



Grocery Delivery Management System

ASSIGNMENT 2

Udeshika Dissanayake (s3400652) | COSC2640 Cloud Computing | Semester1 2020 RMIT

Contents

Signed Contribution Agreement.....	2
Links.....	3
Summary.....	4
Introduction.....	5
Related work	6
Software Design/Architecture.....	7
HIGH-LEVEL architectural diagram	7
Main Components of the Application.....	8
<i>Coding Language</i>	8
<i>Google App Engine</i>	8
<i>Google Functions</i>	8
<i>Google Cloud Storage</i>	9
<i>Google Cloud Datastore</i>	9
<i>Google Distance Matrix API</i>	9
<i>Google Geocoding API</i>	9
<i>Google Maps JavaScript API</i>	9
<i>Google Charts</i>	9
<i>Google forms</i>	9
<i>Twilio API for WhatsApp</i>	9
<i>Data Structure used in the Datastore</i>	10
Implementation.....	11
User manual	25
User manual for General Users.....	25
User manual for Service Owner	26
User manual for delivery Partner	30
Improvements	33
References	34
 Figure 1: High Level Architectural Diagram	 7

Signed Contribution Agreement




RMIT Classification: Trusted

Appendix: Student Contribution Agreement

Project title here

Group name here

Grocery Delivery Management System

Student Name: Udeshika Dissanayake	Student Name:
Student ID: s3400652	Student ID:
Contributions: 1. Design & Implementation 2. Report	Contributions: 1. 2.
Contribution Percentage: 100%	Contribution Percentage:
By signing below, I certify all information is true and correct to the best of my knowledge. Signature:  Date: 22/05/2020	By signing below, I certify all information is true and correct to the best of my knowledge. Signature: Date:

Page 7 of 7

Links

URL to the Grocery Delivery Management System Web Application:

<https://cloud-function-test-274101.ts.r.appspot.com/>

Summary

In this assignment, a 'Grocery Delivery Management' system is developed using Google Cloud Platform and some other third-party services. The developed system enables Service Owners (Admis) and multiple Delivery Partners (DPs) to interact automatically on active Delivery Orders.

The developed system possesses a distributed model architecture, where multiple Google Cloud Services such as App Engine, Cloud Functions, Cloud Storage, Cloud Datastore, Google Charts, Google Forms, and Google APIs such as Distance Matrix API, Geocoding API, Maps JavaScript API have been used as building blocks. In addition, Twilio API for WhatsApp has been used for messaging service.

Objectives:

- The main objective of this project is to develop a 'Grocery Delivery Management' system using highly scalable cloud platform with a distributed model concept.

Distinct cloud services have been used to build different modules of the system. For example, GCS App Engine to develop the Web Application, GCS Functions to build serverless event-driven modules, GCS Datastore for storing data records, Google APIs for calculating distance & duration between two locations, Twilio WhatsApp API for sending automatic WhatsApp messages, ect. Each of these modules synchronously communicates over the network by passing messages (http/API) in order to maintain healthy operations of the system.

Such distributed system architecture enables the potential of improving the performance of each module independently without impacting the operations of the other modules. Also, such systems would cope with future growth and demand requirements by seamlessly scale over cloud infrastructure without any latency.

Introduction

Online Grocery Shopping is becoming more and more popular with recent social distancing measures imposed around curbing COVID19 spread. Currently, the home delivery services for grocery shopping is not readily available except for large supermarket chains such as Coles and Woolworths. The main intention of this project is to develop a '**Grocery Delivery Management System**' to manage the Delivery Orders of small Groceries. The concept has lot of similarities to 'Uber Eats' and 'Menulog', where third-party Delivery Partners are assigned delivery orders. The developed system is capable to manage multiple Delivery Partners and cater for large number of Delivery Orders at real time using highly scalable cloud services. The owner of the 'Grocery Delivery Management System' ---The Admin--- will oversee the operations around Orders and DPs.

The Admin has the privilege to register/de-register new Delivery Partners (DPs) to the system. When a new Order is received, the Admin will have to assign the Order to a DP and incept the process by copying the Order file (.csv file) to cloud storage. The Order file consist of mandatory details on the order such as pick-up address, drop-off address, date, time, DP Name, DP mobile no., etc. These details will be automatically analyzed by the system and obtains the distance & duration between pick-up and drop-off locations. Also, delivery fee is calculated based on the distance. Finally, a WhatsApp message is sent by the system to the corresponding DP stating the details of the Delivery Order.

A Web Application is developed for both Admins and DPs to log and view their orders under their profiles. Also, interactive dashboards have been developed for Admins and DPs to view their earnings, distance traveled, etc. For DPs, an interactive map shows the drop-off location for each Orders.

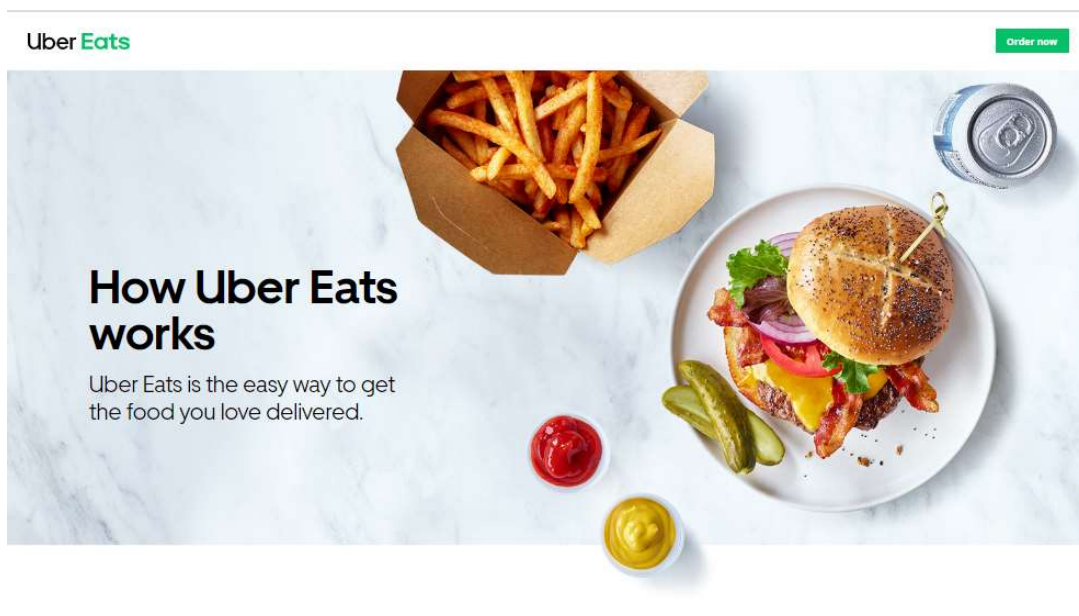
The developed system will be quite useful in near future (particularly around implementing new social distancing regulations) for small and medium scale Grocery owners to get home delivery service done via third-party DPs.

