Design Document

Timetable app

Donal burke, Naqi Ahmad, Sean mcguire

Table of Contents

[1. Introduction 3](#_Toc512622894)

[2. Purpose 3](#_Toc512622895)

[3. System Requirements 3](#_Toc512622896)

[4. Technology Used and Why 4](#_Toc512622897)

[4.1 Ionic 4](#_Toc512622898)

[4.2 MongoDB 4](#_Toc512622899)

[4.3 Node.js 4](#_Toc512622900)

[4.4 Python 4](#_Toc512622901)

[5. Architecture of the Solution 5](#_Toc512622902)

[6. Design Methodology 6](#_Toc512622903)

[7. Features of the Implementation 6](#_Toc512622904)

[7.1 Database design 6](#_Toc512622905)

[7.2 Server design 8](#_Toc512622906)

[7.2.1 Node server 8](#_Toc512622907)

[7.2.2 Python web scraper 9](#_Toc512622908)

[7.2.3 Server management & security 9](#_Toc512622909)

[7.3 App design 11](#_Toc512622910)

[7.3.1 Mock up 11](#_Toc512622911)

[8. Limitations 12](#_Toc512622912)

[9. Known Bugs 12](#_Toc512622913)

[10. Recommendations for Future Development 12](#_Toc512622914)

[11. Conclusions 12](#_Toc512622915)

# 1. Introduction

This application was designed to be used by the Students and Staff members of GMIT. The application lets users upon registering to select a course in the college and view their timetable, it also allows users with the appropriate permissions to Edit, Delete, and Add a module to a course.

The project architecture and systems were influenced by the MEAN software stack, a dynamic and modular approach to application design using a four tier architecture.

The main purpose of our app is to give the students and staff member of GMIT a central application for accessing and informing students of various classes going on for each specific user of GMIT’s timetable.gmit.ie website.

As a team, we realized accessing the timetable website is time consuming and navigating through various different links to get to the users timetable is difficult, so we wanted an application that would allow you to create an account, select your course, and allow the user to open the app at any time without having to enter in any more info.

# 2. Purpose

The purpose of this document is to provide a clear outline as to how the application was designed and what features that were implemented. The document should outline how the application works, what languages and technologies were used and to give the reader an understanding of the architecture of the system itself. After reading this document you should have an understanding of how this project was carried out and implemented.

# 3. System Requirements

The application runs on the following operating systems:

* Android
* iOS
* Windows phone
* Modern web browsers such as:
  + Firefox
  + Chrome
  + Safari
  + Microsoft Edge

# 4. Technology Used and Why

## 4.1 Ionic

We decided to use Ionic for the frontend of the application because it is a cross-platform framework that allows you to develop apps for a myriad of different platforms such as Android, iOS and Windows. This allowed us to spend more of our time developing in one language and also gave us the advantage of being able to deploy our app on many different platforms.

Ionic also runs on Typescript, HTML and CSS, this allowed us to develop an application that meets industry standard UI specifications because these languages have the ability to create consistent and easy to use UI’s for the everyday user.

## 4.2 MongoDB

The database solution that we chose was MongoDB mainly because of its NoSQL database format. This allowed us to develop a database that is very scalable and that’s also allows quick retrieval of documents.

MongoDB’s document data schema maps naturally to objects in application code which allowed us to implement it very easily into our node server application. Since MongoDB is compiled to BSON(Binary JSON) this allowed for quick retrieval of documents since JSON uses a key, value pair model to retrieve values allowing quick retrieval of documents by key.

## 4.3 Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side.

Much of Node’s utility comes from its large package library, which is accessible from the NPM command. NPM, the Node package manager, is part of the standard Node.js installation, although it has its own website.

One of the packages that we used for our Node.js server is the Express.js framework. Express.js is a web framework that removes a lot of the tediousness from designing a REST API from scratch with Node’s default HTTP library. Express allows you develop API endpoints with ease. Defining a route can be done in just a few lines of code.

We used Express in combination with Mongoose, another node library that enabled us to define MongoDB schema’s and model’s with relative ease. Mongoose includes built in type casting, validation, query building and more, out of the box.

In simple words, Mongoose acts as an intermediate between MongoDB and server side language(like Node).

