



Discoverus

By

Tomas Brazas



Applied Project & Minor Dissertation

Department of Computer Science and Applied Physics

In partial fulfilment of
Honours Bachelor Degree in Computing
In Software Development

Supervised by Kevin O'Brien

10th of May 2020

Table of Contents

1	Introduction	5
1.1	Context.....	5
1.2	Objectives	5
1.3	Chapter summaries	6
2	Methodology.....	8
2.1	Project Management and Research	8
2.2	Development Methodology	8
2.3	Distinguish between Methodologies for project	9
	Waterfall	9
	Agile	10
	Comparison	11
	Table 1. Difference between the models.....	11
	Conclusion	11
2.4	Version Control.....	12
3	Technology Review	13
3.1	MEAN versus MERN Stacks.....	13
3.2	Research Competition	14
3.3	Chosen Technologies	16
	Following research, the project was decided to use the MEAN stack as it fits the objectives and scope of the project. This section consists of the technologies, tools, aiding technologies, frameworks and languages that have been implemented and deployed in the project.	16
	Angular	16
	MongoDB	16
	ExpressJS	17
	NodeJS	17
3.4	Languages	18
	Typescript	18
	JavaScript	19
	HTML	19
	CSS	19
3.5	Programming Tools	20

Visual Studio Code	20
MongoDB Compass	20
3.6 Other Tools	21
Adobe Photoshop CC 2015	21
4 System Design	22
4.1 Overview	22
4.2 User Registration and Login – Back End	23
Dependencies	23
Model	24
Routes	25
Express Server	28
4.3 User Registration and Login – Front End	30
Authentication Service	30
Auth-guard Service	31
Angular Routing	32
4.4 Views	33
Welcome Page	33
Login Page	34
Register Page	34
Profile Page	35
Home Page	36
4.5 Google Maps	37
Directions	39
5 System Evaluation	40
5.1 Evaluation	40
5.2 Testing	41
5.3 Limitations and Opportunities for Improvement	41
6 Conclusion	43
6.1 Learning outcomes	43
6.2 Further Development of Discoverus	44
Adding a Location to Favourites	44
Profile Customization	44

	Allowing users to interact between each other	44
	Mobile friendly	44
6.3	Final Conclusion	45
7	BIBLIOGRAPHY	46
8	Appendix of GitHub	47

Brief Overview

Abstract. Google Maps has gained traction upon development. Personalized and functional maps are more and more common in modern applications such as they represent companies specific needs like displaying hub network of their infrastructure around a map. From being coded by a development team to revolutionizing the world of digital maps.

Discoverus is an accessible and easy to use web application for people who wish to explore and discover the local area. Google maps allows to explore the surrounding area to for activities and other facilities.



1 Introduction

The introduction will function as an overview of the project. The context, objectives and an overview of the chapters of this documentation will be described in this chapter.

1.1 Context

While brainstorming for a project idea I have decided to use new technology and make the base of the application relevant to me and the target audience of those who love to travel and explore different venues and locations. The project must also hone existing skills and allow for the natural development of new techniques and process while also being worthy in scope.

Discoverus, a navigation application to venues and locations using Google maps was my aim to achieve during this project. Searching for a location will specify the nearby venues from the Discoverus database, showcasing the information about the venue and directions if interested while existing as a simple to use platform. As an avid travel enthusiast, I found a shortage of web applications that allow the user to find the best spots in a local area.

1.2 Objectives

The main objective of the project is to create a web application that would let explorers and tourists indulge in the local area providing the best venues and activities around the Galway county region. The pre-defined locations by the developer are present on the map handpicked by reviews.

Main objectives of the project are:

- Design and develop a user-friendly application that is easy to use and understand.

- Apply and practice a software development methodology with the project.
- Integrate state of art technologies focusing on Google Maps.
- The web application will allow user to register an account, login, discover venues around the area and get directions from their current location to their selected venues location.

1.3 Chapter summaries

1. Introduction

This chapter outlines the context of the project with the objectives of the web application. Brainstorming result and technologies to be used in the project.

2. Methodology

This chapter outlines the summary of how the project was planned, approached and managed. Provides layout of testing and development. The way difficulties were approached.

3. Technology Review

This chapter outlines all the research that has been carried out before beginning code and research throughout the development of the web application. In depth research about each of the technology being used in the project and the future and past of google maps applications.

4. System Design

This chapter outlines a detailed explanation of the overall system architecture with the aid of diagrams, code snippets and screenshots.

5. System Evaluation

This chapter evaluates the project against original objectives that have been set out. Highlights the limitations in the project approach and limitations of the technologies used.

6. Conclusion

This chapter outlines the summary of the project. Summarizes research, result of the project and valuable experience gained from development of the project.

2 Methodology

2.1 Project Management and Research

Once the project requirements have been defined and finalized, the overall development of the project had begun. Pre-development research had to be done before the concept of the project had been finalized. Development began once I had an idea and had enough research done to begin and set up the environment. This chapter is focused on pre-development process and initial project direction.

During the initial supervisory meeting, the requirements for the project were stated and question has been raised if the project should be undertaken in a group or individually. Conclusion was made to undertake the project individually as experience was already obtained with group projects from the previous years. Sketched visual aid was done and presented to show the direction of the project and allow flexibility for change.

During the first few weeks of development and supervisor meetings, the idea of a navigation application was pitched to the supervisor, a web application that would heavily focus on the integration of google maps as a new technology. The project was undertaken throughout the course of the semester.

Research was undertaken of similar web applications that integrate google maps as one of its core features. There are only a handful web applications that solely focus on google maps as the main core feature. This is usually the basis of a navigation application. More about this is covered in detail in the next chapter of the dissertation.

2.2 Development Methodology

To undertake this project, a specific methodology had to be in place to sufficiently work through the project. There were numerous methodologies to pick from and tackle the solution for the project with such as

“Waterfall”, “Feature-driven Development”, “Agile” and “Extreme Programming”. The final two methodologies were chosen to be Agile or Waterfall.

2.3 Distinguish between Methodologies for project

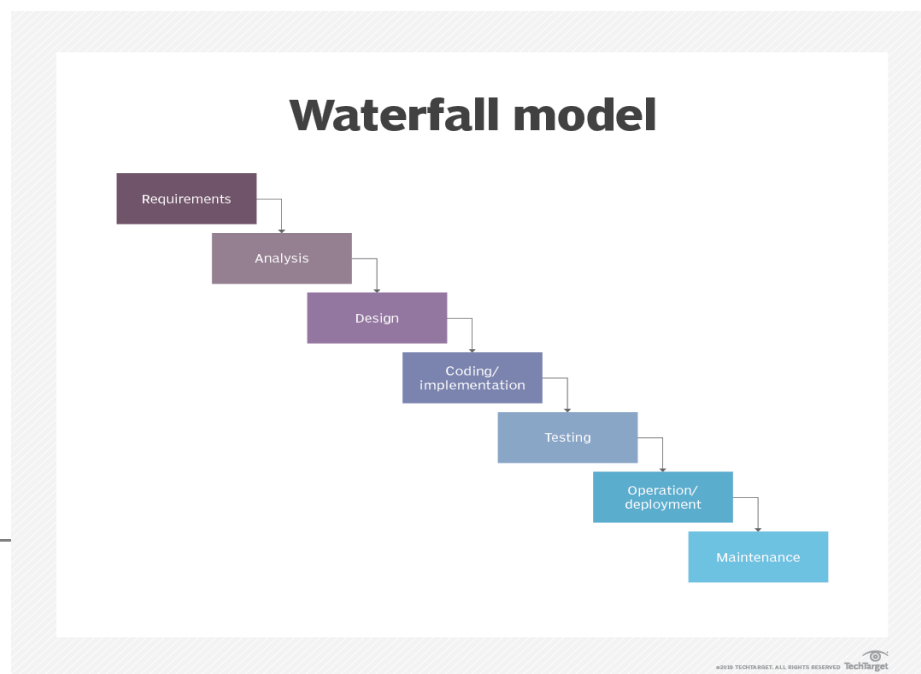
Upon reaching the final decision of applying two methodologies, research had to be undertaken to consider the best fit methodology for the project that will cover the objectives and consider the scope. Research of the two methodologies is thoroughly covered including the advantages and disadvantages of each, comparisons and an analysis of each methodology considering the project to be undertaken.

