

Road Surface Estimation
FIT3036 – Computer Science Project
XI YU LIEW
26227592
Assoc. Prof. Andrew P. Paplinski

Abstract

This report provides an overview of how a computer science project was done starting from researching towards until the project meet the requirements. The project objective is to calculate the total surface area of the road within a designated area in Google Map. Formula is included to assists to calculate the total surface area.

First, the report will talk about the early researches about the topic, project risks that might occurs in the project and actually occurred in the project, resources requirements and project timeline. Next is the methodology used in the project and overall design of the project. Diagrams are also included to explain more details about the design and the functions of the project.

The type of result that obtained are recorded down as well and assisted with diagrams as well for better understanding. It then describes the input and out of the program along with the complexity of the program. Furthermore, a detailed analysis is done to prove that the results that obtained are actually accurate and able to meet the requirements of the projects.

Moreover, the report also discusses about the future work of the project as in what could've be improved and the conclusion of the project. Followed by the bibliography used throughout the project. Lastly, there are instructions on how to use the application with diagram attached.

Keyword: surface area, road, Google Map, formula

Table of Contents

1.0 Introduction	1
2.0 Background	2
2.1 Review of academia literature	2
2.2 Project Risks	5
2.3 Resources Requirement.....	5
2.4 Project Timeline.....	6
3.0 Method	8
3.1 Methodology used	8
3.2 Internal Design	10
3.3 Software architecture	11
3.4 Functions and Algorithms	12
4.0 Result	14
4.1 Result obtained	14
4.2 Input / Output.....	16
4.3 Performance	16
5.0 Analysis and discussion.....	17
5.1 Result Analysis.....	17
6.0 Future work	20
7.0 Conclusion.....	21
8.0 Bibliography.....	22
9.0 Appendices	23
9.1 Production and Deployment	23
9.2 User Interface	24
9.3 Internal testing procedures.....	27

1.0 Introduction

The company secured a contract for a local council in Victoria re-surface roads in a designated square kilometer area. The project objective is to calculate the total area of roads in the nominated square kilometer. There are a few requirements for the project such as Google maps or related, high level programming code and an elegant Graphical User Interface (GUI).

The functional requirements of this project are that the program should have access to the Google Map and enable user to nominate a square. The program should be able to generate Static Map URL and send it to the server for further processing. The static image must be able to differentiate roads and non-roads. Correct total surface area of the road should be displayed in the client side for the user to see.

Non-functional requirement of the program is can be run by any user without any difficulty. It is also expected to be able to run multiple times and displays different total surface area. The only project constraints of this project is that the Google Map database are outdated and the Google Map is not up to date which shows different data from the actual surface area.

2.0 Background

2.1 Review of academia literature

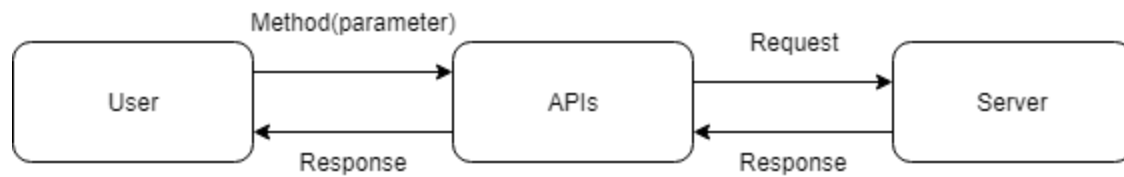
There is no similar of such projects in the internet. Therefore, there is no case studies to refer to in this project. Based on the project requirements, the project can break down few different parts as following: -

World Map Services

There are few options such as Google Map, OpenStreetMap and GraphHopper to choose from. In early research, all three-world map service also looks the same and provide APIs to deploy the world map services to the project.

Application Programming Interface (APIs)

What is APIs? According to Gazarov(2016), API is part of the server that receives requests and sends responses. The following diagram will show how APIs works.



API is the middle-man between User and the Server (API provider). For example, the server is Google. Google has copyright to their products but with APIs, it is still possible to use their map services in user's program. The user requests API from Google, then Google will give 'access' for the user to use their services. The user request from Google with the 'access', API and Google (Server) will responds to the method and parameter that passed to them. This is widely used in most of the applications nowadays.

There are many types of APIs that Google Map services provide (Google Map,2018 -a):

- Static API
- Street View API
- Directions API
- Distance Matrix API
- Roads API
- Places API
- Geocoding API
- Geolocation API

Get Image from World Map Services

There are two possible approach found from the early research to get the images from world map services.

Html2canvas allows user to take ‘screenshots’ of the webpages and display it in the browser or download it (Hertzen,n.d.). The initial plan is to take screenshot from the Google Map services and process it with image processing library or program.

Google Static Map allows user to get a Google Maps image on the webpage (Google Map, 2018-b). It basically generates a Uniform Resource Locator(URL) with certain properties that the user can manipulate at their own will. It requires API key and signature from Google in order to use this feature.

Image Processing

After some researches, there are two available library which have image processing library that allows user to manipulate and access the data of the image.

OpenCV (Open Source Computer Vision Library) is built to enhance the usage of machine recognition such as face recognition, identify objects, track movement of the camera, detect similar patterns between pictures and many more (OpenCV team, 2018). This library can develop filters in the social media nowadays. It detects the facial structure of the user and apply filters on it. However, it also provides image processing functions to allow user to manipulate from it. It is available in different programming interfaces such as C++, Python, Java and MATLAB.

GraphicsMagick which originally derived from ImageMagick, is purely image processing library that supports various functions to an image over different kinds of image file format (GraphicsMagick,2018). This change is made by many authors and confirmed that it is far more useful and efficient compared to ImageMagicks. It allows user to manipulate and change the properties of the image at their own will. Compared to OpenCV, it offers more programming interfaces that user can choose from.

World Map Road Database

There exists database which contains all the information of the roads of the whole world. One of the example is OpenStreetMap’s planet.osm which provides all the places, routes and relations between them (OpenStreetMap,2018).

Calculate surface area per pixel

It is possible to count the surface area per pixel in the map services with the formula below. The output of this formula is meters per pixel.

$$\begin{aligned} \text{ground resolution} &= \cos(\text{latitude} * \pi/180) * \text{earth circumference} / \text{map width} \\ &= (\cos(\text{latitude} * \pi/180) * 2 * \pi * 6378137 \text{ meters}) / (256 * 2^{\text{level}} \text{ pixels}) \end{aligned}$$

Figure 1. Ground Resolution Formula. (Schwartz, n.d.).

