in app 9. Driver selects “Picked up Customer” on app, starting the fare calculation (Have to implement and show the user) 10. System routes driver from pickup location to drop-off location and displays navigation.(Its implemented but not showing all the time) 11. Driver drives user to requested drop-off location 12. When drop-off occurs, driver selects “End Ride” in the app 13. System bills the user’s credit card for the final fare amount 14. System credits the balance to driver’s Paypal account 16. Driver is prompted to rate the user before looking for next ride 17. User is prompted to rate the driver the next time they open the app

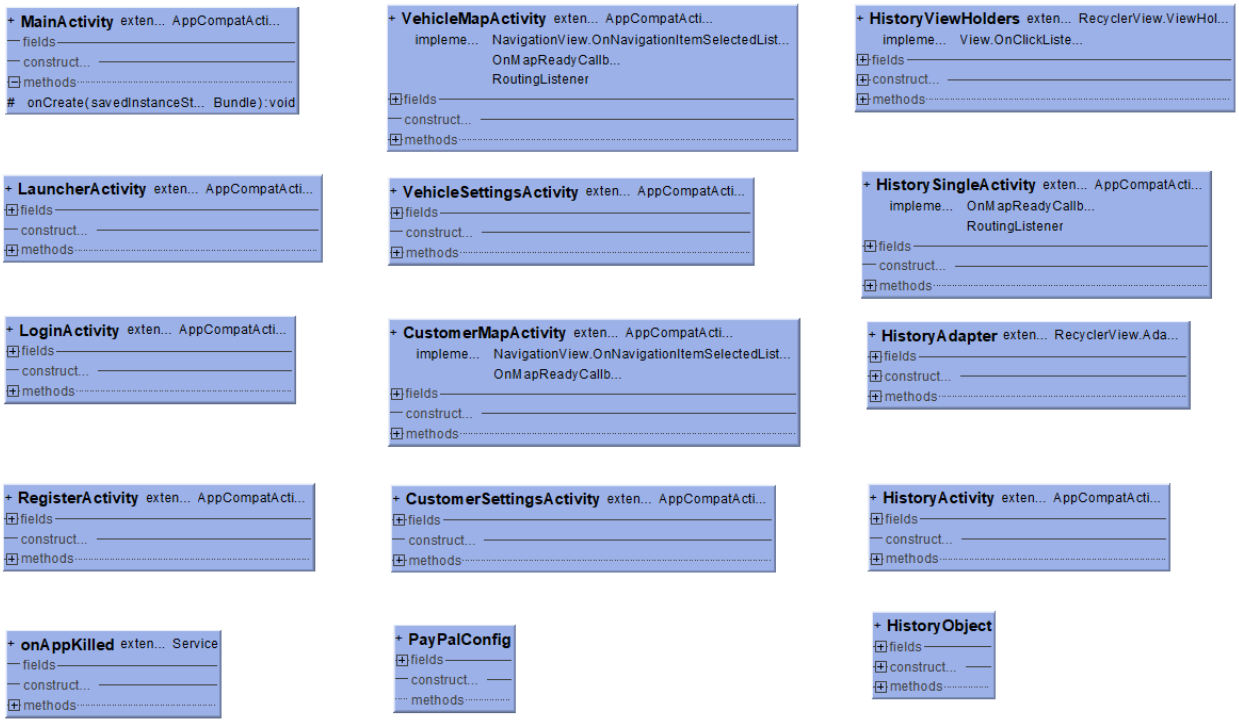
Alternate Courses	AC1: Closest driver does not accept rideIn step 4, if the closest driver does not accept the ride request, the next closest driver is notified until a driver accepts or all drivers within the maximum radius have been notified.
Exceptions	EX1: Driver cancels the ride while the ride is in progress 1. The ride ends and the user exits the vehicle 2. The system does not charge the user for the ride 3. The system prompts the driver to provide an explanation for the canceled rideEX2: User cancels ride request before pickup occurs 1. The user selects “Cancel Pickup” in the app 2. The driver is notified that the pickup has been canceled and is prompted to select the next ride