As a future development, a delivery pooling concept is proposed where multiple deliveries could be bundle together based on pick-up drop-off locations, and this could potentially minimize the average cost per delivery in turn per delivery carbon emission could also be mitigated, hence benefitted to the environment.

Related work

Home delivery services for online purchases are not a new thing: ebay, AliExpress, and Amazon have been leading in this field for past decade. However, when it comes to local deliveries, UberEats, Menulog, and Coles/Woolworths Delivery have been dominating. The system proposed under this work is much closer to UberEats and Menulog where delivery partners are third-party members who registered for the platform. Out of the pool of delivery partners one will be chosen to perform a delivery.

Uber Eats



Software Design/Architecture

HIGH-LEVEL architectural diagram

Below high-level architectural diagram shows the overview of the application. The entire application system is made up of several Google Cloud Services (GCS) and third-party cloud service components. The arrows with numbers represent the communications between each cloud services and their sequencing.

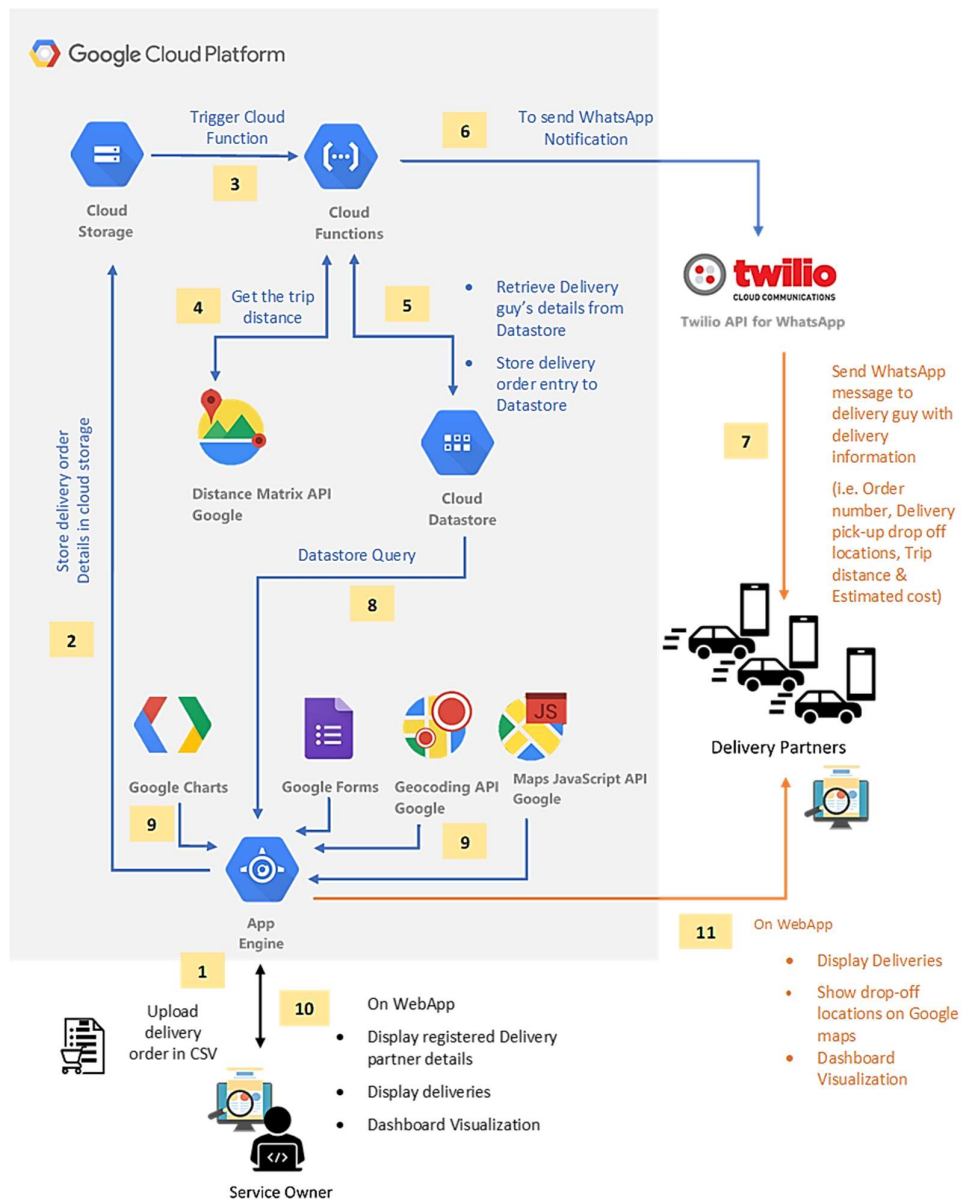


Figure 1: High Level Architectural Diagram

Main Components of the Application

The Google Cloud Services (GCS) has been the core of the developed application. Almost all the services (except for WhatsApp messaging service) are developed on GCS and their APIs. Several coding languages have been used through out this project as outlined below:

Coding Language

- PHP – Web Application back-end devs
- HTML/CSS – Web Application front-end devs
- JavaScript – Web Application back-end devs (Maps and Charts)
- Node Js – Google Functions

Below are the GCS components and services used in the applications:

Google App Engine

The Web Applications is built and hosted on GCS App Engine. The front end of the Web App is designed using HTMS/CSS while the back-end services are built using PHP and JavaScript languages.

Google Functions

GCS Functions are used to implement several event-driven serverless compute modules needed in the application. These modules are highly scalable and implemented to communicate with other services as needed. These modules are either get triggered by http calls or Storage Bucket events. Below are the GCS Functions developed in this project:

readCSVinStorage	Trigger: Bucket Action: Read the content of saved csv order file
driveDistance	Trigger: http Action: Obtain the distance/duration between pick-up and drop-off locations (use Google Distance Matrix API)
Function_Datastore	Trigger: http Action: Save data record to Google Datastore
sendWhatsapp	Trigger: http Action: Send a WhatsApp messages to DPs (use Twilio API for WhatsApp Messaging)

Google Cloud Storage

GCS Storage is used as an Infrastructure as a Service (IaaS) to save all the delivery orders uploaded by the Admins. GCS Storage is highly scalable and possesses advanced security and sharing capabilities needed for the application.

Google Cloud Datastore

All the structured data in this project has been recorded, retrieved, and queried in the GCS Datastore. This includes Delivery Partners login details, Admins login details, and Delivery Order details. GCS Datastore is a highly scalable NoSQL database designed to automatically scale to very large datasets maintaining the minimum latency needed for the application.

Google Distance Matrix API

Used to get the drive distance and travel time estimation between pick-up and drop-off locations.

Google Geocoding API

Used to convert Street Address to Geocoordinates (i.e. Lat, and Lon).

Google Maps JavaScript API

Used to show the drop-off locations on Google Maps.