2.2 Project Risks

Risk Register

No	Description	Causes	Probabilities	Risk reduction Strategy
1	User might not able to run the program in the client side	There is no active internet connection	High	Risk acceptance – accept the risk and find a active internet connection nearby
2	Found a major bug in the code	Not fully understand the algorithm or functions	High	Risk mitigate – reduce the occurrence by understand the algorithm fully before implementing
3	Project went overschedule	Poor time management	Medium	Risk mitigate – avoid this risk by proper manage the time and progress of the project
4	The total area displayed to the client side is not accurate	Communication between server and client is not functioning	Medium	Risk mitigate – proper implementation and testing will reduce this risk to happen
5	Misunderstand the user requirement	Lack of communication with advisor	High	Risk avoidance – communicate with advisor to get the precise project requirement
6	Project requirements changes during implementation stage	Advisor might add new features to the project	Low	Risk acceptance – If there is additional requirement during project implementation, developer has to accept it
7	Lack of motivation during project implementation	Faced a lot of difficulties during implementation	Low	Risk avoidance – Continuously researching and working until project is delivered

Risk register could easily identify the risk might occurred in this project. It included the causes, probabilities and suggested risk reduction strategy. The risk that actually occurred in this project is ‘project went overschedule’. The implementation phase of this project went overschedule and hence the deliverable was unsatisfied. There is not enough time for Graphical User Interface. Nonetheless, the project objectives was achieved.

2.3 Resources Requirement

Hardware

- Computer with proper internet connections

Software

- JavaScript
 - o Back end programming language (Server side)
- Hyper Text Markup Language (HTML)
 - o Front end programming language (Client side)
- Google Maps APIs
 - o Provides keys and signature to the project
 - o Allows extra features for Google Map

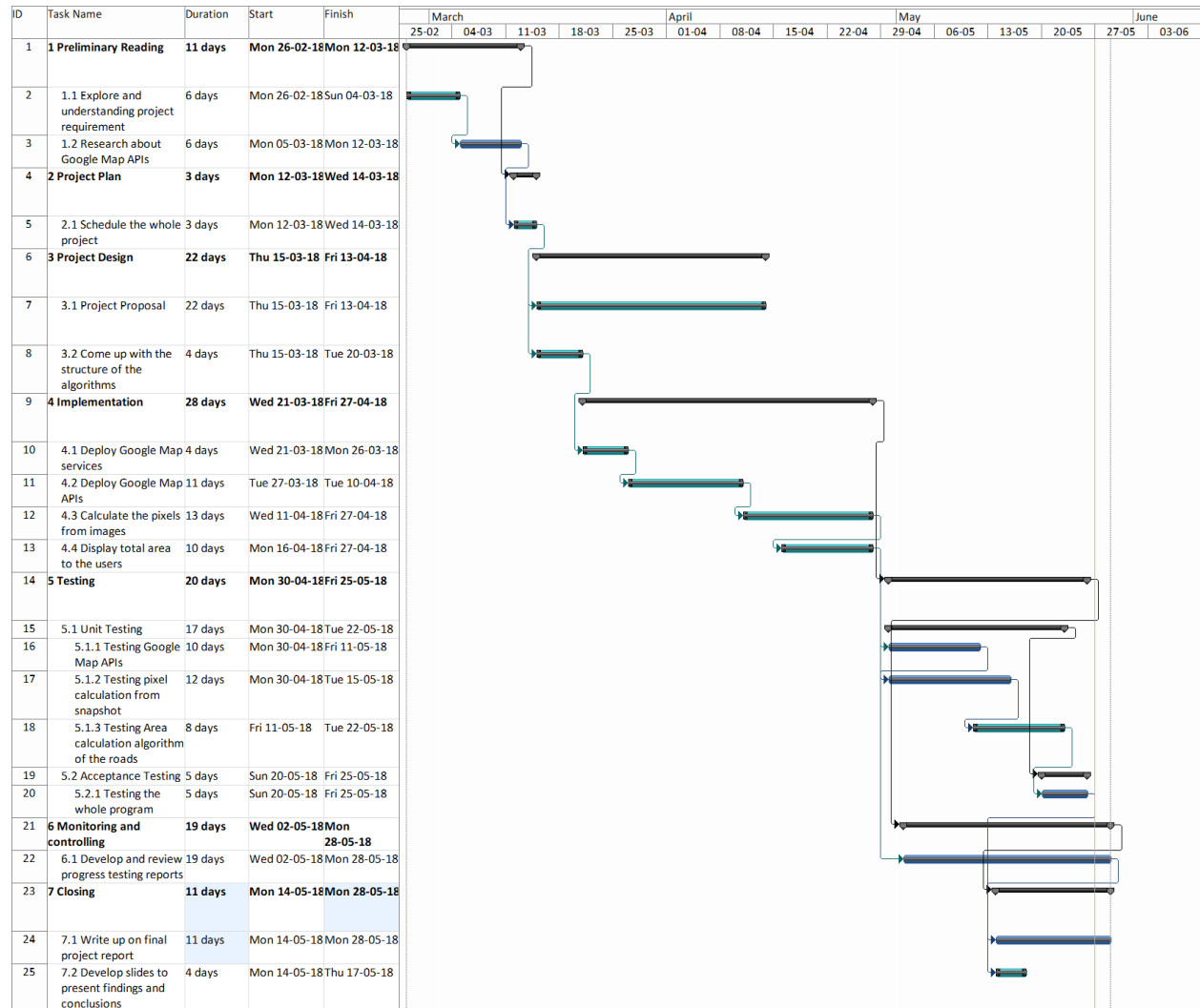
2.4 Project Timeline

Required tasks and dependency relations

1. Set up local server
2. Server communicate with client in real time
3. Able to capture image from world map services
4. Calculate the surface area from the image
5. Display the total surface area to client

These are the five main required tasks in order to produce the final project. Every task is dependent on the previous task.

Completed project timeline



As mentioned in the risk above, this project faced some difficulties in the implementation part and hence the project schedule went off. Everything was push behind and cramped during the last few weeks. Due to the overscheduling in implementation, testing and final project report are doing concurrently. However, the project still able to deliver the project and able to achieve the project requirements.