Waterfall

Waterfall methodology[7] is a traditional software development methodology. It is easily understandable and manageable model. Waterfall model is based on a sequential design process much like a nature’s waterfall filling lower level pools, phases in the Waterfall model flow one to another. Also like the pools filling completely before water spills into the next pool, the waterfall model finishes one phase before the next phase can begin. Since the overall design of the application is undertaken in the early phases, changes according to the customer can not be made once design is finished. Waterfall methodology lends itself to projects that are small in size and whose requirements can be definitely determined upfront.

The Waterfall model is well known development model as it has been around for decades and its overall simplicity. Its simplicity allows for easy manageability.

Fig. 1. Waterfall Model



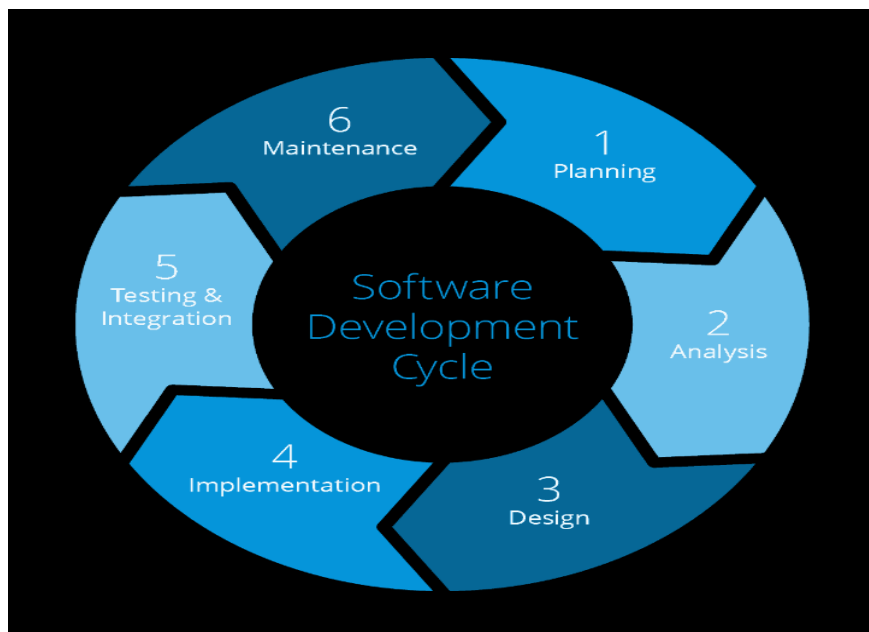
Agile

Agile development[6] time-boxed iterative approach to project management that builds software incrementally from the start of the project through the development life cycle. The timeline of the project is fixed and cannot be extended. While the timeline of the project is fixed the deliverables of the project are prioritized and delivered in a 2-week cycle. Otherwise known as “sprints”. The workflow in Agile is flexible and encourages constant improvements in sprints. Hence, the end product will be very close to what the customer had wanted. There are four key roles in Agile development:

- Product Owner
- Scrum Master
- Team Member(s)
- User

All these roles play an essential role in the agile workflow. Product Owner defines the features of the end product. Scrum master prioritizes the features to be worked on in each sprint. Team member creates and develops the project throughout each iteration. User uses the product and provides feedback. It allows software to be delivered to the user in periodical releases. Agile focuses on flexibility which allows requirements to be changed during the development process. Agile is one of the most popular methodologies in software development. Agile projects are 28% more successful than traditional projects and almost 86% software developers practice Agile in their work. 71% of companies admitted to using Agile approaches.

Fig. 2. Agile Mode



Comparison

Table 1. Difference between the models

Agile	Waterfall
Seperates the project development Lifecycle into sprints.	Software development process is divided into distinct phases.
Incremental approach.	Sequential approach.
Flexible.	Rigid.
Considered as many different projects.	One project.
Changes of project requirements can be changed even though planning has been completed.	No scope of changing the requirements once development has started.
Development is a process in which the requirements are expected to change.	Ideal for projects that have definite requirements and no changes expected.
Testing is performed concurrently.	Testing comes after build phase.
User can request changes.	User can't request changes. Resulting in unsatisfactory project output.

Conclusion

Conclusion has been made to use the Agile methodology due to its flexibility and iterations to postpone procrastination while working on this project. Acquire motivation from visibility of progress when im-

plementing features and testing them. It would make the project more flexible and allow the manipulation of requirements and prototyping. Between the sprints, feedback from the supervisor could be acquired on the implemented features.

2.4 Version Control

Version control or also known as Source Control is the practice of tracking and managing changes to development code. For my version control I have chosen to use GitHub that provides version control of code using Git. The project repository was set up on GitHub and used throughout the development process. To commit my code changes and back-up my code frequently I used GitHub Desktop which is an application that allows for easier committing than using the command prompt.



3 Technology Review

To tackle the project, numerous tools and technologies could be used for a solution. This chapter outlines the different technologies that are used throughout the project. Advantages and disadvantages, and choice decisions are outlined why chosen technologies have been used. LAMP stack has also been considered but since the operating system the development will take place on is Windows this was not a viable option. MERN stack has been considered too.

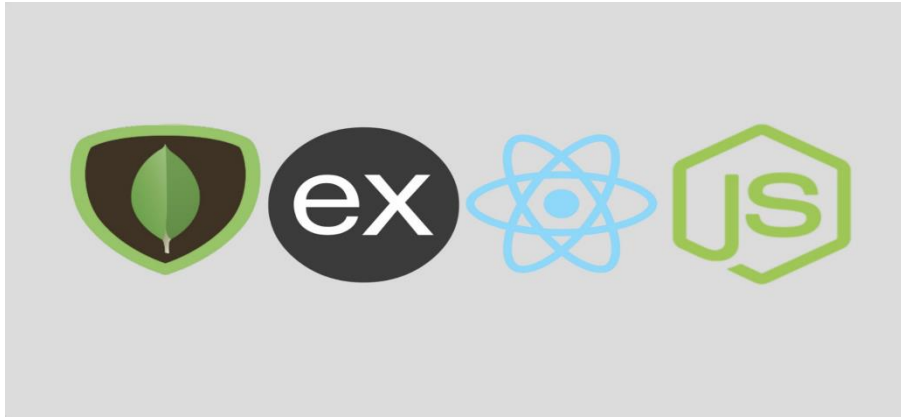
3.1 MEAN versus MERN Stacks



MEAN[10] is a combination of technologies that make it simple and quick to create web applications. It is a combination of open source JavaScript based technologies. Essentially it is a single language development stack. To get the most out of it, in depth knowledge of JavaScript must be acquired.

MEAN stack stands for:

- MongoDB
- Express
- Angular
- Node Js



MERN stack is a JavaScript technology-based stack that is used for fast and easy deployment of full-stack web applications. It provides front-end to back-end development components.

MERN stands for:

- MongoDB
- Express
- React
- Node Js

MEAN stack [10] has been in the market for a longer time compared to MERN. Both stacks have their own share of benefits which they provide to their users. MEAN stack enforces MVC (Model, View, Controller) like design in comparison to MERN overall design is unstructured and more ambiguous. React applications are harder to maintain. MERN[11] is slowly gaining popularity over the recent years.

I have chosen to use the MEAN stack for this project as it is highly flexible, easy to switch between client and server, a right choice for real time web applications and it is time saving. Project can be completed within the given time slot.

3.2 Research Competition

Before beginning to code, research for competitors and inspirations had been conducted. OsmAnd and Waze are two web applications that integrate customized google maps and focus on it as the core feature. I took inspiration from these two applications.