## 4.4 Python

We used python to develop our web scraper to scrape all of the timetables from GMIT’s timetable website. The reason we decided to design a python script instead of a Node script or any other script designed in another language such as Ruby, was because python provides a myriad of libraries that makes web scraping easy and effective.

We used the BeautifulSoup4 library to manipulate the DOM and access each DOM elements and attributes easily. Another library that we used was mechanize, mechanize makes submitting forms for web applications In python extremely easy. The reason we used mechanize for the web scraper was to allow us to submit forms to GMIT’s timetable website that is built on the ASP.NET framework. ASP.NET uses forms to retrieve data from a server and load pages based on the form that was submitted. We needed a solution that would let us manipulate the ASP.NET forms to retrieve each timetable from GMIT’s timetable website.

# 5. Architecture of the Solution

##### 5.1 Outline of System Architecture

The architecture of the system can be easily described by looking at the diagram below.

1. First the python script runs and populate the Mongo database with the course timetables
2. The Node server running on Express and Mongoose query’s the database
3. Angular models are used to bind the data to Ionic views
4. Ionic displays the application and data on to devices



##### 5.2 MEAN STACK

The software design architecture that we followed was the MEAN software stack.

The reason we decided to use the MEAN stack as opposed to any other software stack such as the LAMP stack was mainly due to the fact that we have gained a good understanding of the individual components that make up the MEAN stack from our various modules that we have undertaken in college. Also using the MEAN stack enabled us to mainly develop in one language (JavaScript), this allowed us to save time as we were using a language and suite of technologies that we were comfortable with.

MEAN stands for MongoDB, Express, Angular and Node.

* MongoDB is an open source NoSQL database using a document-oriented data model. This means it relies on the architecture consisting of collections and documents instead of tables and rows (as in relational databases). Thanks to its architecture, this database is extremely scalable and can handle large volumes of data.
* Express is a lightweight server-side JavaScript framework. As a part of the MEAN stack, it is used together with Node.js to simplify the development process. It makes it easier to write secure, modular, and fast applications.
* AngularJS (or simply Angular when referring to the latest versions of the framework) has long become a “default” web frontend JavaScript framework. Built and maintained by Google, the framework allows for rapid development of dynamic, single page web apps. Its modular structure simplifies development and testing and allows for easy scalability.
* Node.js is a JavaScript-based web framework. By compiling the JavaScript source code to native machine code before the execution, it allows for building scalable and performant web apps. Node.js has a rich ecosystem of open source libraries and components, including NPM (Node.js package manager).
* Ionic enables you to develop mobile apps using web technologies and languages like HTML, CSS, JavaScript, Angular, and Typescript. Therefore, Ionic is an ideal solution for creating hybrid mobile applications. Ionic provides a collection of components that mimic the native look, feel and functionality of each mobile platform. Examples of these components include buttons, tabs, menus, lists, cards, modals, and so on.

# 6. Design Methodology

The design methodology that we followed during the development of this project was Agile.

Agile is a design framework that was originally used for software development but is now used in other areas where there is a need to complete large tasks, including project management tasks.

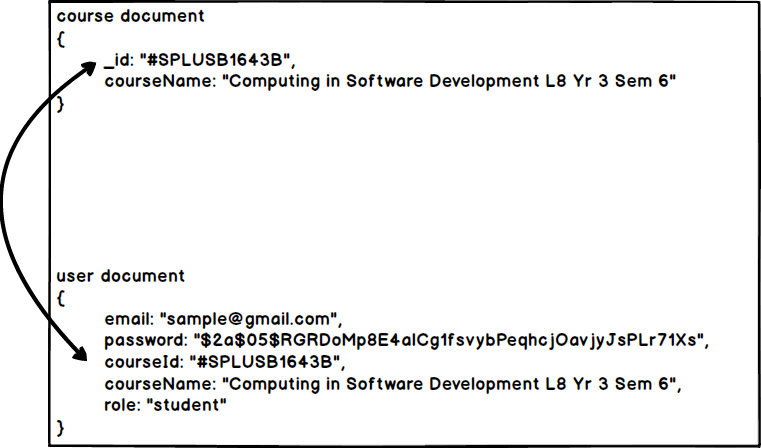
Following the Agile design methodology we held weekly meetings and sprints where we would work on different features in an iterative development cycle. We also had a project leader who would assign task to different members. We also used the project tool on GitHub to manage tasks to be done, when we would finish a task we would mark it done on the project section.