3.0 Method

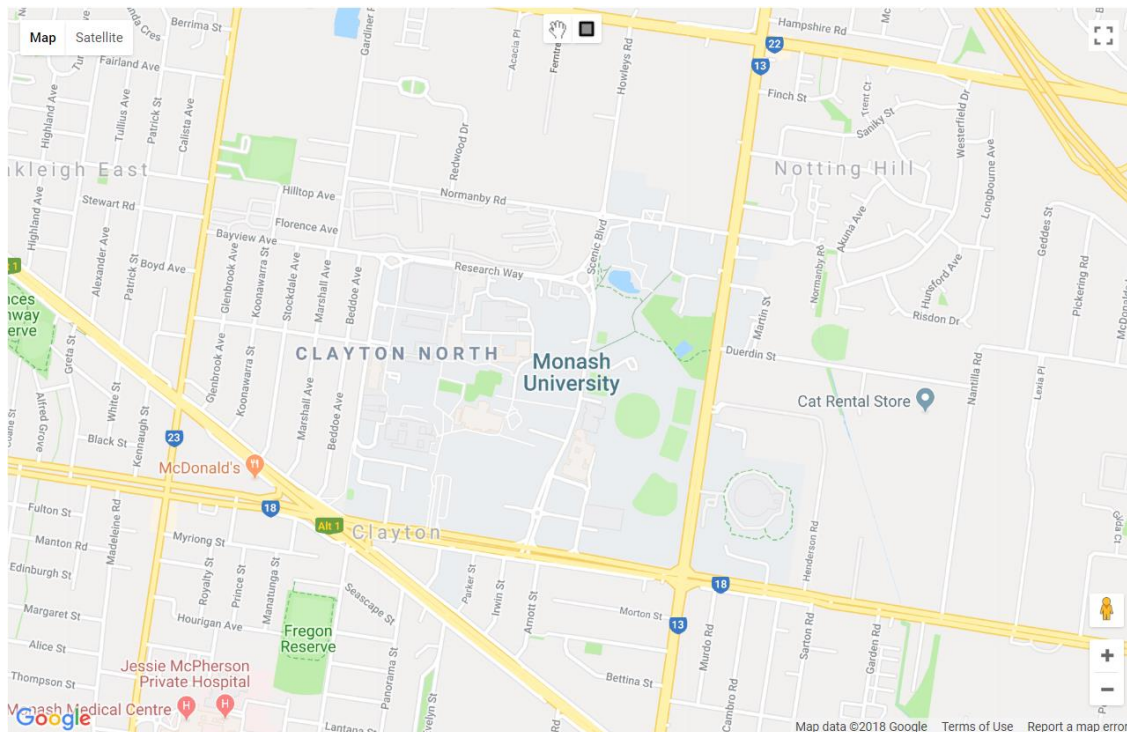
3.1 Methodology used

This project uses waterfall approach as every task is dependent on the previous task. Some of the task can start concurrently with the current task, but mostly is still dependent of the previous few tasks.

The following are the methods that are used in this project: -

Google Map Javascript API

This API allows the program to display and use the functionality of basic Google Map Services. It has normal functions such as zoom in, zoom out, full-screen mode, and change the type of the map. Google also offer drawing tools which the user can drag and draw on the Google Map. This project also included rectangle drawing tools for the user to draw the area that they wanted. This functionality is shown in the top middle of the image below.



This image is retrieved from the Google Map deployed in the project

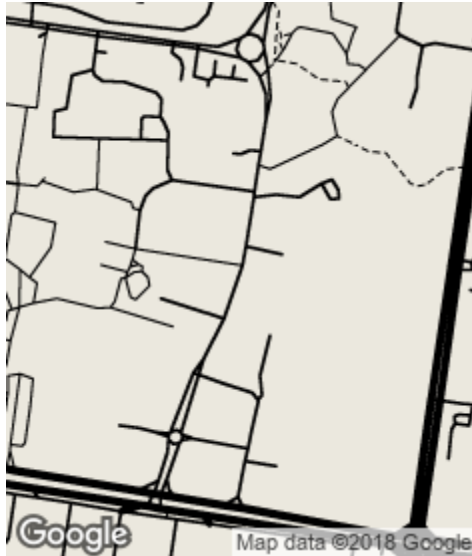
Express

Used to set the web applications and able to run within a local server with a designated port - <http://localhost:3000>

During early research, there are two approaches to find the total surface area of the road which is counting pixel and using world map service database. Using database and get the road details will have better complexity compare to counting pixels in static map. However, counting pixels in static image is more simple approach. After some considerations are done, counting pixels approach is chosen because of project time and code complexity. This decision trade off some time complexity with simpler implementation.

Google Static Map API

This API allows the programs to generate a static URL based on the feature that the program set. This could help to strip everything else other than road only image from Google Map. In this project, the road is color as black (#000000). However, there is google trademark in every static images at the below of the picture. This can be solved by generating longer static images and cut it off during processing.



Sample image which generated from the Google Static Map in the project

Socket.IO

Socket.IO is a bi-directional communication library that triggered based on event for web applications (Kellher, 2014). The front-end has to send static map URL to the back-end of the program for further processing. After that the server has to serve back the value of total surface area to the front-end. Therefore, real-time communication, event-based library is suitable for this project.