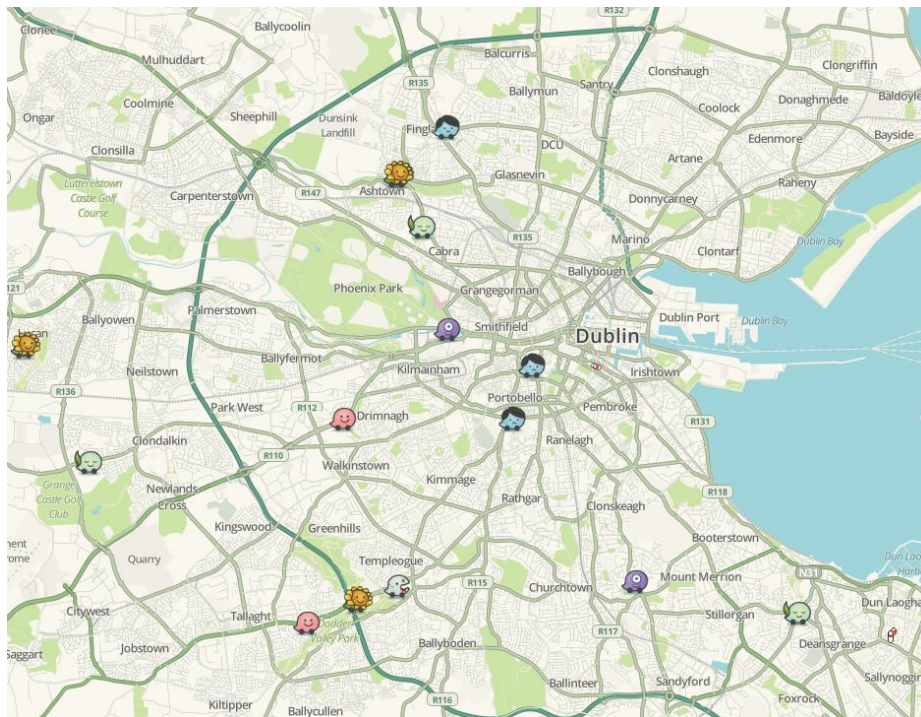
OsmAnd is a different type of navigation application. It caters directly to people who want offline maps. You can download maps that show

different maps with different points of interest depending on what the user specifically needs. To use the application, you do not need an online connection.

However, Waze is different, Waze is built on the same core feature of using customized Google Maps but requires internet connection to be used. The web application is a carpooling option for commuters that want to get to their destination to save a few bucks. It uses a live map to show commuters who are signed up on the application and are willing to bring you along on their trip.

Researching competition showed that there is a target audience and google maps orientated applications are popular amongst commuters and tourists.

Fig. 3. Waze Live Map



3.3 Chosen Technologies

Following research, the project was decided to use the MEAN stack as it fits the objectives and scope of the project. This section consists of the technologies, tools, aiding technologies, frameworks and languages that have been implemented and deployed in the project.

Angular

Angular[1] is a leading framework for building JavaScript heavy single page based web applications. Single page applications (SPA's) load the entire content of a site within a single page. Popularity of SPA web applications has taken off due to how fast they are. Allows to write more succinct code then using plain JavaScript. The angular documentation allowed to implement the MEAN stack sufficiently.

Table 2. Advantages and Disadvantages of Angular

Advantages	Disadvantages
MVC architecture implementation.	Learning curve, easy to learn basics, hard to master.
Enchanted Design Architecture, helps programmer locate and develop code without difficulty.	Weak documentation.
Custom directives that improve HTML functionality (Ng directives).	Complex directives, hard to understand what you are writing.
Two-way data binding.	

MongoDB

MongoDB[5] is a non-relational data store for JSON documents. JSON stands for JavaScript Object Notation. Data that is stored in JSON documents consist of a key and a value: {"**name**": "**Tomas**"}, name is a **key** and Tomas is a **value**. These JSON documents are stored within MongoDB. What makes them non-relational is that the documents contain hierarchy. Relational data storages are a series of tables with column names and rows of data. The fundamental fact that MongoDB stores documents is what makes it easy to operate and program

with. MongoDB is ideal for scalable projects or vast amounts of data such as registered user information[5]. This is one of the major reasons I have chosen to use MongoDB for the project.

Table 3. Advantages and Disadvantages of MongoDB

Advantages	Disadvantages
Flexible Database.	High Memory usage.
High speed.	Limited Nesting.
Scalability.	Joins not supported like relational databases.
Easy Environment Setup.	

ExpressJS

Express[3] is a node.js web application framework that provides a robust set of features for web applications and APIs. Express forms the backend cluster for the MEAN stack, it is responsible for interactions between the front-end and the database. It is the most popular framework that is used with node.js. The purpose of Express is to make sure the developer does not have to repeat the same code repeatedly. It allows for easy routing and server development with a lot less code.

Table 4. Advantages and Disadvantages of ExpressJS

Advantages	Disadvantages
Easy to learn.	May be difficult to understand callback nature.
Easy integration.	
You can use the same language to develop your server and application.	

NodeJS

NodeJS[2] is a JavaScript runtime environment. The Node.js runtime environment includes everything that is essential to execute and run a program written in JavaScript/Typescript. It is the spine of the MEAN stack. Using asynchronous events, NodeJS allows processing of multi-

ple connections simultaneously which makes it ideal for cloud-based applications.

Table 5. Advantages and Disadvantages of NodeJS

Advantages	Disadvantages
Easy Scalability.	API is not stable, backwards-incompatible changes occur and developers are forced to match compability. Does not have a strong library support system in comparison to other programming languages.
Easy to Learn.	
Used as a Single Programming Language, allows front-end and back-end to be JavaScript.	
High performance.	

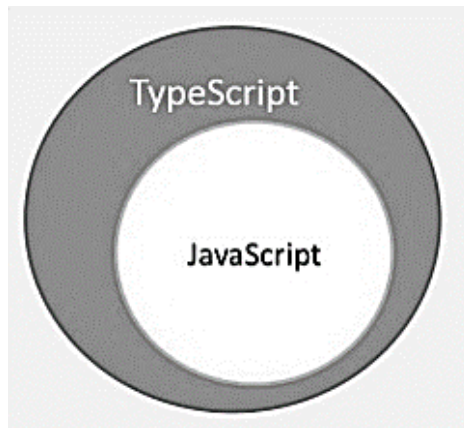
3.4 Languages

The languages that have been chosen to use in the MEAN stack web application are: TypeScript, JavaScript, HTML and CSS. This section will showcase a short overview of the languages that were incorporated into the project.

Typescript

Typescript[12] is JavaScript for application-scale development. It is a strongly typed, object orientated and compiled language. It is a language and also a set of tools. It is a superset of JavaScript. In other words, TypeScript is JavaScript with extra features added on top. It is more developer friendly than JavaScript and functional errors mostly do not forbid compilation which is desired for large applications. TypeScript helps IDE's provide a richer environment for spotting common errors as the developer types the code. TypeScript is portable across browsers, devices and operating systems. TypeScript does not need a dedicated virtual machine or a specific runtime environment to execute unlike its counterparts. This is the main language of the project.

Fig. 4. Typescript



JavaScript

JavaScript [4] is a very powerful high-level client-side scripting language, which means it runs on the user's computer. It is one of the most rapidly growing languages in the world. JavaScript is the spine behind the creation and functionality of dynamic website content. JavaScript can calculate, manipulate and validate data. JavaScript adds behavior and interactivity within the website content. It is used for interaction within the project's web application.

HTML

HTML stands for Hypertext Markup Language which is the standard markup language for documents that are designed to be displayed in a web browser. HTML is assisted by technologies such as CSS (Cascading Style Sheets) and scripting languages (JavaScript). HTML is used throughout the front-end of the project.

CSS

CSS stands for Cascading Style Sheets which is a style-sheet language that is used for in the description of a document that is written in a markup language such as HTML. It is used to add style to webpages

or documents. When CSS is used, a website's standardized elements such as fonts, colors and spacing are all controlled by the CSS file. The significance behind CSS lies in its ability to separate documents content from its presentation. This means the look and feel of the website can be tweaked without changing the content on the page. CSS has a very sky skill ceiling, which means it is easy to begin with but hard to master. CSS is used in this project to form and shape the content within the webpages to make it aesthetically appealing to the eye and follow the color scheme throughout the website.

3.5 Programming Tools

In this section, I showcase the programming tools that have been used to maintain, develop and monitor the project during the development period. They enhanced the procedure of developing this project.