Google Charts

Used for dashboard visualization of delivery partners data in the Delivery Partner's and the Admins login profiles.

Google Forms

Used in 'Contact Us' page for anyone to contact the Admin. Google form input will be automatically saved in Google Sheet and send notification emails to Admin.

Twilio API for WhatsApp

Messaging service to the delivery partners to inform the delivery information such as order number, addresses for pick-up drop-off locations, trip distance, delivery fee and contact number to confirm the delivery order.

Data Structure used in the Datastore

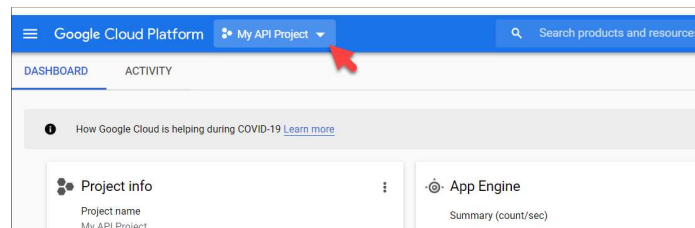
Below are the Datastore kinds implemented in the project:

Admin_users	: keeps login details of Admins
DP_users	: keeps login details of DPs
Delivery_data	: keeps records of each delivery orders including pick-up locations, drop-off locations, date, time, distance, duration, fee, DP name, Contact Number, etc.

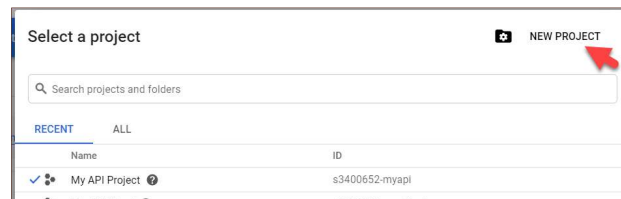
Implementation

Setting-up Google App Engine Project

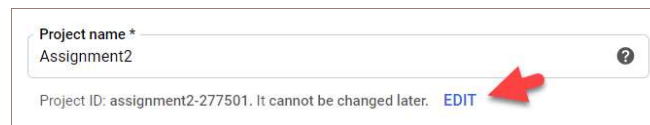
1. Go to the link: console.cloud.google.com and sign-in to Google app Engine console using the personal Google account credential. Once logged in, you will see the Dashboard for the current project.
2. In the top menu of the console click the 'Select a Project' button.



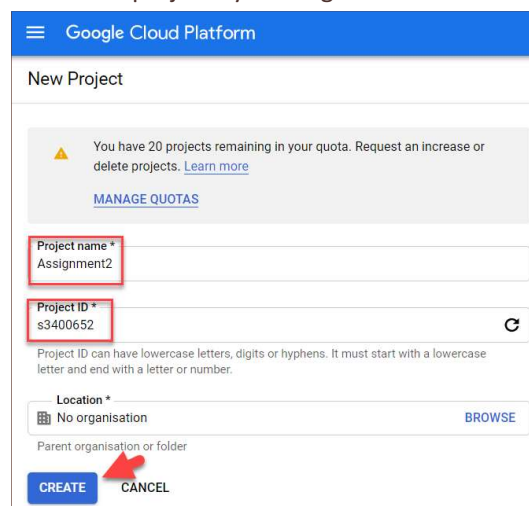
3. Select NEW PROJECT from the Select Project window to create new project.



4. Type a suitable project name and modify the project id by clicking the EDIT button.

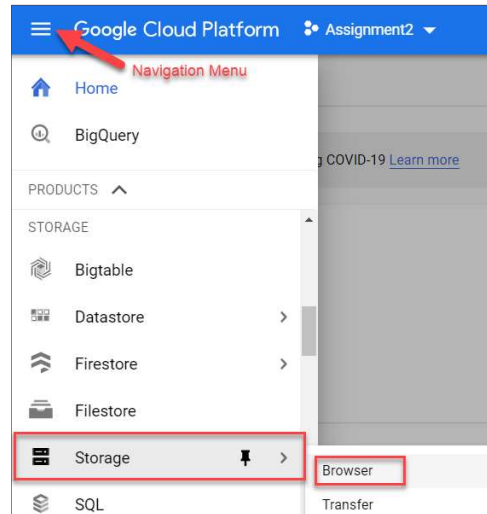


5. Type suitable project ID in the project ID edit window.
6. Create the project by clicking the CREATE button.

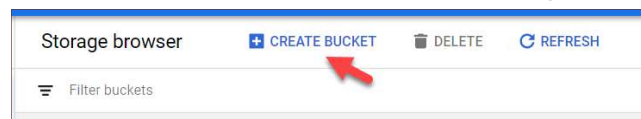


Setting-up Google Cloud Storage:

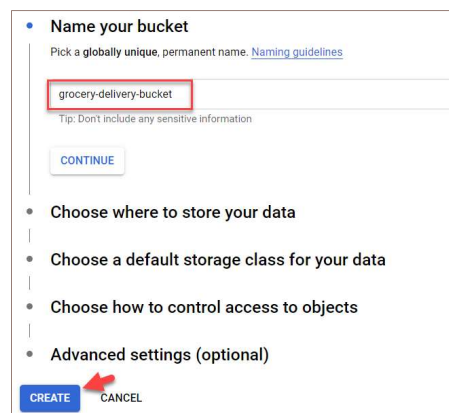
1. Open the dashboard of your current project.
2. In the top left of the console, select Navigation Menu, select Storage>Browser under Storage section.



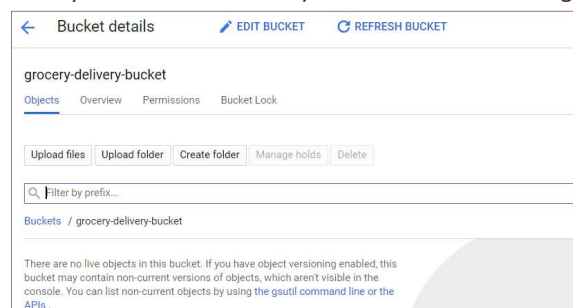
3. Click on the CREATE BUCKET button in the Storage browser window.



4. Select a name for the bucket and click on the CREATE button.



5. Once you created a bucket you will see like following screen.

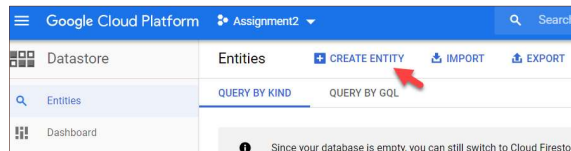


Setting-up Google Cloud Datastore:

1. Open the dashboard of your current project.
2. In the top left of the console, select Navigation Menu, select Datastore>Entities under STORAGE section.



3. Click on the CREATE ENTITY to create entities.



4. To create the entities for Admin Users, type suitable name for the 'Kind' (i.e. Admin_users), select 'Custom name' as the 'Key identifier' and type a unique ID (i.e. A1002) for the Admin-users shown in the figure below. Then click on ADD PROPERTY.