FLOW DIAGRAM EXAMPLE OF USER LOGIN AND REGISTRATION

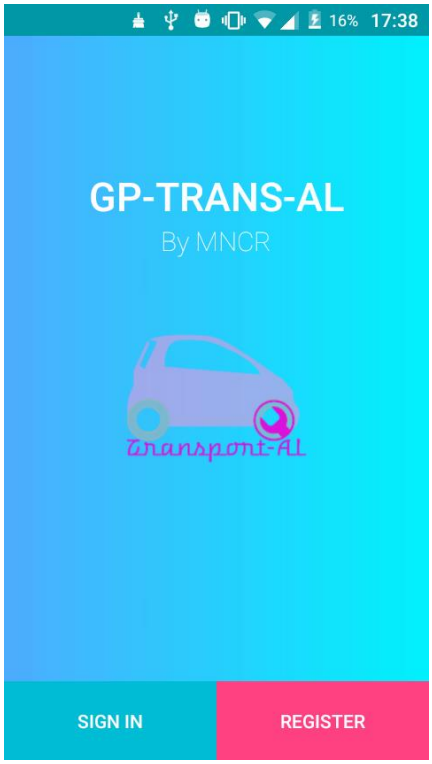




CLASS DIAGRAM

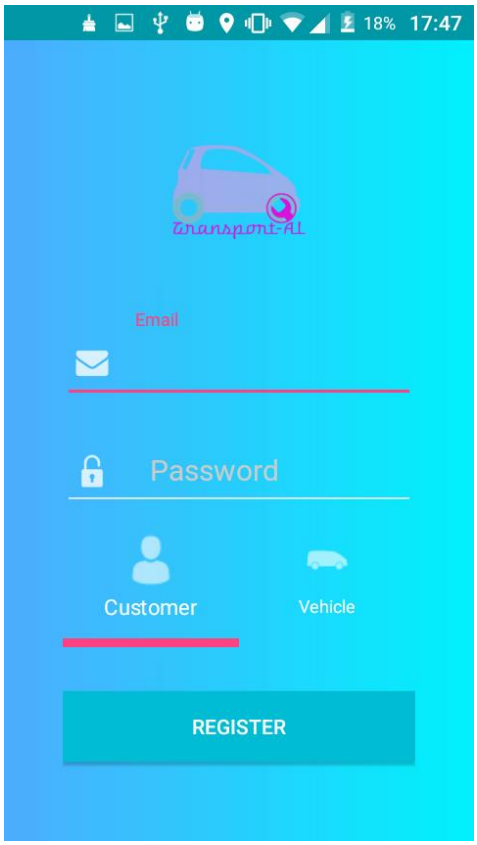


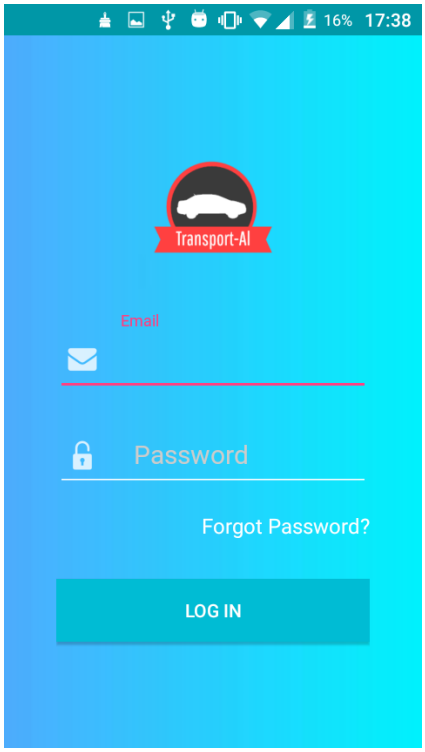
TRANSPORT-AL APP SCREENSHOT EXAMPLE



- Welcome screen
- Logo
- Sign in and Register button

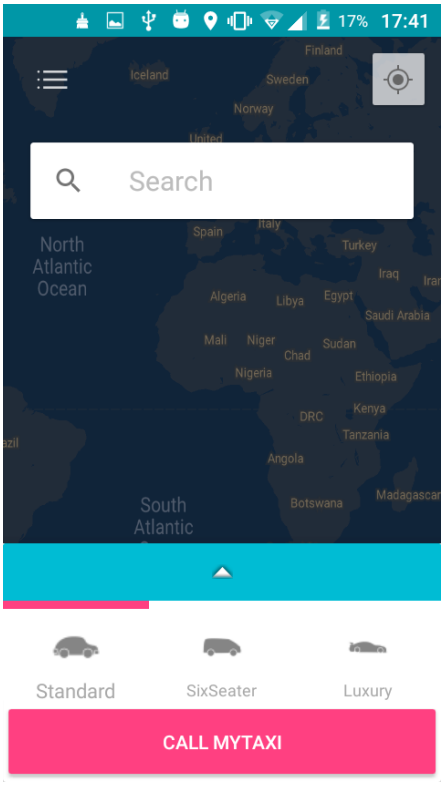
- Register Screen
- Client or Driver/Vehicle
- Email & Password required for register

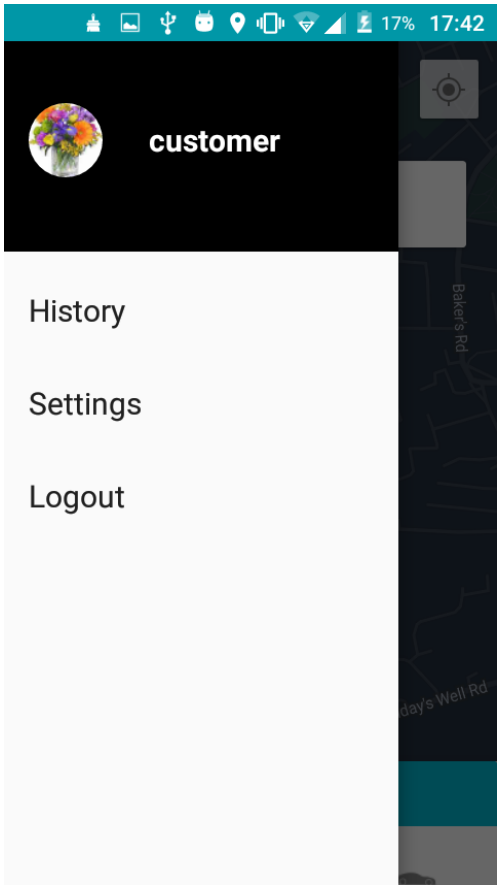




- Log in Screen