GraphicsMagick & pngjs

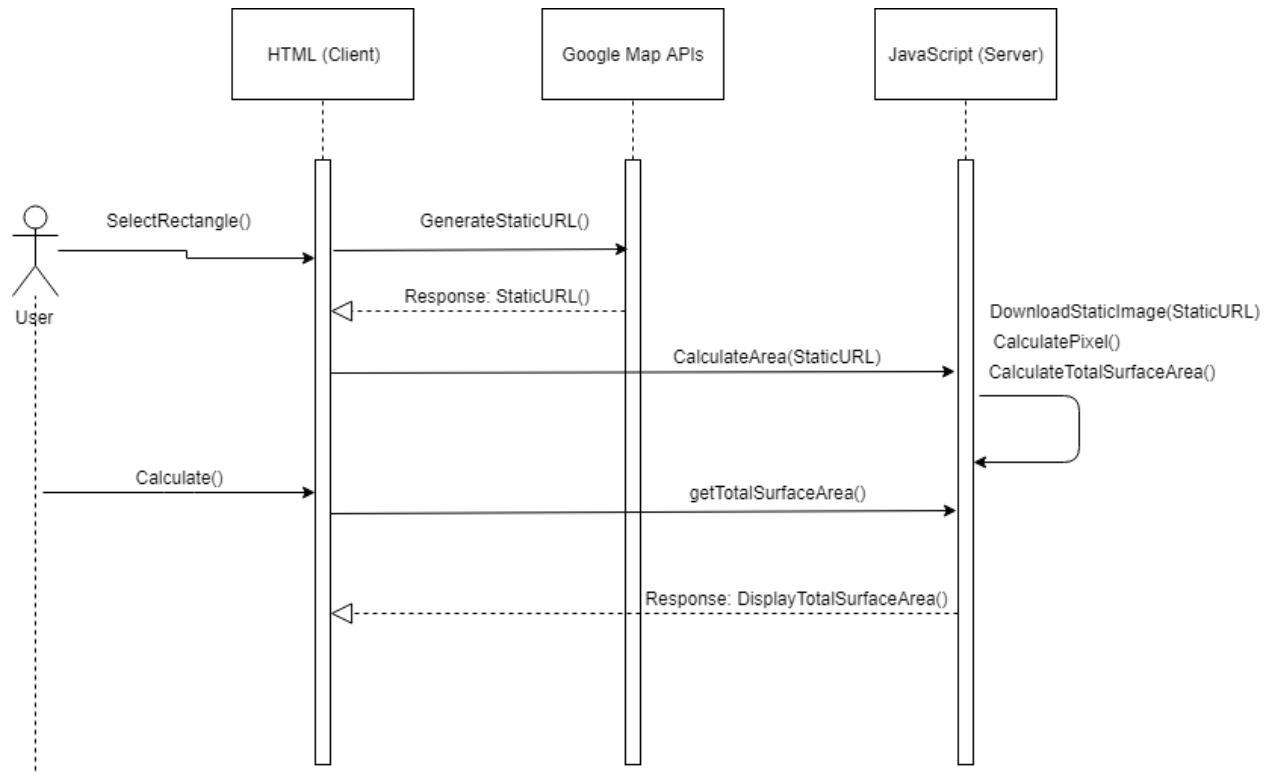
GraphicsMagick access the static image downloaded by the back-end programming code. Pngjs act as a buffer to read every pixel from the image. The program only counts the black pixel and return the count.

Get Rectangle Width & Height

Google Map Services provides methods that get the coordinates of the specific points and returns it in pixel distances which is fromLatLngToPoint() method (Google Map, 2018-c).

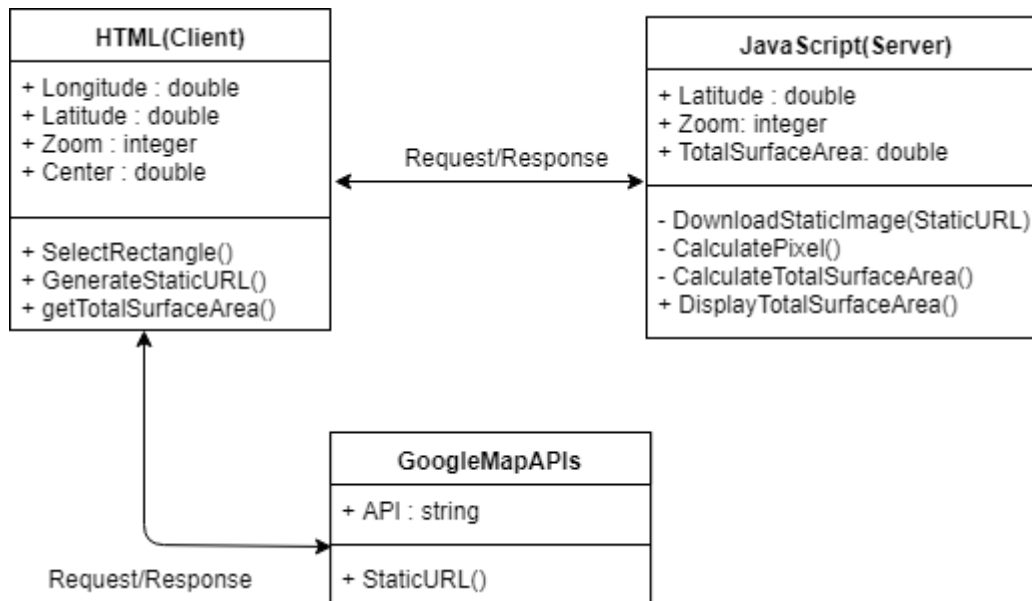
3.2 Internal Design

Sequence Diagram



User select area from Google Map. After the drawing is done, the program will request from Google Map APIs to get Google Static Map URL link of the rectangle that the user draw. Next, the URL will be sent to the server and proc the server to download the static image with that Static Map URL. Server side will access the static image and calculate pixels and total surface area. When the user clicks the 'calculate' button, the server will send back the value of total surface area of the road to the client side and display to the user.

3.3 Software architecture

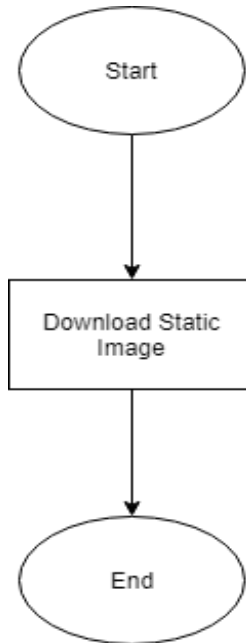


Class diagram describe the overall structure of the code. Basically, client act as middle-man of the program. It gets the Static Image from the Google Map and pass to the server for further processing. Client request the server to download the static image that get from Google Map APIs. Server responded by download the image, calculate pixel, and calculate the surface area of the road. Client could also request the total surface area to be displayed in the client side, and server responded by sending the value to the client side via socket io.