# 7. Features of the Implementation

## 7.1 Database design

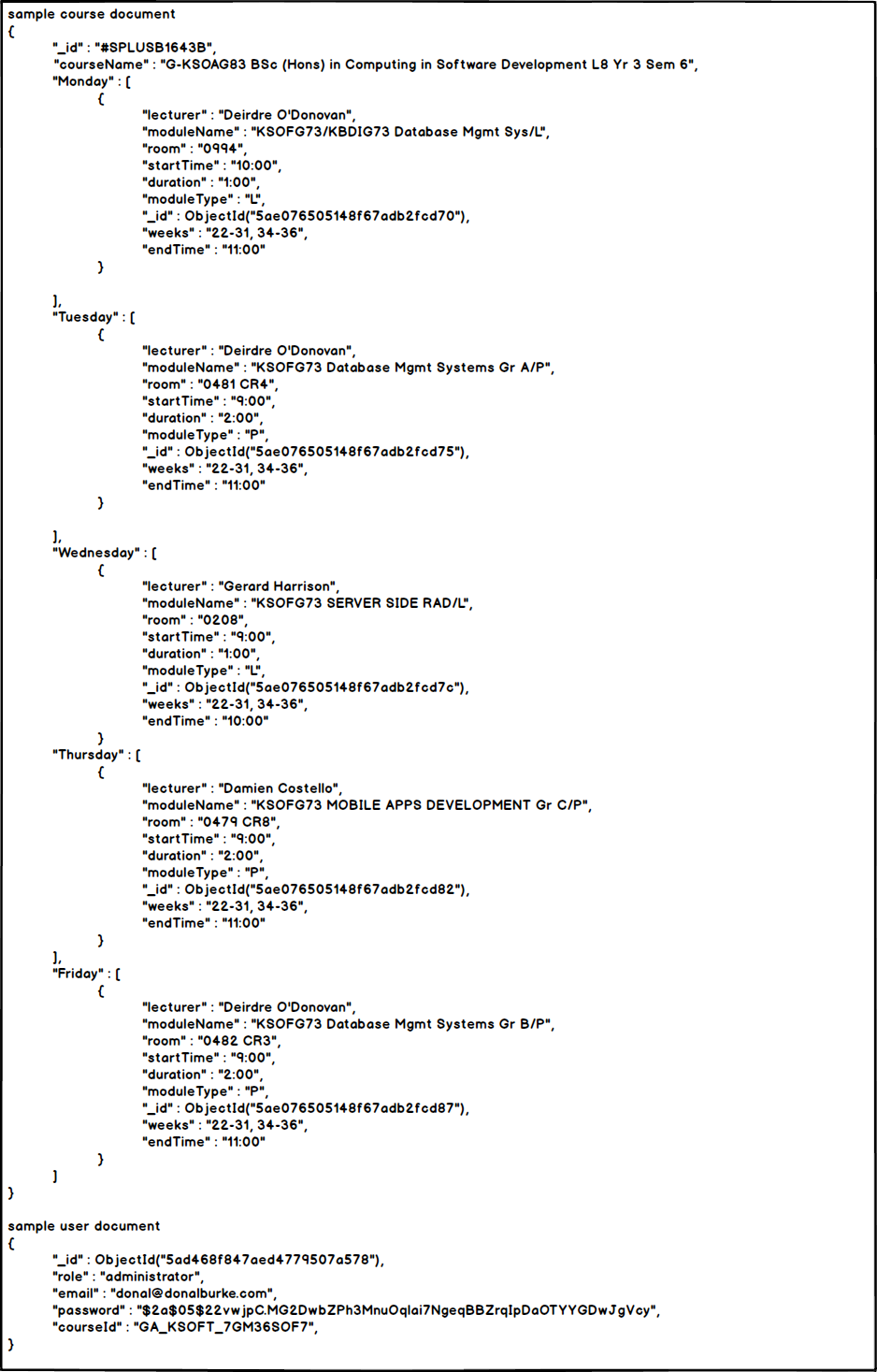
We designed the database in MongoDB because of its NoSQL format and document based structure in relation to MySQL tables and rows data model.