- If user forget password will be send to user email for re-set
- Must be valid email and password for log in and it be authenticated with firebase database

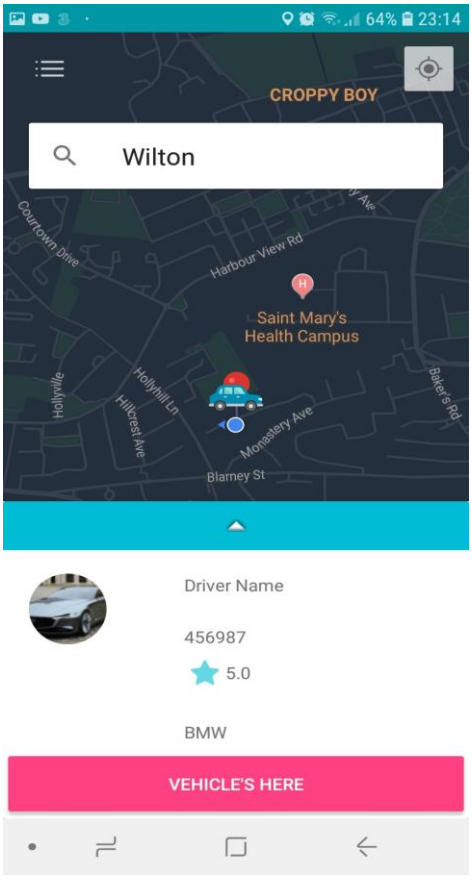
- After successful login user will go to the next stage of the App
- Here user can
  - See the menu
  - To Zoom the map
  - To Search & set the location for pickup
  - Select the vehicle type
  - Update user details
  - See the trip history
  - Hide the bottom layout