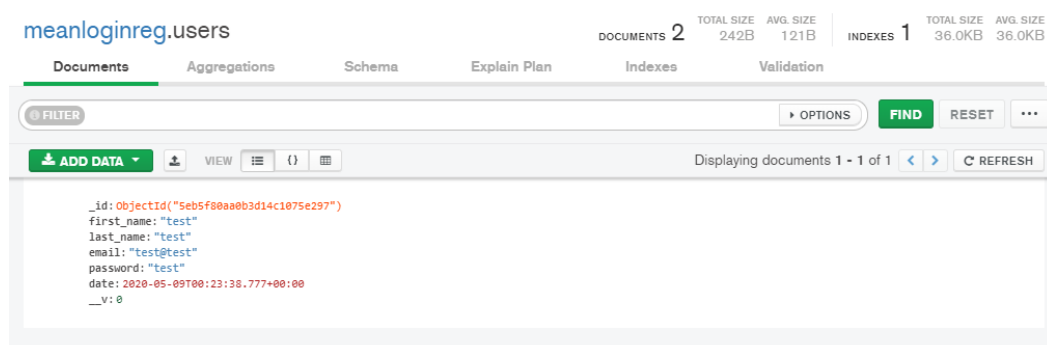
Visual Studio Code

Visual Studio Code is a free source-code editor that has been developed by Microsoft for Windows, Linux and macOS operating systems. It has support for debugging, syntax highlighting and intelligent code completion. Visual Studio Code is a personal favorite and has been used in my previous projects that I have undertaken. This is a main reason that I chose this programming tool for this project. The debugging of all the above-mentioned programming languages is made easy in Visual Studio Code.

MongoDB Compass

MongoDB Compass is a GUI for MongoDB. It helps to visualize the data within the projects MongoDB database. It offers CRUD functionality on the data. In this project it is used to visualize the data.

Fig. 5. MongoDB Compass



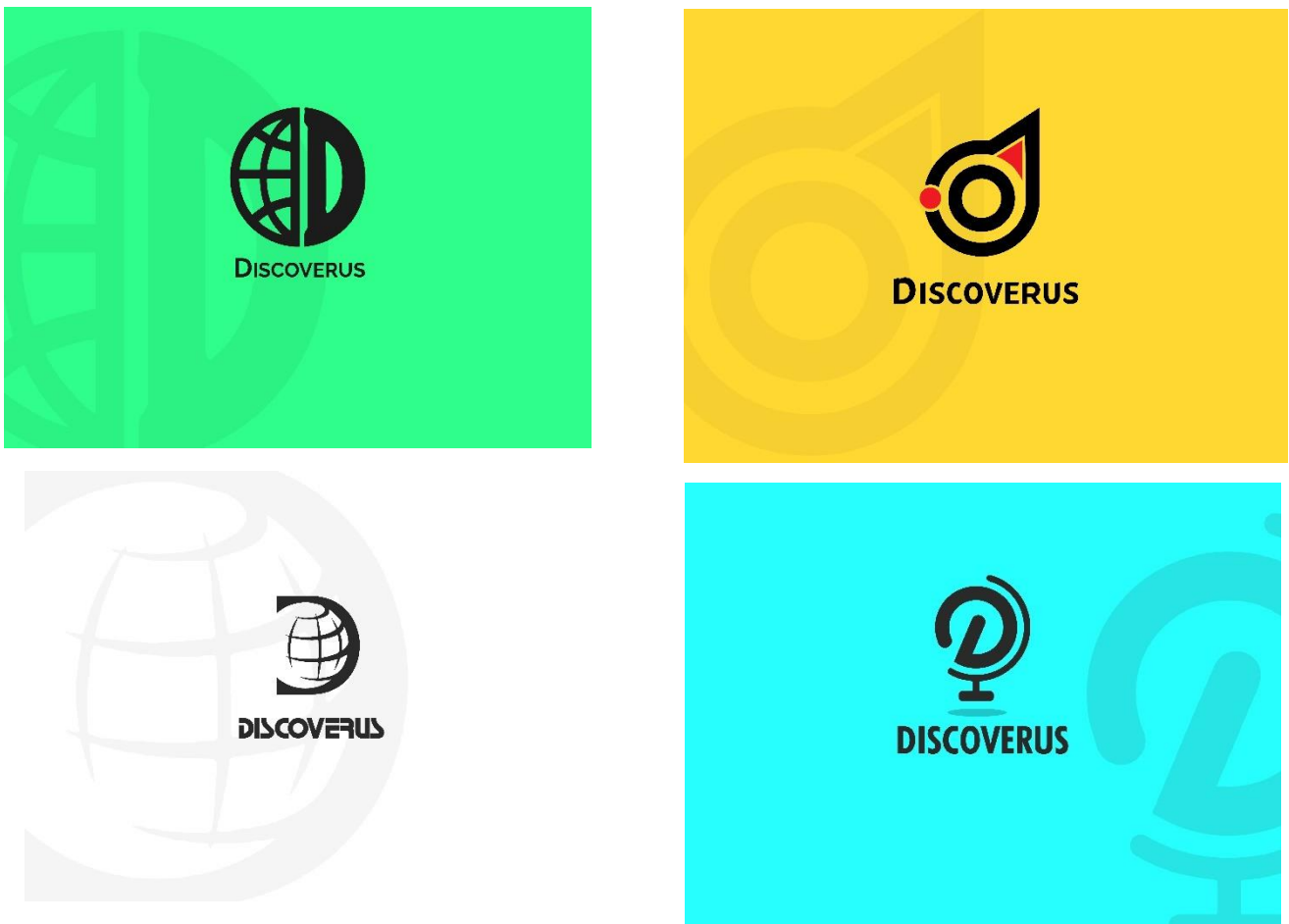
3.6 Other Tools

In this section, tools that have assisted me during this project that are not programming related are showcased.

Adobe Photoshop CC 2015

Adobe Photoshop is a graphics editor developed and published by Adobe Inc. It is used to create or edit graphics. Adobe Photoshop is used to create the logo for this project. A web application has to have a strong logo to assist its functionality. Four potential logos were produced. Reached a decision to use the yellow background logo due as it best describes and backs up the web application and its main functionality using google maps.

Fig. 6. Discoverus Logos

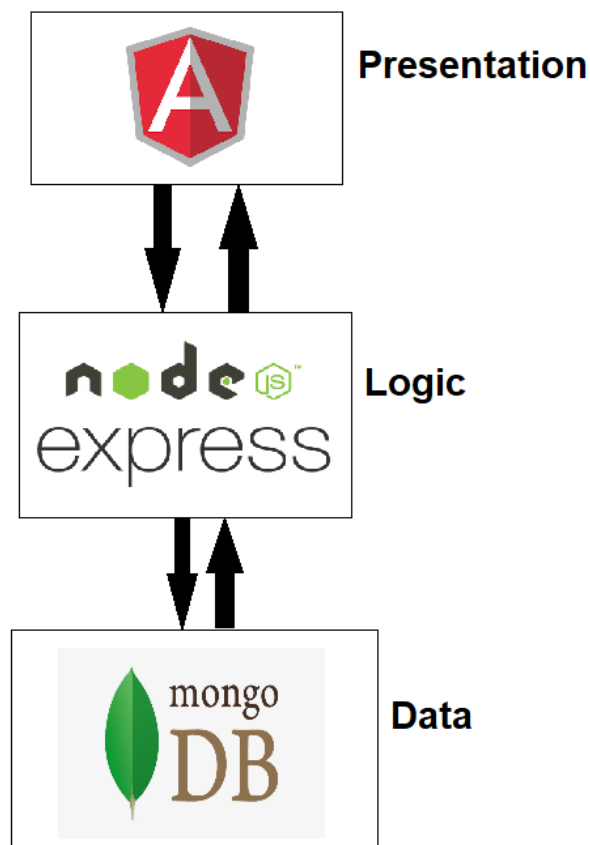


4 System Design

This chapter discusses the system design of the project, analyzing various aspects of the project and defines the purpose of each component from back-end to the front-end.