I have outlined what the schema looks like for the Course and User collections as shown below:

****

The main reason we used MongoDB is because of its easy integration with web applications such as Node and other web frameworks like Angular and Ionic, this is because Mongo’s document structure follows a JSON format. Since Mongo’s document are in JSON format this makes it very easy to build web applications using Mongo as a database solution. Since all modern web frameworks are built on JavaScript this means that its extremely easy to use Mongo in Web based applications.

Sample Documents from Database



## 7.2 Server design

## 7.2.1 Node server

The Node server was built on the Epxress.js framework, a framework that allows you to easily create various routes and endpoints on a node application, the Express framework builds upon Nodes native HTTP library and takes out a lot of the bulk that would normally be involved with designing a REST API.

We used Mongoose to define MongoDB Models and Schema Types and used Mongoose’s built in querying structure to make complex query’s to the database.

Passports.js was used as a authentication middleware for our application, we used it in combination with the JWTPassport.js library to allow us to authenticate users logging into the server. Another library that we used was the bcyrpt.js library, we used this library for hashing passwords for users before the document is stored in the Mongo database.

Server API Endpoints

Base URL: https://donalburke.me

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP method | URL path | Description | Permission Group |
| GET | /api/courses/ | Retrieves a list of all courses from the database | Student, Lecturer, Mod, Admin |
| POST | /api/courses/ | Retrieves a specific timetable from the database | Student, Lecturer, Mod, Admin |
| POST | /api/courses/updateModule | Updates a module in a timetable by id | Lecturer, Mod, Admin |
| POST | /api/courses/createModule | Creates a module for a timetable | Lecturer, Mod, Admin |
| POST | /api/courses/deleteModule | Deletes a module in a timetable | Lecturer, Mod, Admin |
| POST | /api/auth/register | Register a user in the Users table | Student, Lecturer, Mod, Admin |
| POST | /api/auth/login | Logs a user in and sets the users id to a JWT | Student, Lecturer, Mod, Admin |
| POST | /api/auth/updateUser | Updates a user in the database with specified fields | Student, Lecturer, Mod, Admin |
| GET | /api/auth/protected | Checks if the users has a valid session | Student, Lecturer, Mod, Admin |

## 7.2.2 Python web scraper

When designing the python script, we went to the timetable.gmit.ie website and looked at how the website pulls the information from the database of timetables. The website uses a ASP.NET form for sending requests to a server and loads a new page with the information retrieved from the server.

When writing the script, we ran into a problem, the timetable website uses sessions to validate that you are a user, to get around this problem we had to get our web scraper to visit the first page [http://timetable.gmit.ie](http://timetable.gmit.ie/) and go to the next subsequent pages by submitting ASP.NET forms with the proper VIEW\_STATE's and EVENT\_STATE id's.

After getting to the programmes page on the timetable website you will come across a form where you can choose from a dropdown menu what course you want to get the timetable for. We designed our web scraper to loop over each item in the dropdown and pass the course ID into the ASP.NET form and retrieve the timetable for each course.

When the scraper visits a timetable page it loops over the DOM and selects the appropriate information and populates it into a Mongo database.

## 7.2.3 Server management & security

We deployed the server onto Digital Ocean, main reason being that Digital Ocean’s server costs are quite cheap and if you are registered as a student on GitHub you get 50 dollars of free credit for Digital Ocean. We setup the server with a domain name, for easier access when using SSH rather than typing out the IP address every time we thought it would be easier to setup a domain name instead.

The Digital Ocean server was setup to use SSL. We used letsencrypt and certbot to configure the domain to be used with SSL. The reason we chose to setup the server over SSL was mainly because you cannot use JWT without an SSL certificate.

We used NGINX as a proxy server, any traffic that makes a request to the server on any port other than 443 gets redirected to 443(HTTPS port) and gets proxied to the node app running on localhost:8080, having this setup allows the server to be extremely secure and makes sure that all traffic coming into the server is on HTTPS.