3.4 Functions and Algorithms

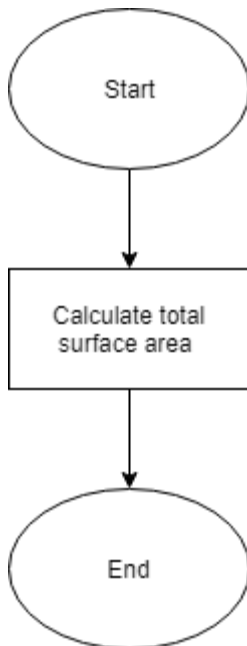
There are three main functions in the server side: -

Downloadimage(url,latitude,mapzoom)



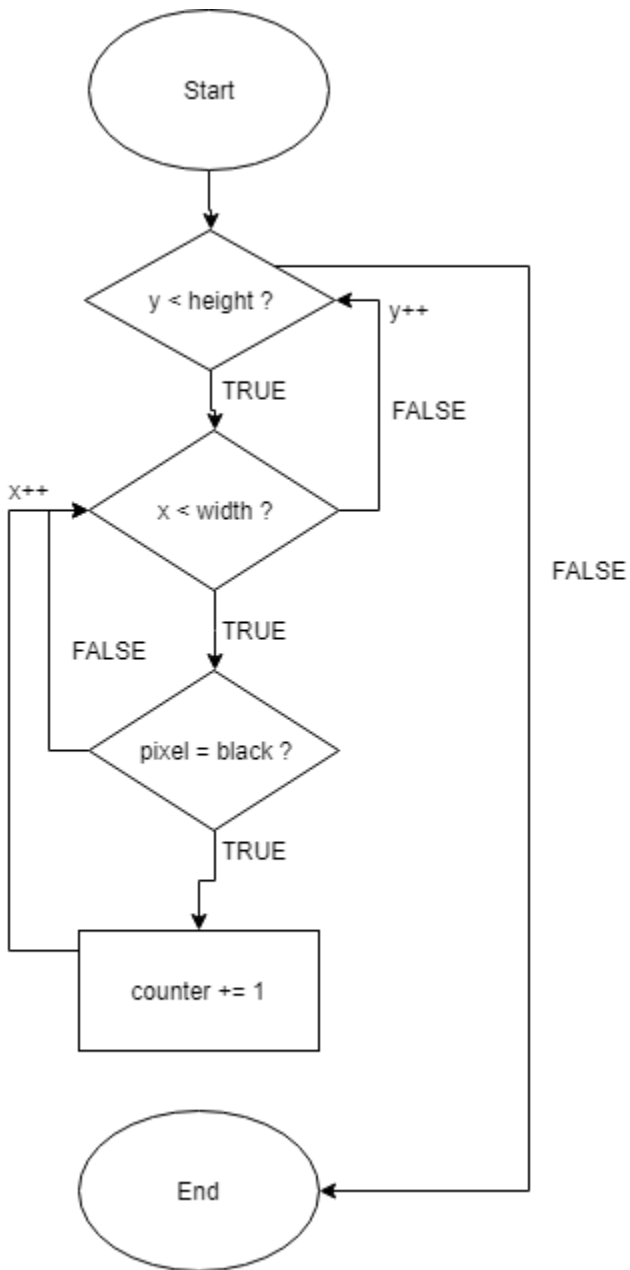
This function receives the Google Static Map URL from the client side. Its function is only to download the image to local storage and call the next function – `calculatepixel()`

Calculatearea(counter)



This function takes the counters (black color pixels) and calculate the total surface area with the formula included and return it. This function uses the formula found in the early research to calculate the surface area.

Calculatepixel()

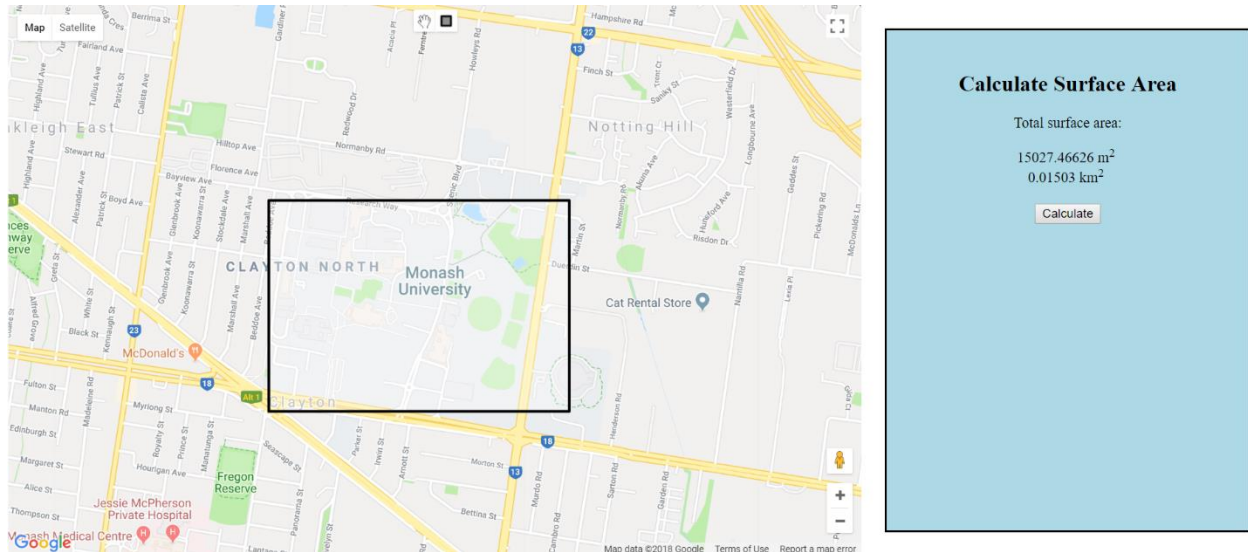


This function takes the static image and process it. It loop through the height and width of the image to determine whether the current pixel is black in color or not. This function will terminate when it done the looping and call calculatearea function and pass the parameter counter.

4.0 Result

4.1 Result obtained

The result of this program is different every time because the user decides the area that they wanted to calculate. Unless the user draws the exact same rectangle on the exact place, and the result will be the same.



The result obtained above from the program is the total surface area of the rectangle drawn by user is 15027.46636-meter square and 0.01503-kilometer square.



The image above is the static image generated by the program.

```
Final Year Project is listening on port 3000!  
Socket is connected !  
Server received the static map URL from client side !  
Total surface area is sent to the client side !  
Downloaded the Google static map image from URL !  
Calculating the area of the road ...  
Time taken : 1367  
Accessing the pixel of the static map image ...  
Counter = 3987  
Calculating the total surface area ...  
Surface area = 15027.46626074228 meter square  
Total surface area is sent to the client side !
```

The image above is the screenshot of the server side.

The time taken in milliseconds: 1367

The time taken is measured between after the drawing is done and done calculating the total surface area. User click the 'calculate' button doesn't take into consideration for time measuring is because every user has different timing of clicking the button.

Total of black pixel detected: 3987

Total surface area of the road: 15027.46626-meter square

The total surface area displayed in the server and in the client are the same. The value is processed in the client side to become kilometer square.