A screenshot of the 'Create an entity' form. The form has fields for 'Namespace' (default), 'Kind' (Admin_users), 'Key identifier' (Custom name), and 'Custom name *' (A1002). Below these fields is a section for 'SPECIFY PARENT' and a 'Properties' section. The 'ADD PROPERTY' button in the Properties section is highlighted with a red arrow.

5. Add 'name' to the new property window with Type String along with the value and click on DONE.

A screenshot of the 'New property' form. The form has fields for 'Name *' (name), 'Type' (String), and 'Value' (UdeshikaWD). There is a checkbox for 'Index this property' which is checked. The 'DONE' button is highlighted with a red arrow.

- Similarly, add the password property with Type String along with the value (i.e. shika123)
- Then click on the CREATE button

- Once you created the Admin users object in the Datastore you will see like following screen.

Name/ID	name	password
name=A1002	UdeshikaWD	shika123

- Similarly, create the delivery-partner users object as shown in the picture below.

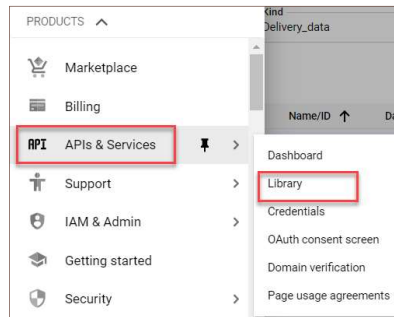
Name/ID	name	password
name=D1001	Bathiya Santhush	che123
name=D1002	Mahinda Rajapakshe	mahi123

- Similarly, create the delivery-data object with indicative values for each property as shown in the picture below. This object will be updated automatically upon new delivery orders are uploaded to the google cloud storage bucket.

Name/ID	Date	DelNo	Distance	Dropoff	Duration	Fee	Name	Pickup	Time	tel
name=G1002	25/6/2020	D0001	10	test drop	10	8	test name	test pick up	15:30	430456702

Setting-up Google Distance Matrix API

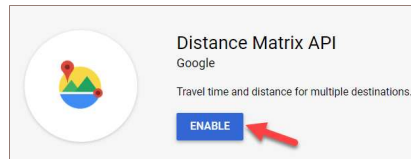
1. Open the dashboard of your current project.
2. In the top left of the console, select Navigation Menu, select APIs & Services>Library under PRODUCTS section.



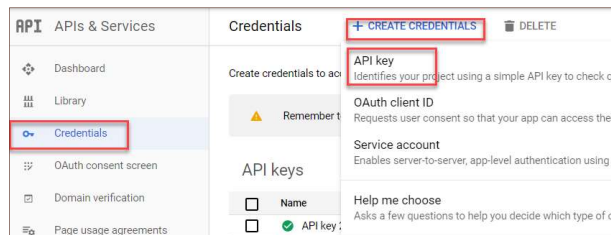
3. Select Maps from the left side category list.



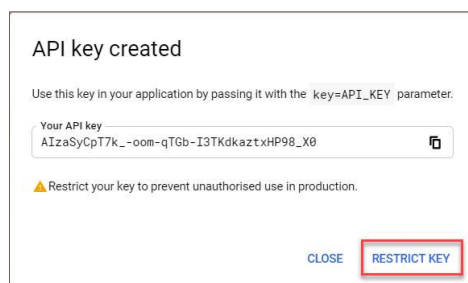
4. Select Distance Matrix API and click on the ENABLE.



5. to create a key for the API, Select Credential and Click on CREATE CREDENTIALS from to top menu and Select 'API Key' from the drop-down menu.



6. Click on RESTRICT KEY



7. Select the 'Restrict Key' and check Distance Matrix API from the list and SAVE

Restrict and rename API key

REGENERATE KEY DELETE

An application restriction controls which websites, IP addresses or applications can use your API key. You can set one application restriction per key.

☒ None
☐ HTTP referrers (websites)
☐ IP addresses (web servers, cron jobs, etc.)
☐ Android apps
☐ iOS apps

API restrictions

API restrictions specify the enabled APIs that this key can call

☐ Don't restrict key
 This key can call any API
☒ Restrict key

Distance Matrix API

Distance Matrix API

Note: It may take up to 5 minutes for settings to take effect.

SAVE CANCEL

8. Newly created Key will be appeared in the Credential Dashboard.

API keys				
<input type="checkbox"/>	Name ↑	Creation date ↓	Restrictions	Key
<input checked="" type="checkbox"/>	API key 3	18 May 2020	Distance Matrix API	AIzaSyCpT7...ztxHP98_X0

Setting-up Google Geocoding API

1. Follow the same steps described in the 'Setting-up Google Distance Matrix API' section and select Google Geocoding API for the step 4.

Setting-up Google Maps JavaScript API

1. Follow the same steps described in the 'Setting-up Google Distance Matrix API' section and select Google Maps JavaScript API for the step 4.

Setting-up Twilio API for WhatsApp

Twilio API for WhatsApp is used as the message sending engine of this project. Through a Google Function, Twilio WhatsApp API could be triggered in order to send messages to the DPs notifying newly created delivery Orders. Below are the steps in configuring Twilio API for WhatsApp

1. Sing-up for a Twilio Account - <https://www.twilio.com/try-twilio>
2. Activate Twilio Sandbox for WhatsApp. This allows to prototype messaging using the shared Twilio phone number to add sandbox users. The sandbox users can be added by sending whatsapp message to provide number with specific code word. In this project below two mobile numbers have been added.

Dashboard

Learn & Build

SMS

WhatsApp

Beta

Learn

Sandbox

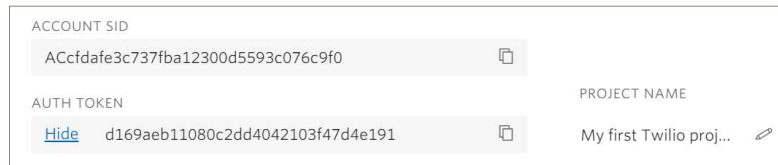
Senders

Sandbox Participants

Invite your friends to your Sandbox. Ask them to send a WhatsApp message to +1 415 523 8886 with code **Join wrapped-first**.

USERID
whatsapp:+61430456702
whatsapp:+61400946591

3. Get the Twilio account credentials by logging into Twilio Console/Dashboard:



ACCOUNT SID
ACcfdafe3c737fba12300d5593c076c9f0

AUTH TOKEN
d169aeb11080c2dd4042103f47d4e191

PROJECT NAME
My first Twilio proj...