##### 7.2.4 NGINX configuration file

server {

    # Enable HTTP/2

    listen 443 ssl http2;

    listen [::]:443 ssl http2;

    server\_name donalburke.me;

    # Use the Let’s Encrypt certificates

    ssl\_certificate /etc/letsencrypt/live/donalburke.me/fullchain.pem;

    ssl\_certificate\_key /etc/letsencrypt/live/donalburke.me/privkey.pem;

    # Include the SSL configuration from cipherli.st

    include snippets/ssl-params.conf;

    more\_set\_headers 'Access-Control-Allow-Origin: $http\_origin';

    more\_set\_headers 'Access-Control-Allow-Credentials: true';

    location / {

        if ($request\_method = 'OPTIONS') {

            more\_set\_headers 'Access-Control-Allow-Origin: $http\_origin';

            more\_set\_headers 'Access-Control-Allow-Methods: GET, POST, OPTIONS’;

            more\_set\_headers 'Content-Length: 0';

            return 204;

        }

}

##### 7.2.5 Server folder structure

Outlined below is the tree structure of what the server folder looks like.

.

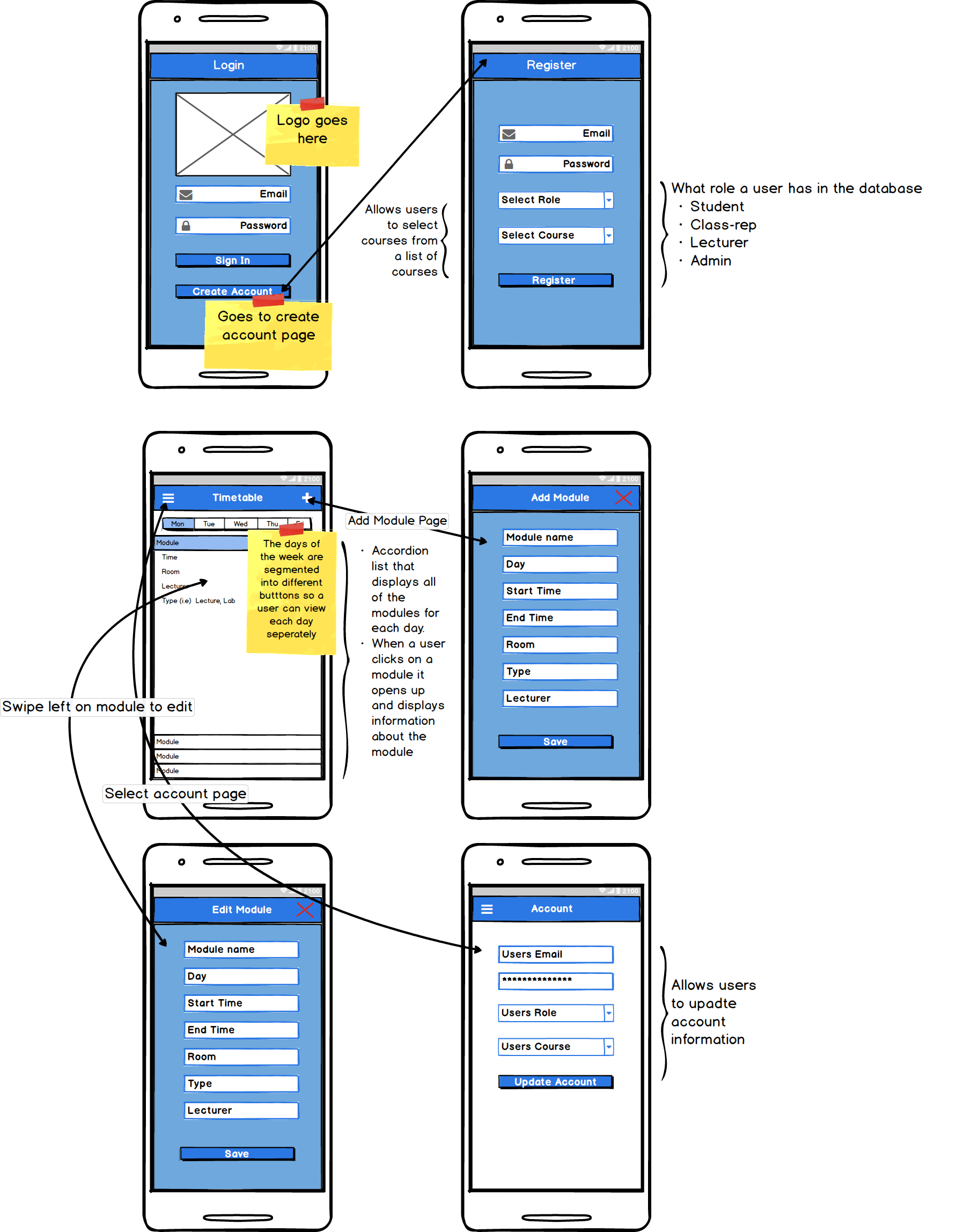
├── app

│   ├── controllers

│   └── models

└── config

## 7.3 App design

7.3.1 Mock up 

When designing the application we used the mock-up shown above as a guideline, the app was designed using the Ionic framework as mentioned above. The reason we chose Ionic was because of previous experience developing apps with the platform and Ionics cross-platform design. This allowed us to develop our app using one framework and also allowed us to deploy it on many different devices.

Login page

When designing this page we tried to follow the format that was specified in the Mock-up. We wanted a page that when displayed to the user was aesthetic in appearance and easy to use with regards to UI. From this page the user can navigate to the Home Page after logging in or to the Register Page.

Register page

This page was also designed in the same aesthetic theme that we chose for the Login page. The Register page has a few input boxes where the user can enter their personal information, we added a Select-Search component that we made in Ionic to the page where once clicked the user is navigated to a page with a dropdown list of all of the course in the college, they have the option to either scroll through the list or use the search bar at the top of their page to search for their course.

Home page

The Home page of our application is where the user can view their timetable. We decided to Segment the page into different days. We used the Segment component in Ionic to achieve this, when a user clicks on a segment which are marked by each day of the week Monday – Friday, they will be able to view each module for each day. The modules are then ordered by start time in descending order. If the user swipes left on a module they have the ability to either edit or delete a module.

Edit Module page

When the user clicks on the edit button for a module they are redirected to this page, only certain users with the correct permission group are allowed to edit a module. The user has the ability to change the fields pertaining to the module and then they are updated to the database.