4.2 Input / Output

Input

- The user draw rectangle on the Google Map

Output

- Static Image of the rectangle
- Total surface area of the road in meter square
- Total surface area of the road in kilometer square

4.3 Performance

Time complexity: $O(WH)$

- W – width of the static image
- H – height of the static image
- The program process through the static image to get the black pixels in the static image which is height*width.

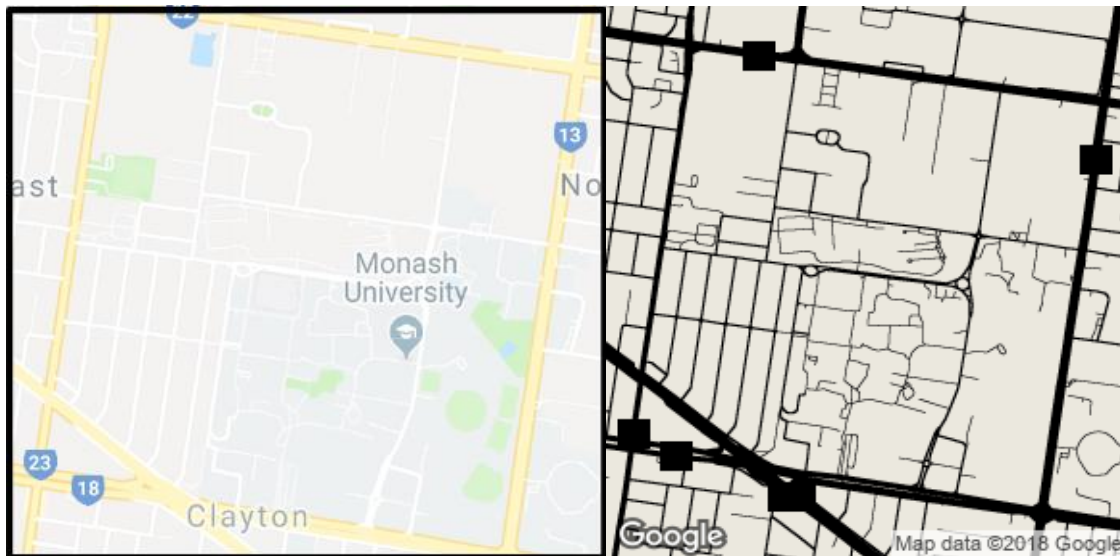
Space complexity: $O(WH)$

- W – width of the static image
- H – height of the static image
- The program downloads the static image with size of height*width pixels

The time taken in milliseconds: 1367 – refer to 4.1 results obtained.

5.0 Analysis and discussion

5.1 Result Analysis



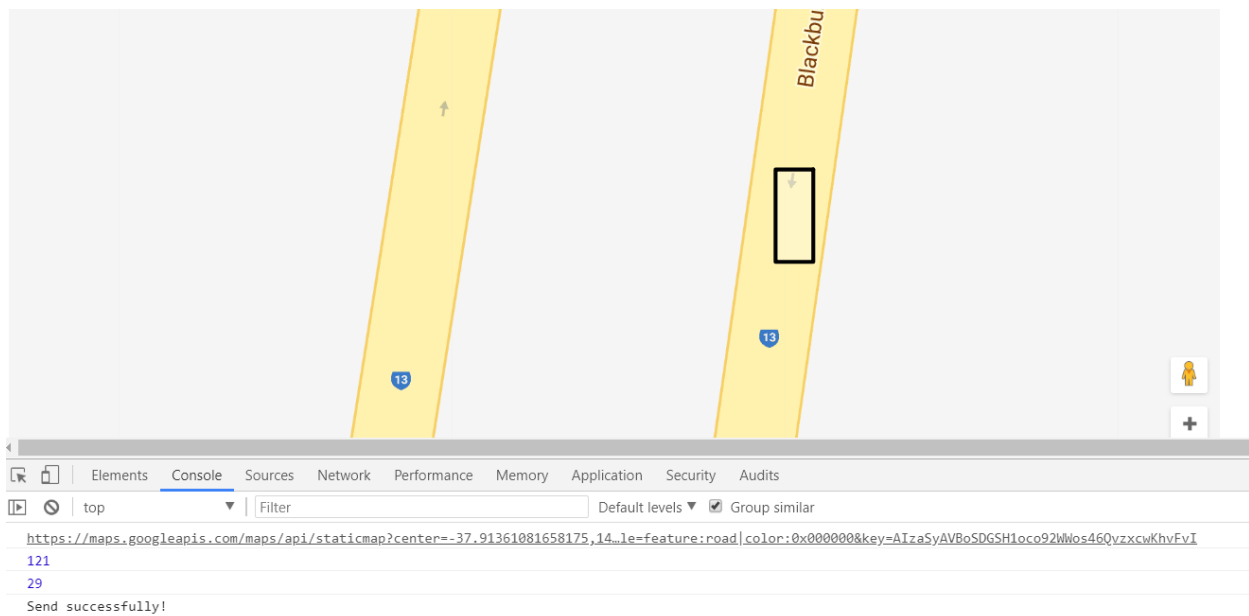
The size taken is larger compared to the results obtained in 4.1.

```
Final Year Project is listening on port 3000!  
Socket is connected !  
Server received the static map URL from client side !  
Downloaded the Google static map image from URL !  
Calculating the area of the road ...  
Time taken : 660  
Accessing the pixel of the static map image ...  
Counter = 5509  
Calculating the total surface area ...  
Surface area = 41530.16741195151 meter square  
Total surface area is sent to the client side !
```

However, the time taken is 660 which is lesser than 1367. This is because the program uses buffer to go through the pixels and it will be faster if there's more memory in the server.

However, the stripping part of Google Static image is not that perfect as there is still black boxes which shows the number of the road. This definitely affected the accuracy of the output and contributed more to the surface area.

Next analysis is take the road only in the rectangle.



The first line of the console is the static map URL

The second line of the console is the height of the rectangle - 121

The third line of the console is the width of the rectangle – 29



The image beside is the static map generated from the rectangle. As mentioned above, there is an issue with the google map trademark which affect the pixels counting.

```
Final Year Project is listening on port 3000!  
  Socket is connected !  
  Server received the static map URL from client side !  
Downloaded the Google static map image from URL !  
Calculating the area of the road ...  
Time taken : 584  
Accessing the pixel of the static map image ...  
Counter = 2059  
Calculating the total surface area ...  
Surface area = 121.25597118226267 meter square
```

Nonetheless, the server still can calculate the correct surface area. While generating the static image, the height of the static image is added by 50 pixels more and will cut the 50 pixels when processing it. Therefore, the height got in the client side's console is $121 - 50 = 71$ pixels only.