4. Finally, setup the Node.js development environment in order to implement Google Function to send WhatsApp message. Install dependency needs for Cloud Function at the console prompt:
`npm install --save twilio`

5. Update the package.json file with Twilio dependencies:



```
{
  "name": "sample-http",
  "version": "0.0.1",
  "dependencies": {
    "request": "^2.81.0",
    "twilio": "^3.30.0"
  }
}
```

6. Use Twilio Account credentials obtained in step 3, and include Twilio module in the Node.js implementation of corresponding Google Function.

```
const client = require('twilio')(accountSid, authToken);
```

Google Functions

Four different Google Functions have been developed as a part of this project. Those functions and their triggers and actions are as follows:

readCSVinStorage	Trigger: Bucket Action: Read the content of saved csv order file
driveDistance	Trigger: http Action: Obtain the distance/duration between pick-up and drop-off locations (use Google Distance Matrix API)
Function_Datastore	Trigger: http Action: Save data record to Google Datastore
sendWhatsapp	Trigger: http Action: Send a WhatsApp messages to DPs (use Twilio API for WhatsApp Messaging)

Setting-up Google Function - readCSVInStorage

The expectation is to build a function that triggers when a new .csv file is copied to a Cloud Storage Bucket (mentioned in previous section as - grocerydelivery-bucket). Thereafter the function should read the content of the .csv file and pass that information to other subsequent functions to do the needful.

1. Create the GCP Cloud Function. Go to 'Cloud Functions' section from Navigation Menu and select "Create Functions". Set the details as follows:

The screenshot shows the 'Create function' form in the Google Cloud Platform console. The form is titled 'Create function' and has a 'cloud function Test' button. The fields are: Name (readCSVInStorage), Memory allocated (256 MB), Trigger (Cloud Storage), Event type (Finalize/Create), Bucket (grocerydelivery-bucket), Source code (Inline editor), and Runtime (Node.js 8).

2. Set the dependencies in the package.json file as follows:

```
{
  "dependencies": {
    "@google-cloud/debug-agent": "^4.0.0",
    "@google-cloud/storage": "^2.5.0",
    "axios": "^0.19.2",
    "escape-html": "^1.0.3",
    "googleapis": "^40.0.0",
    "node-fetch": "^2.6.0",
    "pug": "^2.0.3",
    "request": "^2.81.0",
    "supertest": "^4.0.2"
  },
}
```

3. Install the client library of `@google-cloud/storage` through node package manager (npm) in the cloud console,
`npm install @google-cloud/storage`
4. Include the required modules and implement asynchronous function to read the .csv file.

```
'use strict';
const {google} = require("googleapis");
const {Storage} = require("@google-cloud/storage")
const request = require('request');
```

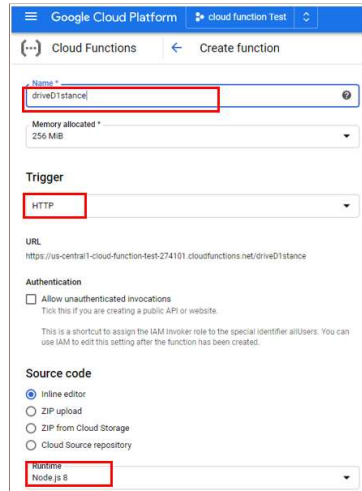
```
exports.readCSVInStorage = async(data, context, callback) => {
```

5. Deploy the function as,
`gcloud functions deploy readCSVInStorage --runtime nodejs8 --trigger-resource grocerydelivery-bucket --trigger-event google.storage.object.finalize`

Setting-up Google Function - driveDistance

This Cloud function is triggered by http call and will get the distance and duration between pick-up and drop-off locations through Google Distance Matrix API. The pick-up and drop-off locations are passed into the function as parameters.

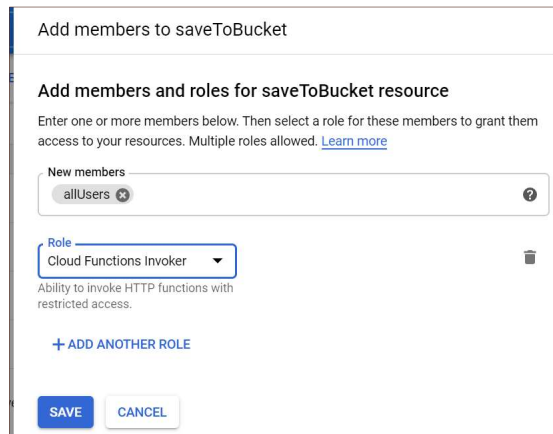
1. Create the GCP Cloud Function. Go to 'Cloud Functions' section from Navigation Menu and select "Create Functions". Set the details as follows:



2. Set the dependencies in the package.json file as below:

```
{
  "name": "sample-http",
  "version": "0.0.1",
  "dependencies": {
    "request": "^2.81.0"
  }
}
```

3. Build the function in index.js and deploy by running below code in cloud console,
`gcloud functions deploy driveDistance --runtime nodejs8 --trigger-http --allow-unauthenticated`
4. Finally, allow unauthenticated invocation access of the function by adding the allUsers member type to the function as, Go to Functions Page > check the box next to Function > Show Info Panel > Permission > Add member > New Member > type allUsers > Select Cloud Function-Cloud Function Invoker from drop down>Save



Setting-up Google Function - FunctionDataStore

This Cloud function is triggered by http call and will save Order records in to Cloud Datastore (created in a previous section as - Delivery_data). The `@google-cloud/datastore` module is needed. Create Node.js 8 Function to be triggered by http call as explain in the previous section.

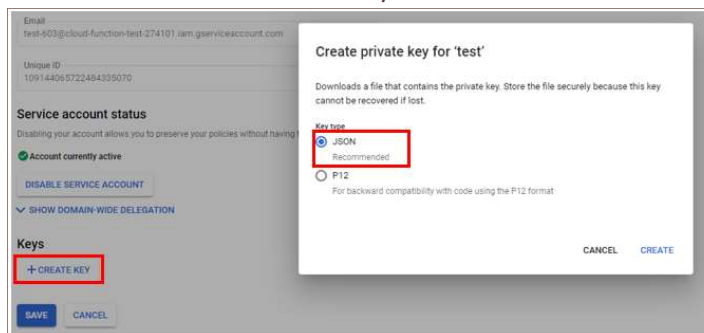
1. Install the client library of `@google-cloud/datastore` through node package manager (npm) in the cloud console,
`npm install @google-cloud/datastore`
2. Set the dependencies in the package.json file as below:

```
{
  "name": "sample-http",
  "version": "0.0.1",
  "dependencies": {
    "@google-cloud/datastore": "1.4.1"
  }
}
```

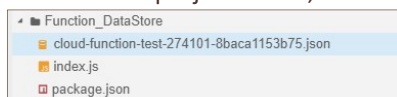
3. Get the Datastore credentials. Go to Navigation Menu > APIs & Services > Credentials > Create Credentials > Service Account > Put name > select under Role – Datastore -> Cloud Datastore User. You should see the Service Account is created as below under API Services > Credentials section:

Service Accounts	
<input type="checkbox"/> Email	Name ↑
<input type="checkbox"/> cloud-function-test-274101@appspot.gserviceaccount.com	App Engine default service account
<input type="checkbox"/> cf-datastore@cloud-function-test-274101.iam.gserviceaccount.com	CF-DataStore
<input type="checkbox"/> 947781351788-compute@developer.gserviceaccount.com	Compute Engine default service account
<input type="checkbox"/> gcp-storage-upload@cloud-function-test-274101.iam.gserviceaccount.com	gcp-storage-upload
<input type="checkbox"/> test-603@cloud-function-test-274101.iam.gserviceaccount.com	test

4. Download the JSON formatted key file.



5. Save it in the project folder,



6. Include the required modules, set the JSON key file, and set the projectID in the index.js file,

```
const request = require('request');

const Datastore = require('@google-cloud/datastore');
const datastore = new Datastore({
  projectId: 'cloud-function-test-274101',
  keyFilename: 'cloud-function-test-274101-8baca1153b75.json'
});
```

7. Deploy the function as,

```
gcloud functions deploy Function_DataStore --runtime nodejs8 --trigger-http --allow-unauthenticated
```

Setting-up Google Function - sendWhatsapp

This Cloud function is triggered by http call and will send WhatsApp messages to Delivery Partners as needed. The **twilio** module is needed. Create Node.js 8 Function to be triggered by http call as explain in the previous section.

1. Install client library twilio through npm,
`npm install --save twilio`
2. Get Twilio Account credential by logging in the Twilio account (explained in detail in one of the previous sections)
3. Include the required modules in package.json file,

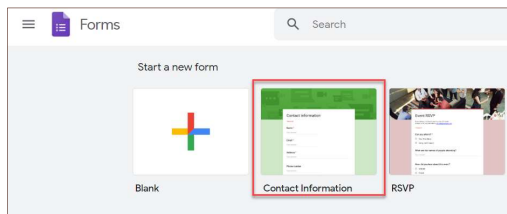
```
{
  "name": "sample-http",
  "version": "0.0.1",
  "dependencies": {
    "request": "^2.81.0",
    "twilio": "^3.30.0"
  }
}
```

4. Deploy the function as,

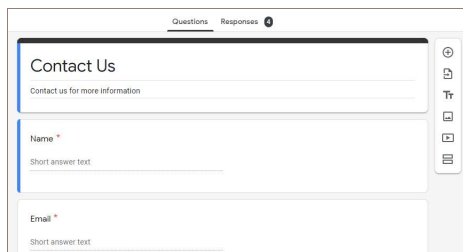
```
gcloud functions deploy sendWhatsapp --runtime nodejs8 --trigger-http --allow-unauthenticated
```

Setting-up Google forms

1. Go to [google forms](https://forms.google.com) and select the template for 'Contact information'.



2. Edit the template with the required information and style



3. Update settings as per the below pictures and click on Save.

The left screenshot shows the 'Settings' dialog box with the 'General' tab selected. It includes options for 'Collect email addresses', 'Response receipts', 'Requires sign in', 'Limit to 1 response', and 'Respondents can'. The right screenshot shows the 'Settings' dialog box with the 'Presentation' tab selected. It includes options for 'Show progress bar', 'Shuffle question order', 'Show link to submit another response', and a 'Confirmation message' field. Both screenshots have a red arrow pointing to the 'Save' button.

4. Go to Responses tab, select 'Accepting responses' and click on 'More' at top.

The screenshot shows the 'Responses' tab in Google Forms. The 'Responses' tab is selected. The 'Accepting responses' toggle is turned on. A red arrow points to the 'More' button (three dots) at the top right.

5. To receive email notification from google form submissions, Enable 'Get email notifications for new responses'.

The screenshot shows the 'Responses' tab in Google Forms. The 'Get email notifications for new responses' toggle is turned on.

6. Click on the 'Create Spreadsheet' button.

The screenshot shows the 'Responses' tab in Google Forms. The 'Create Spreadsheet' button is highlighted with a red arrow.

7. To see all responses in a spreadsheet, select 'Create a new spreadsheet' and click on Create button. This will create a new spreadsheet in google sheets and it will be updated automatically upon response arrival.

The screenshot shows the 'Select response destination' dialog box. The 'Create a new spreadsheet' option is selected. The 'Create' button is highlighted with a red arrow.

8. To view responses spreadsheet, click on the 'view responses in sheets'.

The screenshot shows the 'Responses' tab in Google Forms. The 'View responses in Sheets' button is highlighted with a red arrow.

Google charts

JavaScript has been embedded in the web page HTML code with following components.

1. Load some Google Chart libraries
2. Define the data to be visualized
3. Customize the chart
4. Create a chart object with an id.
5. Create a <div> with that id to display the Google Chart at the end of the web page HTML code.

```
<html>
  <head>
    <script type="text/javascript"
      src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">

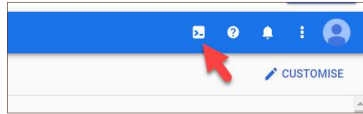
      // Load the Visualization API and the corechart package.
      google.charts.load('current', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.charts.setOnLoadCallback(drawChart);

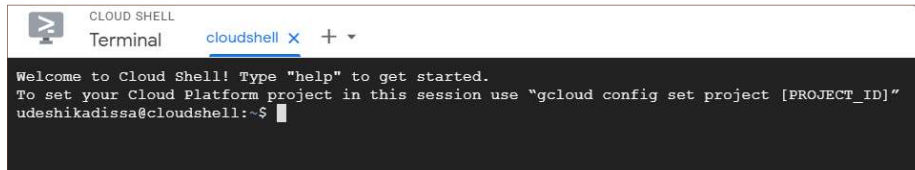
      function drawChart() {
        // define the data table
        var data = new google.visualization.DataTable();
        // Set chart options
        var options = {};
        // Instantiate and draw our chart, passing in some options.
        var chart = new google.visualization.PieChart
          (document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div"></div>
  </body>
</html>
```


Deploying the application

1. In the top menu of the console select 'Activate Cloud Shell'.



2. Cloud Shell will be opened.



3. Set your current project to your project id (ie s3400652) using the following command.

```
cloud config set project s3400652
```

4. Go to the directory that contains the project codes using the following command.

```
Cd Ass2/
```

5. Deploy the application using following command.

```
gcloud app deploy
```

6. Application can be view on the web browser using following command

```
gcloud app browse
```

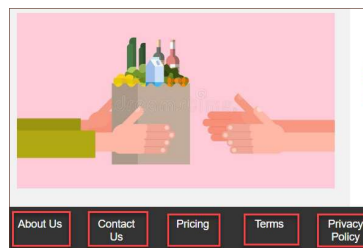
User manual

User manual for General Users

1. Go to the link: <https://cloud-function-test-274101.ts.r.appspot.com/>
2. From the menu on top and bottom of the window, click on About button to explore more about us and click on Contact Us button to Contact us.