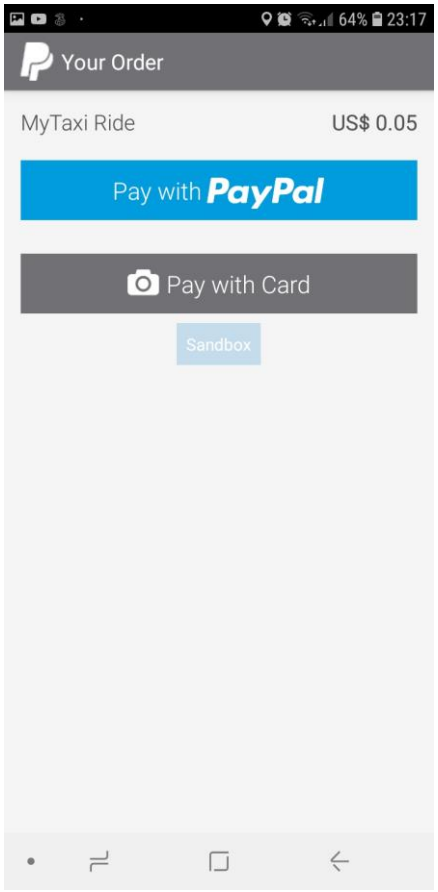




- User menu option
  - Can see the trip history
  - Update user details by setting button
  - Logout from the Application

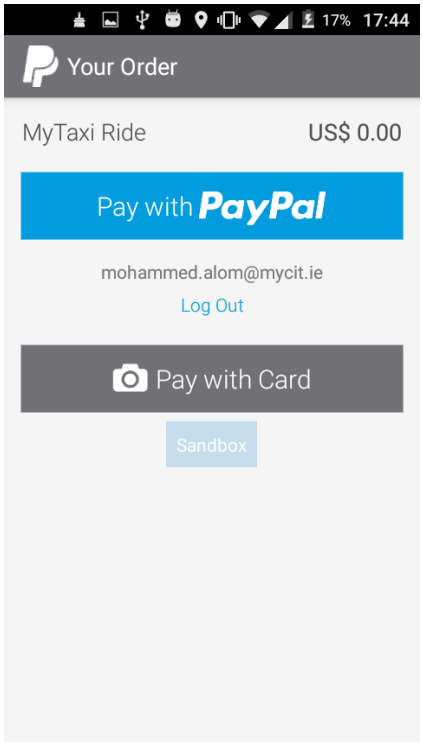
- User will receive notification from the vehicle
  - Name
  - Contact no
  - Vehicle rating
  - Vehicle type

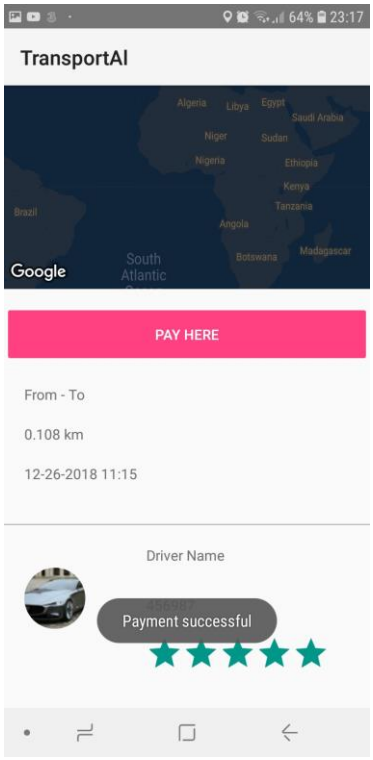




- User will pay after finishing the trip
- Will show total amount in euro
- Pay by PayPal or by Credit Card

- User need to login with PayPal account for pay the ride fees





- After authenticating with PayPal account Client can pay his/her ride bill
- And transaction went successful
- Below is the screen shot of the receipt

PAYMENT RECEIPT FROM PAYPAL SCREEN SHOT

PayPal

Dec 29, 2018 15:52:03 PST
Transaction ID: 69Y44315FJ999290N

Hello Mohammed Alom,

You sent a payment of $0.01 USD to test facilitator's Test Store (asderire-facilitator@yahoo.ie)

It may take a few moments for this transaction to appear in your account.