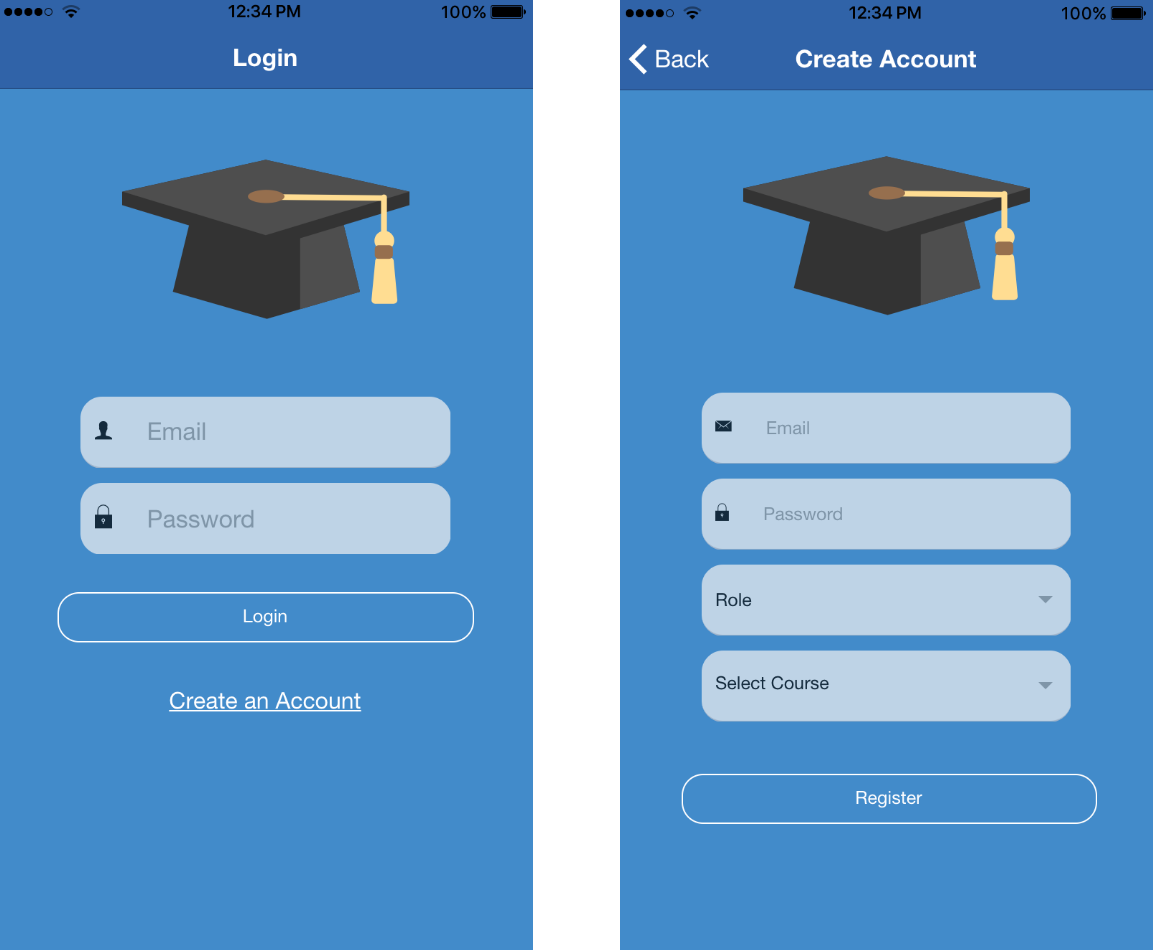
Add Module page

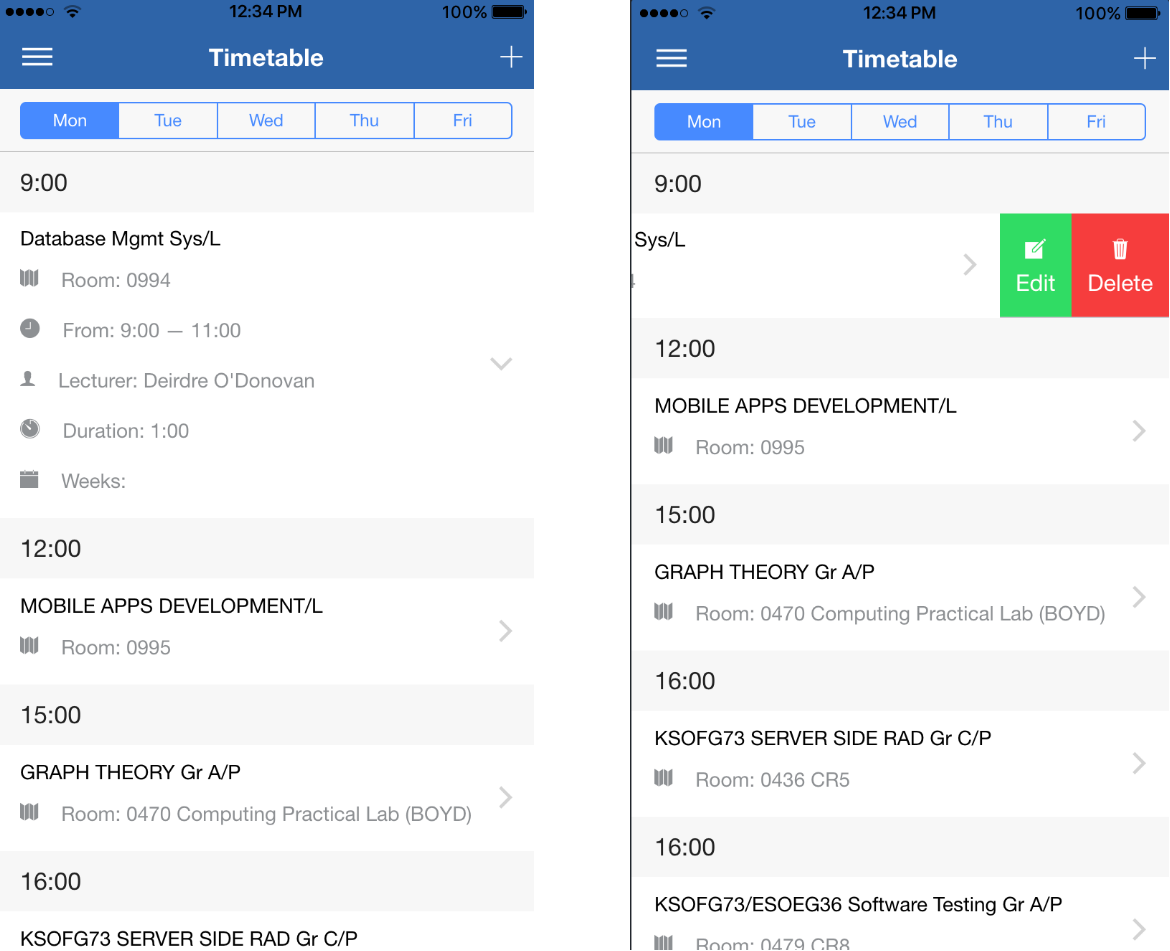
When the user clicks the add button on the navigation bar they are redirected to the Add module page, here they can add a new module some of the fields are Name, Type, Day etc.