4.1 Overview

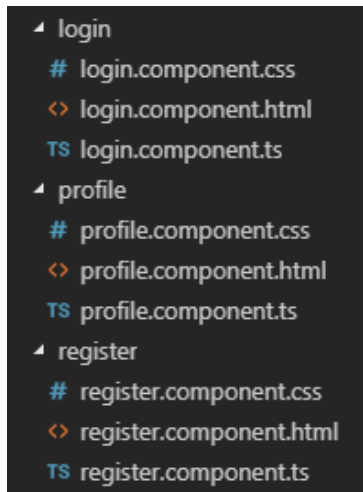
This project is structured using the multi-tier architecture (full-stack development). This is a client-server architecture. The web application of the project is separated into three different tiers. The presentation layer: which is what the user sees in the browser, logic (application) layer; which holds the application logic and data layer; which stores the application's data. While each of these layers work together to make a single application, they also operate independent of one another. The web application which represents the presentation layer, allows the user to visualize the content on the webpages such as login, register, welcome page, dashboard and google maps feature. NodeJS and ExpressJS represent the logic layer. They handle the return and parse requests of user information, log in and registration. MongoDB represents the data layer, which stores user data.



4.2 User Registration and Login – Back End

The development began with the initial folder setup of all components that will consist in the web application. Dummy components consisting of mock-up of HTML forms were set up inside the client to test and develop the user registration and login system with authentication.

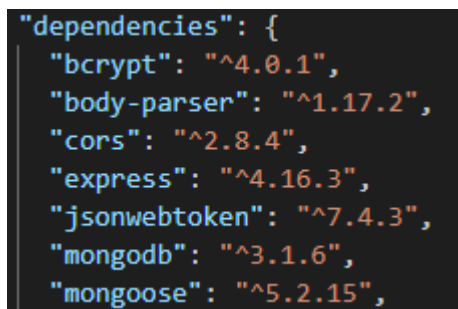
Fig. 7. Initial Component set up



Dependencies

Collection of project dependencies are stored in the package.json file. These dependencies are the modules that the project relies on to function properly. Having dependencies in this project's package.json allows the project to install the versions of modules specified that it needs to correctly run. When the user runs `npm install` from the root folder of a given module, it will install any modules listed in the `dependencies` hash.

Fig. 8. List of dependencies of Discoverus



- **Bcrypt** – This is used for hashing and encrypting passwords to the database.
- **Body-parser** – This is for extracting data from objects like forms or JSON and make it a JSON like structure to be easily used in the application.
- **Cors** – Used this to make sure the front-end and the back end can interact with each other.
- **Express** – Used for implementing middleware. Used in the backend.
- **Jsonwebtoken** – Used for hashing user data so it can be stored all in and guarding certain routes.
- **Mongodb** – Used to interact with the mongodb database.
- **Mongoose** – Shortens server code and makes it easier.

Model

The model that contains the schema of users is stored in `models/user.js`. Import mongoose module and creating an object called `Schema` which is from mongoose. The model file describes the structure the data will be saved as in the database and can be used anywhere in the web application. The required field can be set to true or false. In this instance the user has to enter all information when trying to register to use the web application. `Default: date.now` logs the creation of the user on the database. Exports the user model so that other JavaScript files can see it data with the name of users.

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema
const UserSchema = new Schema({
  first_name: {
    type: String,
    required: true
  },
  last_name: {
    type: String,
    required: true
  },
})
```



```

    email: {
      type: String,
      required: true
    },
    password: {
      type: String,
      required: true
    },
    date: {
      type: Date,
      default: Date.now
    }
  }
})

module.exports = User = mongoose.model('users', UserSchema)

```

Routes

This folder takes care of the server routing. Declare the `SECRET_KEY` for jsonwebtoken as it a required parameter for hashing data. Everytime the user submits the form of register this method will get invoked with request and result parameters. Declaration of today's date also happens inside this method. `Userdata` which came from the requested body which pertains to the form of the front end in the register HTML file of the component.

```

process.env.SECRET_KEY = 'secret'

users.post('/register', (req, res) => {
  const today = new Date()
  const userData = {
    first_name: req.body.first_name,
    last_name: req.body.last_name,
    email: req.body.email,
    password: req.body.password,
    created: today
  }
})

```

This `findOne` method which will find one row of data. Find the email with the value of the submitted from the form of the client side. Then if there is no user found with that email then the user will be created using the `userData`. Declaring a payload will contain the data retrieved from the mongoDB database. This then generates a token which is used in the front end. Method finishes off catching errors if a user already exists.

```
User.findOne({
  email: req.body.email
})
.then(user => {
  if (!user) {
    User.create(userData)
      .then(user => {
        const payload = {
          _id: user._id,
          first_name: user.first_name,
          last_name: user.last_name,
          email: user.email
        }
        let token = jwt.sign(payload,
          process.env.SECRET_KEY, {
            expiresIn: 1440
          })
        res.json({ token: token })
      })
      .catch(err => {
        res.send('error: ' + err)
      })
  } else {
    res.json({ error: 'User already exists' })
  }
})
.catch(err => {
  res.send('error: ' + err)
})
})
```

Every time the user submits a form for the login part of the user authentication, it will be directed to this route. Like the /register part it will also have the request and result parameter. Find the email with the value of the submitted from the form of the client side. This then generates a token which is used in the front end to login. And in the front end it will get the token from sessionStorage of the application and decodes to display on the Profile page component. Else clause if the user does not exist, the client side picks that up.

```
users.post('/login', (req, res) => {
  User.findOne({
    email: req.body.email
  })
  .then(user => {
    if (user) {
      const payload = {
        _id: user._id,
        first_name: user.first_name,
        last_name: user.last_name,
        email: user.email
      }
      let token = jwt.sign(payload,
        process.env.SECRET_KEY, {
          expiresIn: 1440
        })
      res.json({ token: token })
    } else {
      res.json({ error: 'User does not exist' })
    }
  })
  .catch(err => {
    res.send('error: ' + err)
  })
})
```

For the profile route it also contains request and result parameters. Then it decodes the authorization option that is sent from the front end part. The token that is stored in sessionStorage is sent to this route decoded and verified with `jwt.verify`. It gets the data from the token and uses in this method to retrieve the data from the database. Then the data is displayed on the user profile page.

```
users.get('/profile', (req, res) => {
  var decoded = jwt.verify(req.headers['authorization'],
process.env.SECRET_KEY)

  User.findOne({
    _id: decoded._id
  })
    .then(user => {
      if (user) {
        res.json(user)
      } else {
        res.send('User does not exist')
      }
    })
    .catch(err => {
      res.send('error: ' + err)
    })
  })
})
```

Express Server

Express, cors, body-parser, mongoose is imported into the server. Express is initialized with `var app = express()`. The middleware is mounted, and a port is set. MongoDB's database URI is set with a port of 27017. Server is hosted locally by typing `nodemon server.js` in your command prompt when the user has navigated to the root project folder. Mongoose is used to connect accepts the mongoDB URI, if successful it will log it in console or any errors. The last thing it does, it listens for the port and starts the server.

```

const mongoURI = 'mongodb://localhost:27017/meanloginreg'

mongoose
  .connect(mongoURI, {useNewUrlParser: true})
  .then(() => console.log("MongoDB connected"))
  .catch(err => console.log(err))

var Users = require('./routes/Users')

app.use('/users', Users)

app.listen(port, function () {
  console.log("Server is running on port: " + port)
})

app.use('/users', Users)

app.listen(port, function () {
  console.log("Server is running on port: " + port)
})

```

Fig. 9. Server running locally in command prompt

```

C:\Users\zyrxe\Documents\GitHub\Finalyearproject>nodemon server.js
[nodemon] 2.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
(node:29668) DeprecationWarning: current Server Discovery and Monitor
future version. To use the new Server Discover and Monitoring engine,
ngoClient constructor.
Server is running on port: 3000
MongoDB connected

```

4.3 User Registration and Login – Front End

Authentication Service

```
import {Injectable} from '@angular/core'
```

Injectable are decorators that are used in development of services. Services are also referred as dependencies in Angular which are objects that the class needs to perform its process or functions.

```
export interface UserDetails {  
  _id: string  
  first_name: string  
  last_name: string  
  email: string  
  password: string  
  exp: number  
  iat: number  
}
```

The code above is a declaration or a representation of the data from that will be passed around from the client side to the server side.

The `getUserDetail` method decodes the token to JSON. Retrieves the token from sessionStorage and checks if a token exists and splits it, `.atob` decodes base64 encoded string, use it and parse it to JSON which then can be used.