Merchant
test facilitator's Test Store
asderire-facilitator@yahoo.ie

Instructions to merchant
You haven't entered any instructions.

Description	Unit price	Qty	Amount
MyTaxi Ride	$0.01 USD	1	$0.01 USD

Subtotal	$0.01 USD
Total	$0.01 USD
Payment	$0.01 USD

Charge will appear on your credit card statement as "PAYPAL *TESTFACILIT"
Payment sent to asderire-facilitator@yahoo.ie
Payment sent from mohammed.alom@mycit.ie

Funding Sources Used (Total)

Visa x-3489	$0.01 USD

ADMIN SECTION

THE ADMIN PANEL

The admin website posed quite a few challenges along the way many of which to do with firebase database data manipulation, and Google map integration.

Down below are the, goals, functionality, use cases of the admin panel.

CORE FEATURES OF THE SITE

1. Firebase login
2. Active booking requests
3. Vehicle data
4. Users and there data
5. User transaction history
6. Map view and route tracking
7. Data search and management functions
8. Help and support

The features of the web panel are purely based for admins. It uses data related to firebase and the app, as such these features are built to be easily expanded upon allowing for the web panel to be a go to service for web admins.

As is the reason most companies would opt to use these kinds of data management.

OBJECTIVES

- Create a secure login page
- Allow the ability to accept or deny booking requests of users
- Get active route tracking using Google Map API
- Put firebase data into neatly organised tables
- Create an responsive and visual appealing webpage
- To create a web panel that administrators can use to monitor, modify data related to the APP.
- Help and support for admins to understand the functionality of the page, and gain user feedback

FUNCTIONALITY

Login page - Here admins can login using there registered emails and password in the firebase database

Active bookings - The tab initially opened by the website, it allows admins to accept or deny booking requests, while also the requests to be tracked using Google Maps.

Map - The map allows users to get the route data from bookings and the total distance needed to travel to make the trip. But it also allows for adding in points and locations manually to see the route and the distance needed to make that happen.

User database – User IDs and there registered emails can be view in a table format.

User History – User history shows all the past transactions of the Apps users, this is also a table of data, but allow for the search of certain users.

These kinds of functionality provide crucial data visualisation and management admins to quickly and efficiently provide a more in depth expandable service.

USE CASES

User Login

1. User enters in there username and password and clicks login.
2. User is shown the full web panel web page.

Bookings

1. User login.
2. User once logged in is on the booking tab with a booking table.
3. The user enters a user id into the given user id text field.
4. The user click on the accept button to accept the booking.

Tracking

1. User login.
2. Once logged in is on the booking tab with a booking table.
3. The user enters a user id into the given user id text field.
4. The user clicks track to track the route of the transaction
5. The user clicks on the map tab at the top of the page and can see the booking route, coordinates and total distance.

Database Management

1. User login
2. Once logged in the user clicks on the database tab on the top of the page.
3. The user presented with a data table with several text field below.
4. The user enters in data into each text field ID, Car, Name, and Phone and clicks update.
5. The table is now populated with new data for another vehicle.
6. The user then enters in the car type in the Car text field and clicks search.
7. A search table appears and shows results for any car matching the entered car name, including the data entered earlier.

CHALLENGES

- Deciding on a webpage layout
- Creating a login page for administrators
- Learning to use Java Script queries
- Outputting script data into a dynamic tables
- Communication with a NoSQL database
- Creating search queries
- Updating firebase database thorough quires
- Creating a Google Maps API window
- Adding functionality such as tracking to the Google Maps API

When overcoming these challenges the goal in mind was to create something that provided easier data visualization and management than firebase for the app.

But also allowing for additional functionality for app related processes, like route tracking.

DEVELOPMENT

Web Design Layout

When creating the website the initial desire was to have something that looked nice so web template bootstrap was considered, but ultimately deemed unnecessary.
While bootstrap can provide readily available professional level visuals they also come with a level of complication to use effectively making it unnecessarily difficult to present firebase data.
A simple visual website with all the necessary data functionality was decided to be best.