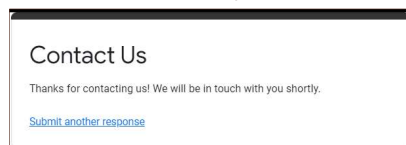


3. From the menu on bottom of the window, click on Pricing button to explore more about our pricing and click on Terms and Privacy Policy buttons to get more information on pricing, terms and conditions.



4. To contact us, go click on the Contact Us button as shown in step1, fill the required fields and your query and click on the submit button.

5. You will get following message on successful form submission. Click on the link “Submit another response” to send another query.



6. Click on the Home button on the top left corner any time to navigate to the home page.

User manual for Service Owner

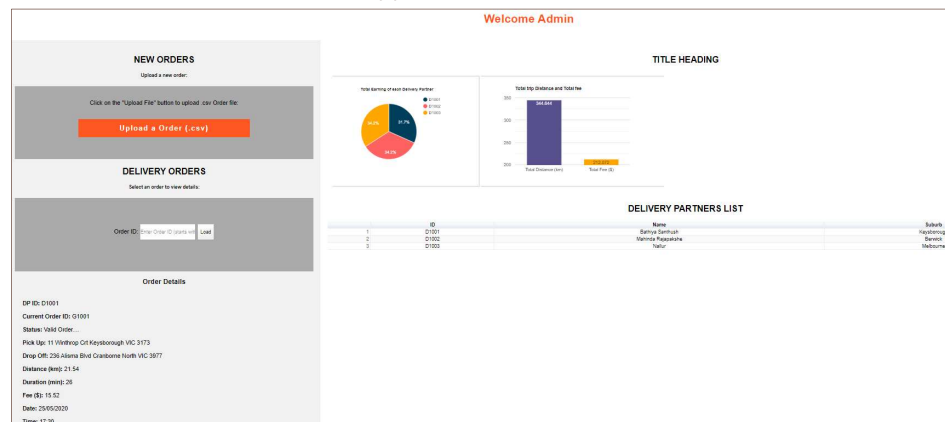
Log-in to Web App

1. Go to the link: <https://cloud-function-test-274101.ts.r.appspot.com/>
2. Click on the Login button on the top right corner and Select Admin.



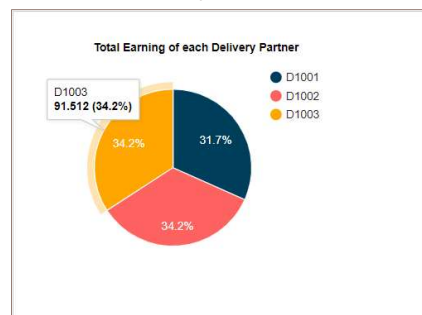
3. Type user ID and the password and click on Login button.
(e.g. ID: A1002, Password: shika123)

4. Admin user's dashboard will be appeared.



Visualization of Deliveries

1. First chart represents the total earning (in dollars) of each Delivery partner registered. Hover over each portion to view the amount.



2. Second chart represents the total distance in km and total Fee for Delivery partners.



Registered Delivery partners

1. Delivery partner list shows all the DPs registered with their ID, name and their suburb.

DELIVERY PARTNERS LIST		
ID	Name	Suburb
D1001	Bathiya Santhush	Keysborough
D1002	Mahinda Rajapakshe	Berwick
D1003	Nallur	Melbourne

Query Order details

1. To get information on any existing orders, scroll to 'DELIVERY ORDERS' Section at left side pan and type an order number (e.g. G5001) and click on Load button.

The screenshot shows a section titled "DELIVERY ORDERS" with the instruction "Select an order to view details:". Below this is a search bar containing "Order ID: G5001" and a "Load" button. A red arrow points to the "Load" button.

2. Order details will appear as below.

The screenshot shows the "DELIVERY ORDERS" section with the instruction "Select an order to view details:". Below this is a search bar containing "Order ID: Enter Order ID (starts with G)" and a "Load" button. Below the search bar is a section titled "Order Details" containing the following information:

- DP ID: D1002
- Current Order ID: G5001
- Status: Valid Order....
- Pick Up: 236 Alisma Blvd Cranbourne North VIC 3977
- Drop Off: 17 Harper St Werribee VIC 3030
- Distance (km): 81.512
- Duration (min): 66.2
- Fee (\$): 45.756
- Date: 29/05/2020
- Time: 18.30

Create a new order for a delivery partner

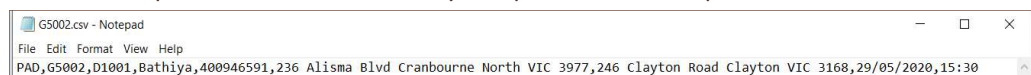
1. Rename the .csv template with next available order number (e.g. G5002).



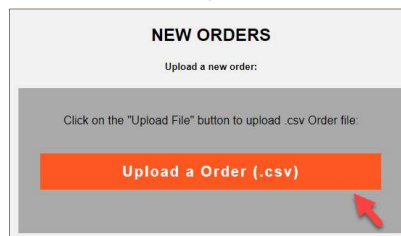
2. Update the G5002.csv with new delivery order information (update all the highlighted fields below without putting any commas for the pick-up and drop off locations)

PAD,Order No,Delivery Partner ID,Delivery partner Name,Delivery Partner mobile number,
Pick-up address,drop-off address,Delivery Date,Delivery time

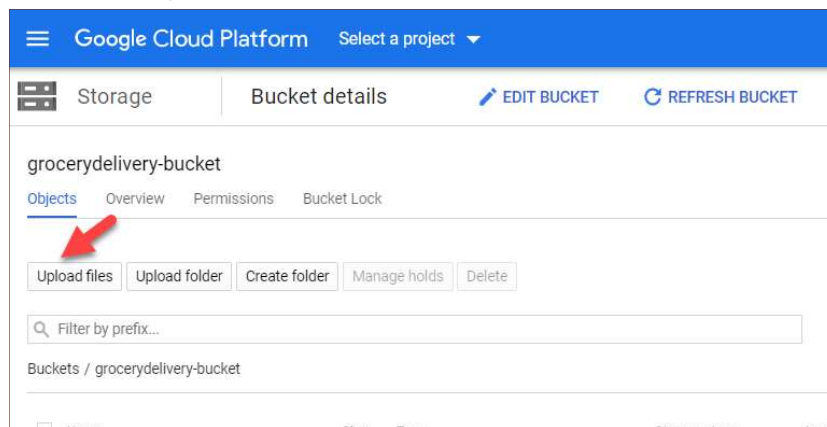
3. Refer the snapshot below from correctly completed order template



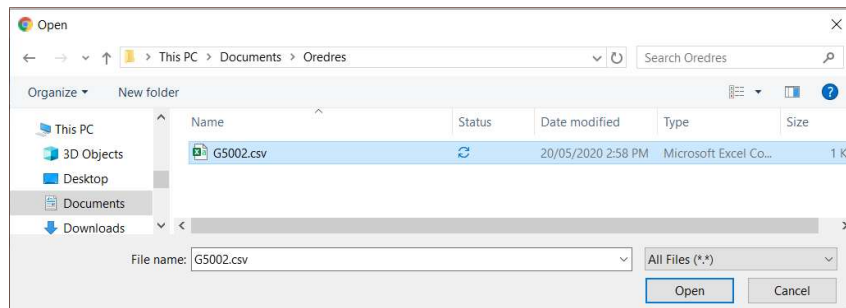
4. Once the csv file is ready, navigate to the 'New Orders' section and 'Upload New Order' Button. This will open a new window.