```
public getUserDetails(): UserDetails {  
  const token = this.getToken()  
  let payload  
  if(token) {  
    payload = token.split('.')[1]  
    payload = window.atob(payload)  
    return JSON.parse(payload)  
  }else{  
    return null  
  }  
}
```

`Login()` passes the user payload to the url `/users/login` using the post method. It queries the user that matches the token payload. The pipe method waits for the past data to be available and then maps past value. If there is a response the token is saved into sessionStorage. The same

logic applies to the `Register()` method in this service but with different url (`/users/register`). The `isLoggedIn` method returns whether the user is logged in or not throughout the application.

```
public login(user: TokenPayload): Observable<any> {
    const base = this.http.post('/users/login', user)

    const request = base.pipe(
        map((data: TokenResponse) => {
            if(data.token){
                this.saveToken(data.token)
                alert("You have successfully logged
in.");
            }
            return data
        })
    )
    return request
}
```

The `logout()` method removes the user token in `sessionStorage` and goes back to the default url which is the Welcome component in the web application.

```
public logout(): void {
    this.token = ''
    window.sessionStorage.removeItem('usertoken')
    alert("You have successfully logged out.");
    this.router.navigateByUrl('/')
}
```

Auth-guard Service

The auth-guard service also contains an injectable import because it is a service too. Router is also imported so that the service can route to other views.

```
import { Router, CanActivate } from '@angular/router'
```

`CanActive` checks if an user can 'visit a route'. It is a built-in angular guard that guards a view and checks if an user can access the view. This is used with the `canActive()` method. Checks if the user is logged

in, and if the user is not logged in, returns to the empty path which is the welcome page in the web application.

```
@Injectable()
export class AuthGuardService implements CanActivate {
  constructor(private auth: AuthenticationService, private
router: Router) {}

  canActivate() {
    if (!this.auth.isLoggedIn()) {
      this.router.navigateByUrl('/')
      return false
    }
    return true
  }
}
```

Angular Routing

The application has basic angular routing with its corresponding path and component. Home page, login page, register page, blank url which is the welcome page and profile page that runs the AuthGuardService. Module.ts file also contains all the imports that are essential for the web application. These consist of Angular Google Maps modules that are needed for the home page view which is covered in the next section.

```
const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: '', component: WelcomeComponent },
  {
    path: 'profile',
    component: ProfileComponent,
    canActivate: [AuthGuardService]
  }
]
```

```
imports: [
  BrowserModule,
  GooglePlaceModule,
  AgmDirectionModule,
  AgmOverlays,
  AgmCoreModule.forRoot({
    apiKey: 'AIzaSyCQ9ahuGv8XqvSws7q6pQxMD7xnVhokzu8',
    libraries: ['places', 'geometry']
  }),
  FormsModule,
  HttpClientModule,
  RouterModule.forRoot(routes)
],
providers: [AuthenticationService, AuthGuardService],
bootstrap: [AppComponent]
```


4.4 Views

This section will display the numerous page views of Discoverus and the functionality of each page. A navigation bar is constantly present at the top of the web application. It is dynamic and the contents of it are managed by the logged in state of the user. The Home and Profile pages can only be seen upon a successful login. The Register button is changed to Logout after a successful login.

When the user is logged out/not registered:



When the user is logged in:

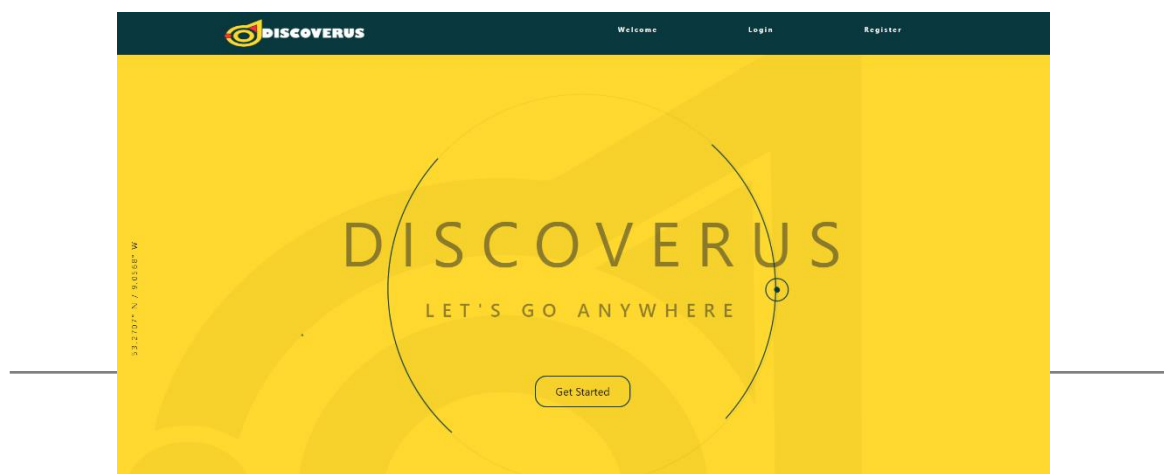


The Home and Profile pages can only be seen upon a successful login.

Welcome Page

The user is prompted with the welcome page upon first ever visit or successful logout. The user is also directed to the welcome page when the user fails to fill in a sign in/register form. The view consists of a button and “Get Started” which re-directs to the Login page. The rest of the view is animated CSS code using divs that represent rotating ellipses, this can be seen in the video demonstration. The use of @keyframes animates the divs rotations.

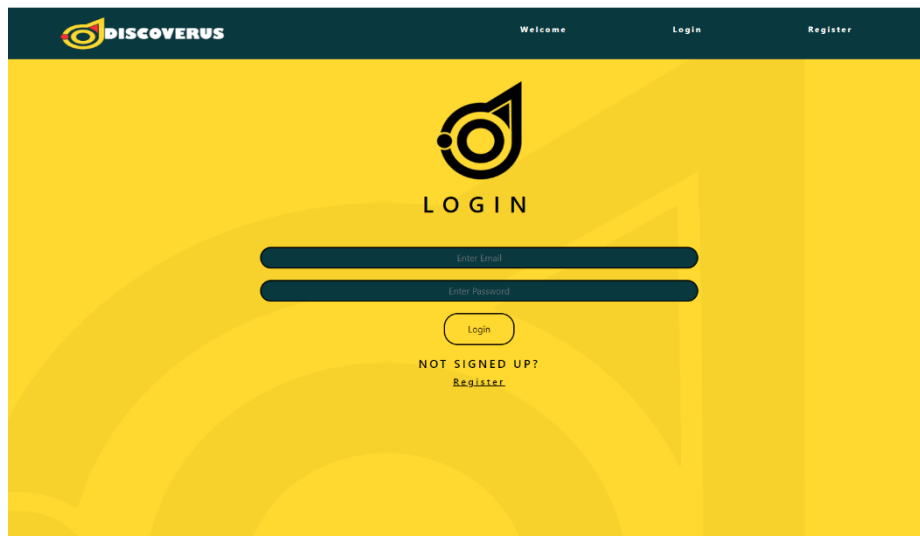
Fig. 10. Welcome Page



Login Page

Either the user pressed the “Get Started” button on the welcome page or pressed the Login link on the navigation bar to navigate to the login page it is presented the same. If the user does not have an account, the user can navigate to the Register page from the link below the text “Not Signed Up?”.

Fig. 11. Login Page



When the user enters correct information corresponding to their user information on the database an alert will pop up telling the user he has successfully logged in and will be redirected to the Profile Page. If the user enters the wrong information a corresponding error will be shown and will be redirected to the welcome page. nGModel is a directive that is used in the forms to bind the input to data and compares it to the token to see if a match is possible.