Firebase NoSQL & JavaScript

Assessing the firebase NoSQL Database presented many complications along the way, from deciding to use a NoSQL to MySQL conversion, node.js, and JavaScript quires.
In the end JavaScript quires ended up best suiting the web panel as Firebase does provide some commands on how to do so.
These quires directly visualise the data from the firebase in real time and allow for communication in real time also.

Google Maps API

The Google Maps API allow for use of Google Maps in html and android.
Implementing this API required the use of an API key which required a registration the Google cloud platform and provides very basic Google maps functionality.
Although it had basic functionality to begin with, increased knowledge of the API JavaScript functions were used for additional functionality.
So for better use with the app the functionality was improved to include route display tracking for each booking with a display of total distance.

OVERVIEW

- Web panel only allows secure login
- Manages firebase database
- Visualizes data into different tables and tabs
- Validates, Invalidates customer bookings
- Shows bookings, customers, vehicles, and customer history
- Data can be searched
- Visual represents customer bookings in Google Maps API

The web panel is designed for simplicity and functionality without being jarring to look at for long period of time.

It’s meant to serve as a web portal tool that has all the necessities to make data management very easy for the app data administrators.

ARDUNIO UNO & 1SHEELD

SECTION

ARDUINO UNO & 1SHEELD

SETUP/CONFIGURATION

1. Connect 1Sheeld with Arduino.

    1.1 Physically position 1Sheeld on top of Arduino, connection all I/O pins.

2. Download 1Sheeld app on smartphone.

2.1 Download, open and connect via Bluetooth to 1Sheeld in use.

3. Connect to Power.

    3.1 Using Arduino USB cable connect with USB port on Computer/Socket.

FUNCTIONALITY

- Get GPS location.
- Send GPS coordinates to Firebase.
- Listen for voice recognition.
- Text-to-speech for user interaction.

DEVELOPMENT

HouseKeeping requirements.

Inclusion of the OneSheeld library and all used shields.

Composing http requests for posting coordinates to our database.

Initialising any variables needed.

TransportAI

```
#define CUSTOM_SETTINGS
#define INCLUDE_INTERNET_SHIELD
#define INCLUDE_GPS_SHIELD
#define INCLUDE_TERMINAL_SHIELD
#define INCLUDE_VOICE_RECOGNIZER_SHIELD
#define INCLUDE_TEXT_TO_SPEECH_SHIELD
#define INCLUDE_CLOCK_SHIELD

#include <OneSheeld.h>

HttpRequest latitudeRequest("https://gprojectal.firebaseio.com/location/latitude.json");
HttpRequest longitudeRequest("https://gprojectal.firebaseio.com/location/longitude.json");

float latitude;
float longitude;
int hour,minute,second;
int ledPin = 13;
char* currentTime;
char latitude_char[10];
char longitude_char[10];
const char myCoordinates[]= "what is my location";
const char theTime[] = "what time is it";
```

SETUP FUNCTION

Initialise OneSheeld.

Give permission to query the Clock shield.

Set the OUPUT to pin 13 for LED response.

Set http request responses for success and failure of requests.

Start voice recognition shield.

```
void setup() {
  OneSheeld.begin();
  Clock.queryDateAndTime();
  pinMode(ledPin,OUTPUT);
  latitudeRequest.setOnSuccess(&onSuccess);
  latitudeRequest.getResponse().setOnError(&onResponseError);

  longitudeRequest.setOnSuccess(&onSuccess);
  longitudeRequest.getResponse().setOnError(&onResponseError);

  Internet.setOnError(&onInternetError);
  VoiceRecognition.start();

}
```

LOOP

This is the code that will constantly loop on the Arduino. The main method holding most functionality.

Here we see some code that has been commented out as it was not working correctly, I make requests to pull the data for the current hours and minutes from the clock shield.

Following this, the current longitude and latitude values of our position are pulled from the OneSheeld App.

These float values are then converted to character array to be printed to the terminal.

```
void loop()
{
  //hour = Clock.getHours();
  //minute = Clock.getMinutes();

  latitude = GPS.getLatitude();
  longitude = GPS.getLongitude();

  dtostrf(latitude, 8,4, latitude_char);
  dtostrf(longitude, 8, 4, longitude_char);

  Terminal.println(latitude_char);
  Terminal.println(longitude_char);
```

Next, this character array is added to the http requests individually.