$71 * 29 = 2059$ pixels

The counter – 2059 is the number of black pixels covered in the static image. This proves that the Google trademark is cut off and the surface area is accurate from the pixel counting method.

There is still some improvement to be done in the static image to make the program even more accurate, but through the second analysis the program does provide the correct total surface area to the Victoria council developer.

6.0 Future work

The first thing is to enhance and beautify the user interface. The implementation part went overscheduled and hence there is not much time to code the user interface. Try to make it more interactive and display more information such as the center of the rectangle, the name of the places and the longitude and latitude.

Next, it would be nice for this program to be available in the internet and not just local server. Everyone can access it through the world wide web and use this application. This might cost a little bit for the maintenance of the website after deployed in the internet. The server might need a database as well to support the data in and out of the application.

Apart from this, changing the implementation of the project by using world map road database approach. This implementation could give faster complexity and run-time compared to pixel-counting. The accuracy of the output also will be accurate as the data are from their own database as well.

Lastly, add more features to the current application such as user can type in the longitude and latitude to determine which area to count and the size of the rectangle as well. Features such as directions, distance between two points can also add in to this application. These features would help to enhance the user experience and satisfaction while using the application.

7.0 Conclusion

In conclusion, the project objectives and requirement are achieved. Although the implementation phase went overschedule and delayed the tasks behind, the project still able to deliver successfully in time. There are some imperfections in the programs such as the static map didn't strip off everything else except roads only and the user interface is not user-friendly. Nonetheless, the basic requirements are met, and the project is considered as successful. As for the testing part, it went smoothly and had no problems with it. There are definitely lots to improve in the future in order to make this application even more appealing and user-friendly.

8.0 Bibliography

- A quick tour of Roadway Characteristics Editor RCE (n.d.) Retrieved from <https://desktop.arcgis.com/es/arcmap/10.4/extensions/roads-and-highways/a-quick-tour-of-event-editor.htm>
- Gazarov, P. (2016). What is an API? In English, please. Retrieved from <https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>
- Google Map (2018-a). Google Maps Platform documentation. Retrieved from <https://developers.google.com/maps/documentation/>
- Google Map (2018-b). Google Maps Static API. Retrieved from <https://developers.google.com/maps/documentation/maps-static/intro>
- Google Map (2018-c). Projection interface. Retrieved from <https://developers.google.com/maps/documentation/javascript/reference/3/image-overlay#Projection.fromLatLngToPoint>
- GraphicsMagick Group (2018). GraphicsMagick Image Processing System. <http://www.graphicsmagick.org/index.html>
- Hertzen, N. V. (n.d.). About HTML2CANVAS. Retrieved from <https://html2canvas.hertzen.com/documentation>
- Kelleher, F. (2014). Understanding Socket.IO. Retrieved from <https://nodesource.com/blog/understanding-socketio/>
- Michaz (2016). Map Matching based on GraphHopper. Retrieved from <https://github.com/graphhopper/map-matching>
- OpenCV team (2018). About OpenCV. Retrieved from <https://opencv.org/about.html>
- OpenStreetMap (2018). Planet.osm. Retrieved from https://wiki.openstreetmap.org/wiki/Planet.osm#Regional_extract_sources
- Open Source (2018). Retrieved from <https://www.graphhopper.com/open-source/>
- Schwartz, J. (n.d.). Bing Maps Tile System. Retrieved from <https://msdn.microsoft.com/en-us/library/bb259689.aspx>

9.0 Appendices

9.1 Production and Deployment

Instructions to install/run the software

1. Unzip the file
2. Open command prompt and go to this file directory
 - a. User can right click the file and select 'properties' to find the file directory
 - b. Typing 'cd' with the file location can change the directory of user's current path.
Example: 'cd Desktop'.
3. Install the dependencies as following: -
 - a. npm install express

```
C:\Users\ASUS\Desktop\FYP>npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
+ express@4.16.3
added 50 packages in 2.119s
```

- b. npm install socket.io --save

```
C:\Users\ASUS\Desktop\FYP>npm install socket.io --save
+ socket.io@2.1.1
added 36 packages in 1.869s
```

- c. npm install gm

```
C:\Users\ASUS\Desktop\FYP>npm install gm
+ gm@1.23.1
added 10 packages in 1.493s
```

- d. npm install pngjs2


```
C:\Users\ASUS\Desktop\FYP>npm install pngjs2
npm WARN deprecated pngjs2@2.0.0: pngjs2 has now taken over the original pngjs package on npm. Please change to use pngjs dependency, version 2+.
+ pngjs2@2.0.0
added 1 package in 1.763s
```

4. Then run the program by typing 'node app.js'

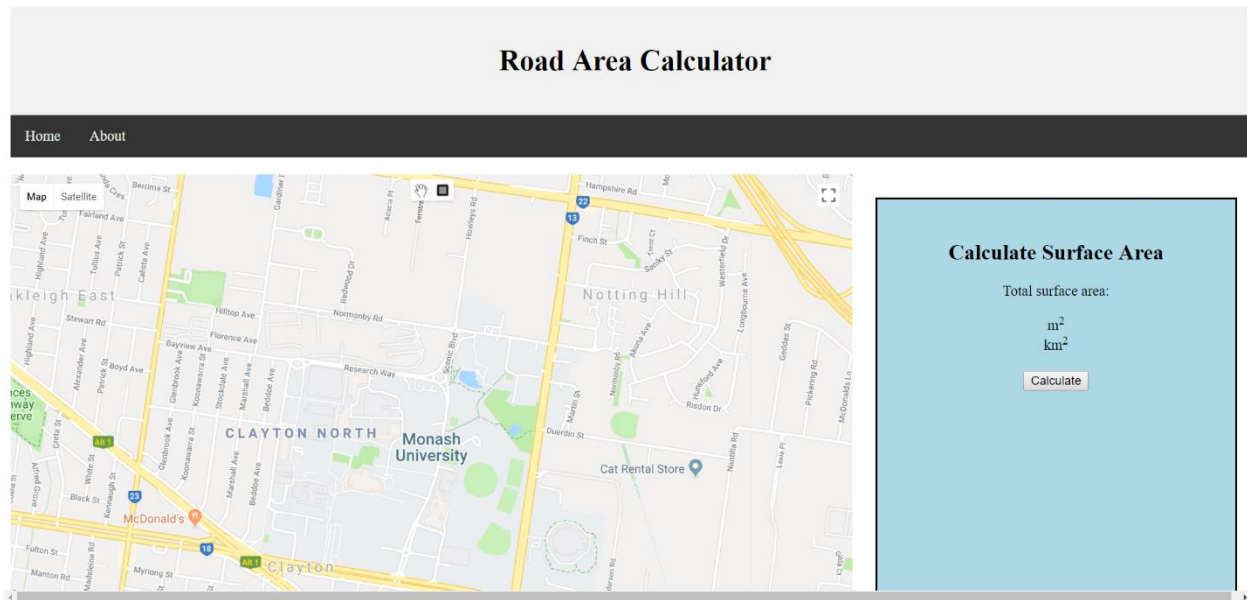
```
C:\Users\ASUS\Desktop\FYP>node app.js
Final Year Project is listening on port 3000!
```