Account page

When the user navigates to the account page from the side menu they have the ability to view their account information and change certain account details such as; email and their course after updating their information it is saved in the database.

App screenshots





# ../Downloads/Untitled%20Diagram%20(3).png

# 8. Limitations

##### 8.1 Time

One of the limitations that we encountered in the project was time, we found that meeting up regularly was quite difficult this was became more of an issue when one of our team members left the county and was not able to meet up in person. We had to then change the way we did our meeting to group calls and video calls on platforms such as skype and messenger.

##### 8.2 Cost

Another limitation we encountered was cost, we found that when we wanted to publish the application to the iOS and Android app stores that you have to pay a developer accounts fee, because of this limitation we were never actually able to upload the apps to the store.

# 9. Known Bugs

##### 9.1 Content resize bug

When designing the Home page of the application we encountered a bug with Ionic where a portion of the content on the page was being hidden under the page header, we found a partial solution this bug by resizing the content when a user clicks on one of the segment buttons.

# 10. Recommendations for Future Development

##### 10.1 Push Notifications

One of the futures that we would have like to have implemented was push notifications, we found that when trying to implement them into Ionic was that we would have to develop push notifications separately for each platform. We also realised that we would need developer accounts on each platform to do so, again we ran into the issue of cost that was mentioned above.

##### 10.2 Room search

Another feature that we felt was out of the scope of this project with regards to time was a room search feature. This feature would allow users to search for any room in the college and check its availability for any given day and hour.

##### 10.3 Library room booking

A feature that we would have also like to have implemented was a library room booking feature. Currently there is a feature on the library.gmit.ie website that allows you to do so but we would have like to have implemented the feature into our application as well.

# 11. Conclusions

##### 11.1 Learning outcomes

Version control

One of the things that we learnt as a team was how to properly use a version control system for designing a software application, the version control system we used was git. Git allowed us to collaborate on the project as a team and it was also a central location where are app was stored. Git has a commit system were commits to the repository are saved in the commit history section this allows you to go over changes that you have made and lets other collaborators edit and make changes to commits.

Project management

Another learning outcome that we derived from this project was project management. For managing the project we used a Facebook messenger chat group to ask each other questions and to go over tasks to be done. We would pin weekly meetings in the chat group that would alert us on the day as to what time and place the meeting would be held. This allowed us to meeting on time and efficiently.

Python

When designing the web scraper we had no previous knowledge of python, as we have never used it In college and none of the team members has ever used it for personal use either. We found that python was a great tool with regards to web scraping and we also found that it is a very robust programming language with a lot of interesting features that you wouldn’t normally see in most modern programming languages.

Node

We found that when designing the server that Node was a great tool with regards to setting up a robust REST API, with the use of libraries such as Express, Mongoose and Passport we were able to setup a server that has role authentication, session based logins and much more.

Server management

We used tools such as PM2 for server management, PM2 is a node library that lets you run processes in the background. It is essentially a process management tool with features such as log, which lets you log all of the terminal output into files. We also used NGINX as a proxy tool to proxy all traffic coming in on the server to the SSL port (443) on the server.

Another outcome from setting up the server was being able to secure it with SSL for encrypted traffic coming in and out of the server. Which we feel will be a useful thing to know for the future.