A PUT request to the firebase database is called for both Longitude and Longitude coordinates.

A delay of 5 seconds is added for processing purposes.

```
latitudeRequest.addRawData(latitude_char);
longitudeRequest.addRawData(longitude_char);
Internet.performPut(latitudeRequest);
OneSheeld.delay(5000);
Internet.performPut(longitudeRequest);
// delay to let the server take time to get the data
OneSheeld.delay(5000);
```

The following section of code in the loop is for our special feature.

The idea is for a passenger to have the option of asking for their current location & the current time.

For this only the location is working as seen previously the clock shield requests for the current time did not function currently.

This arduino “listens” for the passenger to ask a question such as “what is my location?”, upon recognising the question, a reply is given telling the passenger their current longitude and latitude position.

```
if(VoiceRecognition.isNewCommandReceived())
{
  if(!strcmp(myCoordinates,VoiceRecognition.getLastCommand()))
  {
    TextToSpeech.say("Your latitude position is");
    delay(1000);
    TextToSpeech.say(latitude_char);
    delay(2000);
    TextToSpeech.say("Your longitude position is");
    delay(1000);
    TextToSpeech.say(longitude_char);
  }
```

ERROR HANDLING

These methods are called if there is a problem with the response from the http requests or if there is an error with the Internet shield.
It determines the error type and displays the response in the terminal.

```
/* Error handling functions. */
void onResponseError(int errorNumber)
{
  /* Print out error Number.*/
  Terminal.print("Response error:");
  switch(errorNumber)
  {
    case INDEX_OUT_OF_BOUNDS: Terminal.println("INDEX_OUT_OF_BOUNDS");break;
    case RESPONSE_CAN_NOT_BE_FOUND: Terminal.println("RESPONSE_CAN_NOT_BE_FOUND");break;
    case HEADER_CAN_NOT_BE_FOUND: Terminal.println("HEADER_CAN_NOT_BE_FOUND");break;
    case NO_ENOUGH_BYTES: Terminal.println("NO_ENOUGH_BYTES");break;
    case REQUEST_HAS_NO_RESPONSE: Terminal.println("REQUEST_HAS_NO_RESPONSE");break;
    case SIZE_OF_REQUEST_CAN_NOT_BE_ZERO: Terminal.println("SIZE_OF_REQUEST_CAN_NOT_BE_ZERO");break;
    case UNSUPPORTED_HTTP_ENTITY: Terminal.println("UNSUPPORTED_HTTP_ENTITY");break;
    case JSON_KEYCHAIN_IS_WRONG: Terminal.println("JSON_KEYCHAIN_IS_WRONG");break;
  }
}

void onInternetError(int requestId, int errorNumber)
{
  /* Print out error Number.*/
  Terminal.print("Request id:");
  Terminal.println(requestId);
  Terminal.print("Internet error:");
  switch(errorNumber)
  {
    case REQUEST_CAN_NOT_BE_FOUND: Terminal.println("REQUEST_CAN_NOT_BE_FOUND");break;
    case NOT_CONNECTED_TO_NETWORK: Terminal.println("NOT_CONNECTED_TO_NETWORK");break;
    case URL_IS_NOT_FOUND: Terminal.println("URL_IS_NOT_FOUND");break;
    case ALREADY_EXECUTING_REQUEST: Terminal.println("ALREADY_EXECUTING_REQUEST");break;
    case URL_IS_WRONG: Terminal.println("URL_IS_WRONG");break;
  }
}
```

SUCCESS RESPONSE

When a successful request is made to the database this function is called and a message is displayed in the Terminal.

```
void onSuccess(HttpResponse & response)
{
    digitalWrite(ledPin,HIGH);
  Terminal.println("Succeeded");
}
```

REFERENCES

http://msdn.microsoft.com/en-us/library/ms130214

http://www.chambers.com.au/glossary/software_requirements_specification.php

http://ac.aup.fr/~croda/SampleStudentsWork/cs348/finalProjectS07/final%20presentation/final/Volere_Specifications_FV.pdf