```
<input type="email" class="form-control" name="email"
placeholder="Enter Email" [(ngModel)]="credentials.email">
```

Register Page

The Register page is almost identical to the login page. The input from the forms is binded using the nGModel directive to the token and

the token stores a new user in the data if no error occurs from the user's inputs. The user is then redirected to the Welcome Page where the user must log in using the newly created account.

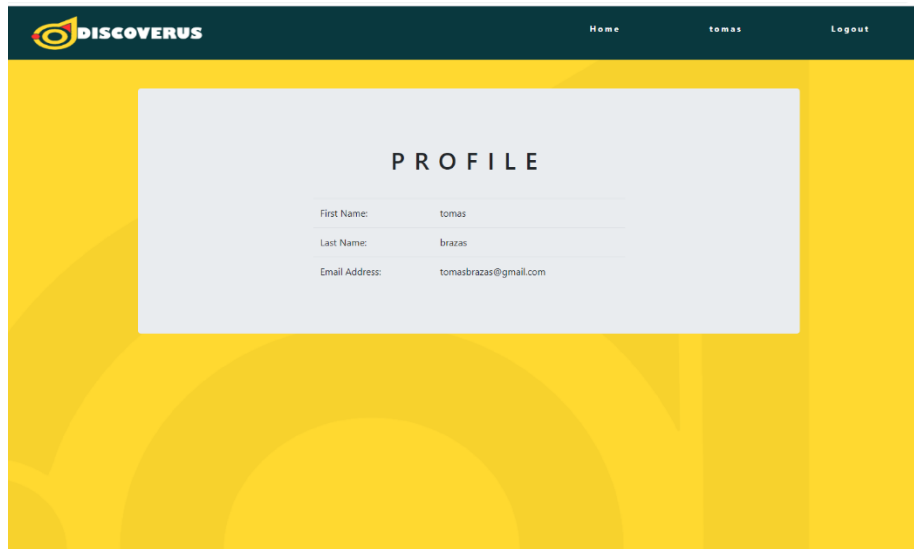
Fig. 12. Register Page

The screenshot shows the Register Page of the DISCOVERUS application. The page has a yellow background with a dark blue header. The header contains the DISCOVERUS logo on the left and 'Welcome', 'Login', and 'Register' links on the right. The main content area features a large 'REGISTER' title, a stylized logo, and five input fields for 'First name', 'Last name', 'Email address', and 'Password' (with a second password field). A 'Register' button is at the bottom.

Profile Page

The profile page outputs the users details that the user registered with. This consists of the first name, last name and an email. The data is displayed in the `ngOnInit` method which is a lifecycle hook called by Angular to indicate that Angular is done creating the component. This retrieves the data from the database and outputs it on the page in the web application.

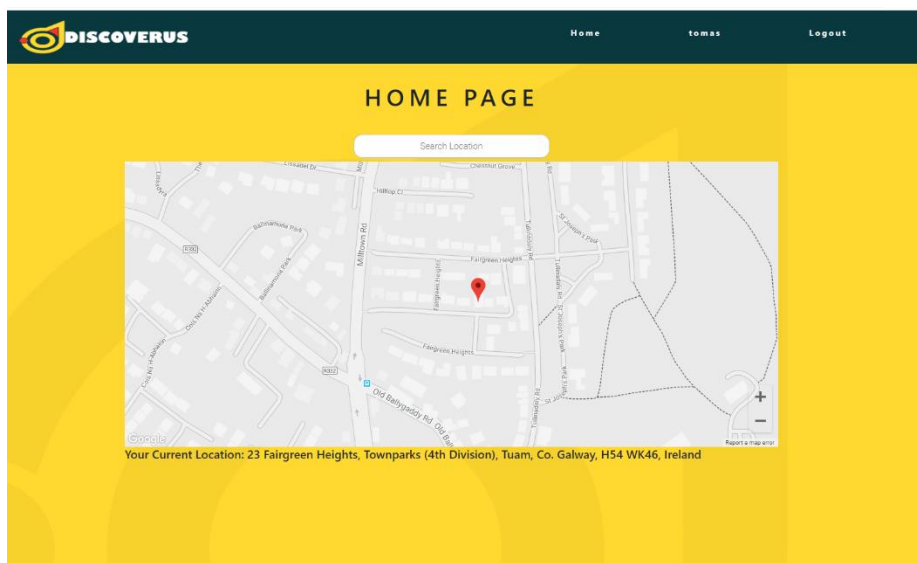
Fig. 13. Profile Page



Home Page

The Home page consists of a Google maps component in a div, a location search bar with autocomplete feature and your current location that is tracked at the bottom, which is updated in real time. The red bouncing marker represents the user's location within the map. Further information is present in the next section.

Fig. 14. Home Page



4.5 Google Maps

Google Maps also known as AGM is used in angular to represent a google maps map. Google Maps can only be deployed with access to their Google Maps API which contains a unique key. The key can be generated at their website and has a limited free use. It is defined by the `<agm>` tags inside HTML. The street-view control is turned off, custom styling has been applied to the map that is present in `home.component.ts` file. The latitude and longitude of the by default is the user's location. Zoom feature is hardcoded, it is responsible for how much zoom is applied to the map initially.

```
<agm-map [streetViewControl]="false" [styles]="mapStyles"
[latitude]="latitude" [longitude]="longitude"
[zoom]="zoom">
```

The user's location is tracked by using geolocation. This is the process of identifying the geographical location of a device by means of information processed by the internet. This method asks the user's device location and sets the variables accordingly to represent on the map and in text below the map.

```
private setCurrentLocation() {
  if ('geolocation' in navigator) {
    navigator.geolocation.getCurrentPosition((position)
=> {
      this.latitude = position.coords.latitude;
      this.longitude = position.coords.longitude;
      this.zoom = 16;
      this.getAddress(this.latitude, this.longitude);
      this.dir = {
        origin: { lat: this.latitude, lng:
this.longitude}
      }
    });
  }
}
```

The AutoComplete feature is done by loading the map API. Setting the autocomplete variable to take input from the search box. When a search reference is entered, the ngZone injectable service is used. It is called to

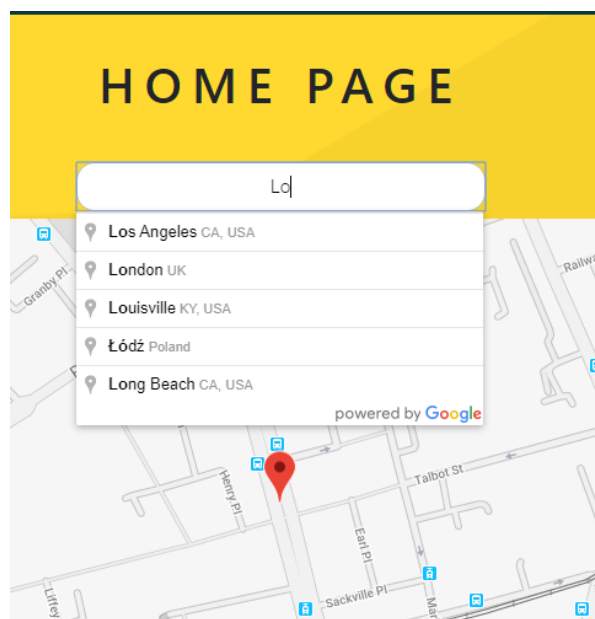
optimize performance when the asynchronous task begins. The place is set according to the searched reference, it is verified using geometry. Longitude and zoom are adjusted to the new location, this is done so that the marker knows where to be put down at.

```
let autocomplete = new
google.maps.places.Autocomplete(this.searchElementRef.nativeElement);
autocomplete.addListener("place_changed", () => {
  this.ngZone.run(() => {
    //get the place result
    let place: google.maps.places.PlaceResult = autocomplete.getPlace();

    //verify result
    if (place.geometry === undefined ||
place.geometry === null) {
      return;
    }