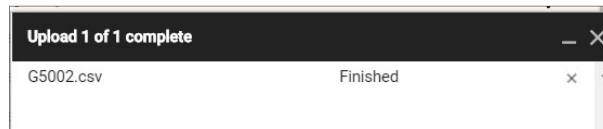
5. Click on the 'Upload files' button.



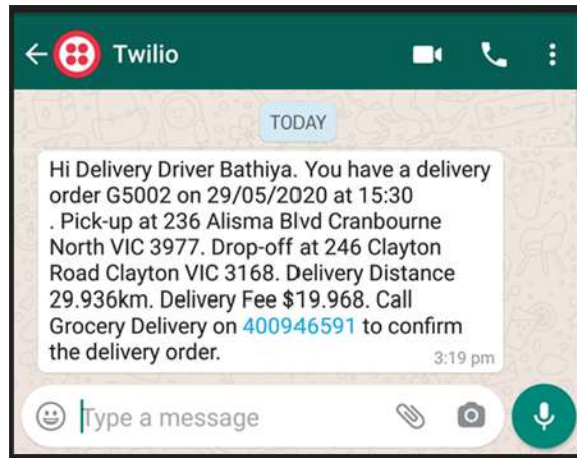
6. Brows the G5002.csv file, select and click on the Open button.



- Following message window will appear upon successful upload.

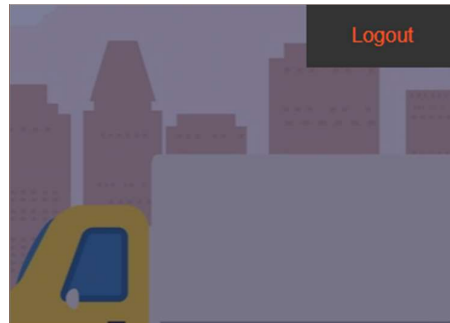


- This will send following whatsapp message to the delivery partner.



Log-out from Web App

- Click on the Login button on the top right corner



User manual for delivery Partner

Delivery order message

1. Once an order is assigned, following whatsapp message will be received to the delivery partner with following information.

Order No: G5002

Delivery Date: 29/05/2020

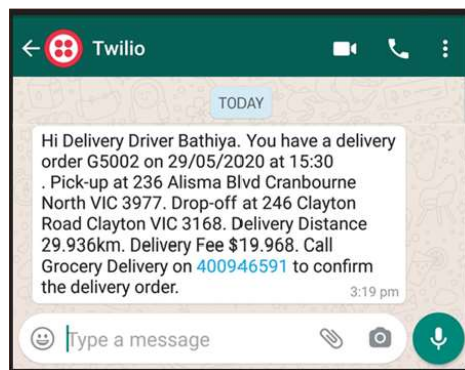
Delivery pick-up time: 15:30

Pick-up location: 236 Alisma Blvd, Cranbourne North, VIC 3977

Drop-Off Location: 246 Clayton Road, Clayton, VIC 3168

Delivery distance 29.93km

Delivery Fee: \$19.97



2. To confirm the order delivery, Delivery partner need to call the given number (i.e. 0400946591)

Web Application for Delivery Partners

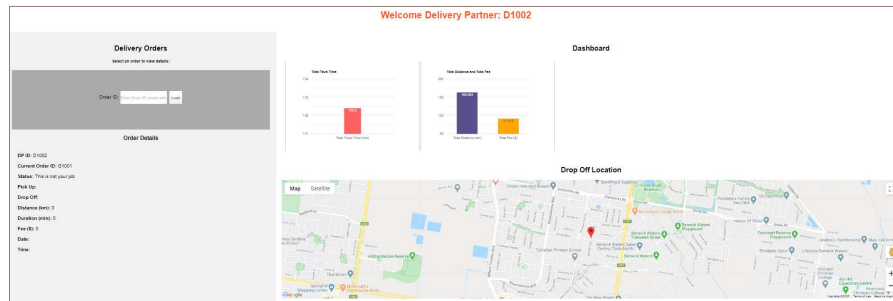
Log-in to Web App

1. Go to the link: <https://cloud-function-test-274101.ts.r.appspot.com/>
2. Click on the Login button on the top right corner and Select 'Delivery Partner'.



- Type user ID and the password and click on Login button.
(e.g. ID: D1002, Password: mahi123)

- Delivery Partner's dashboard will be appeared.



Visualization of Deliveries

- 'Total Travel Time' chart represents the total travel time (in km) of the delivery partner.



- Second chart represents the total distance in km and total Fee of the delivery partner.




Query Order details

- To get information on any existing orders, scroll to 'DELIVERY ORDERS' Section at left side pan and type an order number (e.g. G5001) and click on Load button.

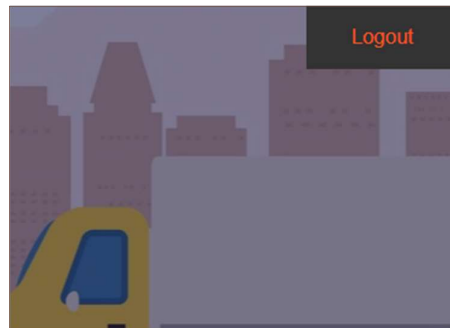
2. Order details and the drop-off location will appear on the bottom of the web page as below.

Order Details
DP ID: D1002
Current Order ID: G5001
Status: Valid Order...
Pick Up: 236 Alisma Blvd Cranbourne North VIC 3977
Drop Off: 17 Harper St Werribee VIC 3030
Distance (km): 81.512
Duration (min): 66.2
Fee (\$): 45.756
Date: 29/05/2020
Time: 18:30

Drop Off Location


Log-out from Web App

1. Click on the Login button on the top right corner



Improvements

Below listed future improvements are identified at the time of writing this report:

- Two-way authentication for Message service
- Interface to add DPs and Admins
- Change password and password recovery for DPs and Admins
- Delivery pooling algorithm to minimize the average delivery cost

References

- [1] "Stack Overflow," Newest Questions, 2020. [Online]. Available: <https://stackoverflow.com/questions>. [Accessed 22 May 2020].
- [2] "W3schools.com," W3schools Online Web Tutorials, 2020. [Online]. Available: <https://www.w3schools.com/>. [Accessed 16 May 2020].