9.2 User Interface

1. Go to the computer's default browser and type in 'http://localhost:3000/'

 localhost:3000

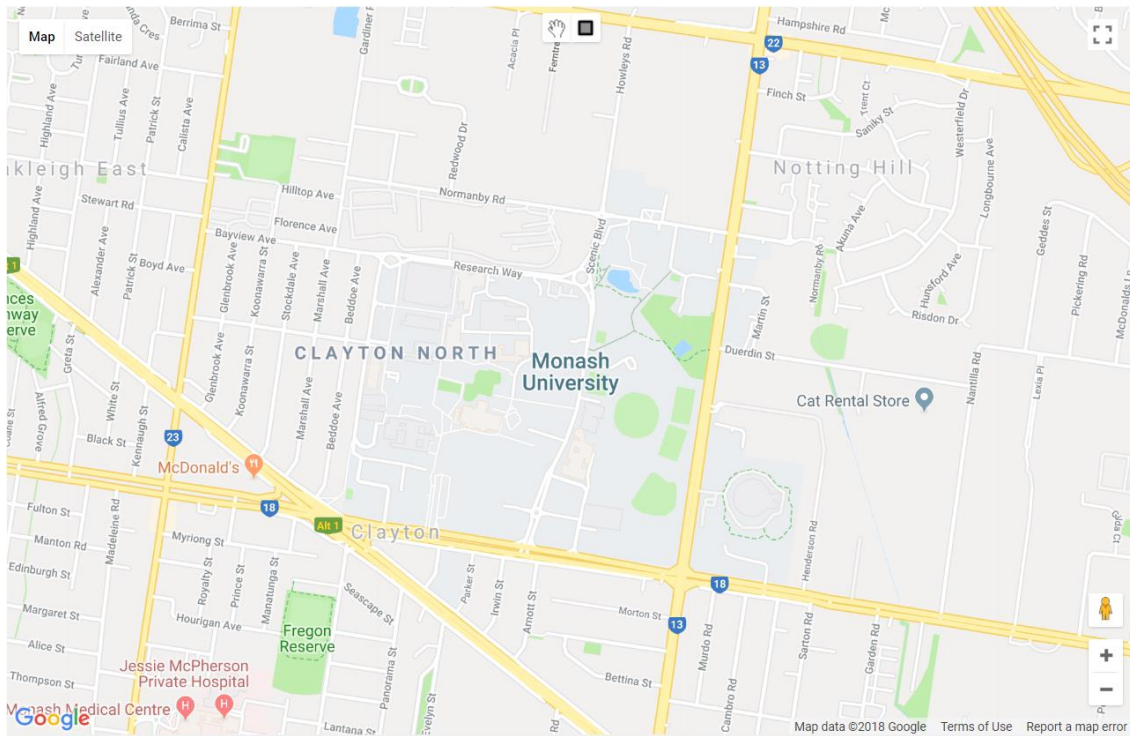
This screen will appear in the browser.



2. The server will be connected with the user

```
C:\Users\ASUS\Desktop\FYP>node app.js
Final Year Project is listening on port 3000!
Socket is connected !
```

3. There are few properties in the Google Map in the program: -



a. Map Type

- i. Default Road Map View (Top Right)

Map

- ii. Google Earth Satellite images (Top Right)

Satellite

b. Drawing Tools

- i. Browse Map (Top Middle)



- ii. Draw Rectangle (Top Middle)



c. Full Screen (Top Right)



d. Zoom In (Bottom Right)



e. Zoom Out (Bottom Right)



4. Click the 'draw rectangle' button and draw a rectangle on the Google Map.



5. Click the 'Calculate' button.

Calculate Surface Area

Total surface area:

m^2
 km^2

Calculate

6. The total surface area will be displayed in m^2 and km^2

Calculate Surface Area

Total surface area:

6976.60839 m^2
0.00698 km^2

Calculate

9.3 Internal testing procedures

- 1) Open command prompt and go to this file directory
 - a. User can right click the file and select 'properties' to find the file directory
 - b. Typing 'cd' with the file location can change the directory of user's current path.
Example: 'cd Desktop'.
- 2) Install the following dependencies: -
 - a. npm install mocha

```
C:\Users\ASUS\Desktop\FYP>npm install mocha
+ mocha@5.2.0
added 22 packages in 2.987s
```

- b. npm install chai

```
C:\Users\ASUS\Desktop\FYP>npm install chai
+ chai@4.1.2
added 7 packages in 1.25s
```

- c. npm install superagent

```
C:\Users\ASUS\Desktop\FYP>npm install superagent
+ superagent@3.8.3
added 15 packages in 1.599s
```

3) Type 'npm run test' will run the unit testing.

```
C:\Users\ASUS\Desktop\FYP>npm run test

> fit3036@1.0.0 test C:\Users\ASUS\Desktop\FYP
> mocha || true

Final Year Project is listening on port 3000!
App
  ✓ App should respond to GET html.

Socket-Server
  Socket is connected !
  ✓ User able to connect to the server.
  Socket is connected !
  Server received the static map URL from client side !
  ✓ User can communicate to the server in real time

App function
  ✓ Server able to get URL & download static map image - downloadimage(url,latitude,mapzoom)
  ✓ Server able to calculate the pixel based on the static image - calculatepixel()
  ✓ Server able to calculate the surface area of the road in the static image - calculatearea(counter)

6 passing (94ms)

Socket is connected !
```