    //set latitude, longitude and zoom
    this.latitude = place.geometry.location.lat();
    this.longitude = place.geometry.location.lng();
    this.zoom = 16;
  });
});
```

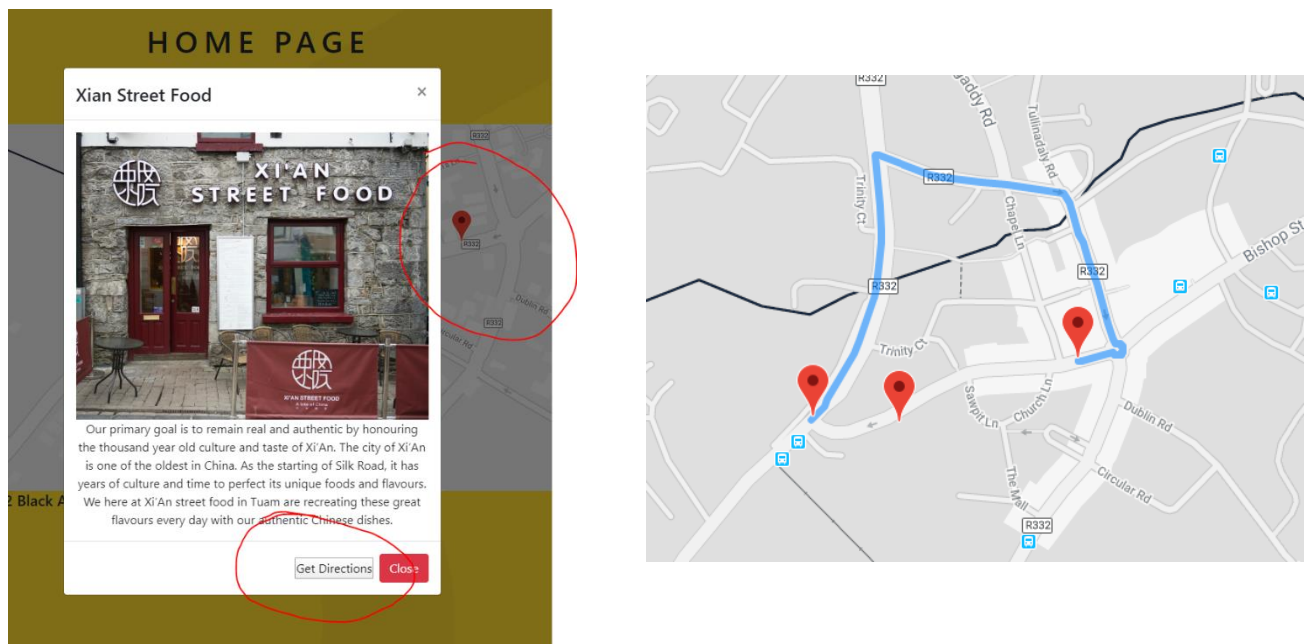
Fig. 15. Autocomplete feature



Directions

For the user to get directions, the user can click one of the available markers (currently only available in the Tuam area) and a modal will be showcased with a Directions button and information about the venue.

Fig. 16. How to get directions



Directions are deployed by the `<agm-direction>` tags in HTML and setting an origin and destination values. The origin location is always tracked with the user's marker and destination is set by a method that contains the longitude and latitude of specific locations.

```
<agm-direction
  *ngIf="dir" [origin]="dir.origin"
  [destination]="destination"
  [renderOptions]="renderOptions"
></agm-direction>
```

5 System Evaluation

The aim of this project was to develop a web application that is user friendly, easy to use and understandable. It explores the local area with pre-defined locations set by the developer that are worth visiting. The main objectives of the project were:

- Design and develop a user-friendly application that is easy to use and understand.
- Integrate state of art technologies focusing on Google Maps.
- Apply and practice a software development methodology with the project.

5.1 Evaluation

- Design and develop a user-friendly application that is easy to use and understand.

The design of the web application is to be minimalistic and simplistic while still able to fulfill the user's needs within the web application; to find local areas to visit. The user can log in and logout with correct authentication and guarding. The user's data is safely hashed and cannot be accessed for malicious actions. The design and feel of the web application is modern and designed to be able to stand out from its competition in the tourism and travel market.

- Integrate state of art technologies focusing on Google Maps.

Google Maps integration is highly increasing amongst developers as maps can be very interactive and can be simplified to be easily readable such as this web applications map. The clutter is removed to deliver precise and clear directions to the target's destination and surrounding area. The map is fully navigational but currently only Tuam area. Co. Galway consist of markers. Further development of the application can

lead to clusters of markers all with their corresponding data and directions.

- Apply and practice a software development methodology with the project.

Research of software development methodologies has been studied to rule out a winner and apply it to the project. The Agile software development methodology has been applied to the project. Sprints have been the weekly supervisor meetings with the expectation of work completed and incorporated every week to lead to the final solution of the project.

5.2 Testing

The web application has only been tested by users within my household and feedback has been obtained about the functionality and UI design. There is no Karma testing implemented.

5.3 Limitations and Opportunities for Improvement

Testing should have been carried out by me. Unit testing such as using Postman to test the user login register system and System testing such as Selenium.

Lack of additional functionality, adding features to edit the profile, more markers, displaying the distance to destination. These are all features that can be implemented to enhance the overall functionality of the web application if to be considered as a competitor in the travel and tourism application market. Applying firebase to manage marker data should have been considered.

The responsiveness of the web application could have been considered throughout development. Some of the pages are responsive but it

is not enough to be a stand-alone application. The responsiveness of a web application is to be considered for further development to open the application to the mobile market.

Overall, the web application functions as intended, offering venues around a local area and able to retrieve directions. It has a login system that is fully guarded and is linked to a database. The importance of testing has been overlooked, and it is clearly represented in the project. The learning outcome has been established and that testing should not be dismissed, this will be applied for future projects.

6 Conclusion

The goal of the project was to create a navigational web application and acquire knowledge using modern technology such as the MEAN stack and google maps API. In conclusion, I feel like the web application has mainly been successful especially in terms of design, this is for sure great practice to be a front-end developer as it is something I am interested in the future career. The application's aim was to help tourists discover the local area using a customized map. With some extra development the project can be a successful one and with the right marketing to tourists.

6.1 Learning outcomes

During the planning and development of the project, I have acquired numerous learning outcomes.

- The Use of New Technologies.

The MEAN stack is well known to most developers, but I had next to none experience with the environment and how everything operated until research had been concluded. Trial and error were the success of bug fixing especially with AGM as the documentation is very limited. Most of the features that are associated with google maps have short documentation and there is no established community as of yet. I am glad I undertook this project as it refreshed my skills on HTML, CSS and JavaScript which I have not practiced in a long time. The development process was also different compared to my go-to Unity projects that is more so less structured.

Practicing documentation analyzing and applying it to code has helped me understand the MEAN stack and how the components are wired.

- Testing

The importance of testing has been overlooked by me. It is something I will strongly consider when undertaking a project such as this.

Components were not working during the user login system and accurate testing could have saved me hours of work with the use of Postman.

6.2 Further Development of Discoverus

I will be revisiting this project at some later stage to implement more features and apply testing as the room for growth is still very possible even though the objectives have been met. Here are some additions that could be implemented in the future versions of the web application:

Adding a Location to Favourites

Upon opening a modal, a favourite button could be present where the user can favourite the location and corresponding data can be presented at the Profile Page. These could be deleted from the Profile Page

Profile Customization

Allow user to edit their information. Add a picture, show case their favourite locations to other users within the database.

Allowing users to interact between each other

Perhaps adding some sort of Chat API into the application that would let users communicate between each other and add each other as friends.

Mobile friendly

Make the application more responsive, therefore opening to a bigger audience. As of currently the application is not 100% responsive.

6.3 Final Conclusion

Overall, Discoverus has been a project I was glad to undertake. It was definitely not easy and a new environment. Numerous obstacles have occurred during development and planning phase but have been overcome through sheer commitment. With further progress at some later stage I believe the web application can be of high potential. I see this application being popular amongst tourists and those who are visiting from local areas. With a small-scale team unfortunately only managing Ireland would be possible.

7 BIBLIOGRAPHY

1. Angular Documentation <https://angular.io/docs>
2. NodeJS Documentation <https://nodejs.org/en/docs/>
3. ExpressJS <https://expressjs.com/>
4. JavaScript <https://devdocs.io/javascript/>
5. MongoDB <https://docs.mongodb.com/>
6. Agile https://en.wikipedia.org/wiki/Agile_software_development
7. Waterfall Model https://en.wikipedia.org/wiki/Waterfall_model
8. AGM docs <https://angular-maps.com/api-docs/agm-core/modules/agmcoremodule>
9. AGM direction <https://www.npmjs.com/package/agm-direction>
10. MEAN [https://en.wikipedia.org/wiki/MEAN_\(solution_stack\)](https://en.wikipedia.org/wiki/MEAN_(solution_stack))
11. MERN <https://www.geeksforgeeks.org/mern-stack/>
12. Typescript <https://www.typescriptlang.org/docs/home>
13. NPM <https://docs.npmjs.com/about-npm/>
14. MEAN VS LAMP <https://dzone.com/articles/what-is-the-mean-stack-and-why-is-it-better-than-l>

8 Appendix of GitHub

Source Code: <https://github.com/tomasbrazas97/Finalyearproject>