# Emerging Technologies
## *Library Web Application*


Gary McHugh - G00308668
Alan Murphy - G00312295
Jason McTigue - G00312233
Darren Fitzpatrick - G00311853

# Table of contents

# Project Documentation

## 1 Introduction

The aim for this project was to develop a single-page web application predominantly written in the Go programming language. As a group, we decided to create a Library web application based off the current GMIT library website. Our goal was to create a professional looking, informative and user friendly website designed for students. We done this using various technologies which we researched at the beginning of this project.

The Library web application provides a portal for students to access various items such as Exam Papers, EJournals, Library Opening Hours and information about the Study Rooms. The Library website was also supposed to allow students to view the books available by category, unfortunately we were unable to return this data to the website (further discussed in the issues section).

This web application uses a Go based Rest API that interacts with a local MySQL database. A user can retrieve all the books available in the library or a single book from the database. The Rest API also has Insert, Update and Delete functionality fully coded, however this functionality was not implemented due to time restrictions.The project also contains a jQuery file which we could not get to work fully with this project, this file would have been used to create a single web page application had it functioned correctly.

**Aims:**

- Create a professional looking and user friendly web application
- Provide relevant information to Students of GMIT
- Implement a Database into the web application using a REST API
- Use jQuery to create a single web page application

**Objectives:**

- Learn how to create a single web application
- Become more familiar with the Go programming language
- Learn how to create a Rest API
- Develop our teamwork skills
- Become more familiar with MySQL

# 2 Architecture

We used a 3-Tier architecture to create this Library web application, each layer of the architecture is discussed below:

**Tier 1:**
The first tier of the application contains the user interface, this consists of a website created using HTML, CSS and JavaScript. We used Bootstrap to quickly develop professional looking interfaces for the website.

**Tier 2:**
The second tier of this web application contains the REST API developed using go, which is used to interact with the database. It also contains the Go code that is used for navigation through the web application. The Macaron framework was used to make the routing in the application simpler.

**Tier 3:**
The final tier of the application contains the MySQL database which we created locally on our machines. The Library.Sql file used to create the database is available on the GitHub repository for this project. The MySQL database contains a table full of the books that are available in the Library. This data is then returned to the user using the REST API from tier 2.

Below is a graphic representation of the 3-tier client server application:

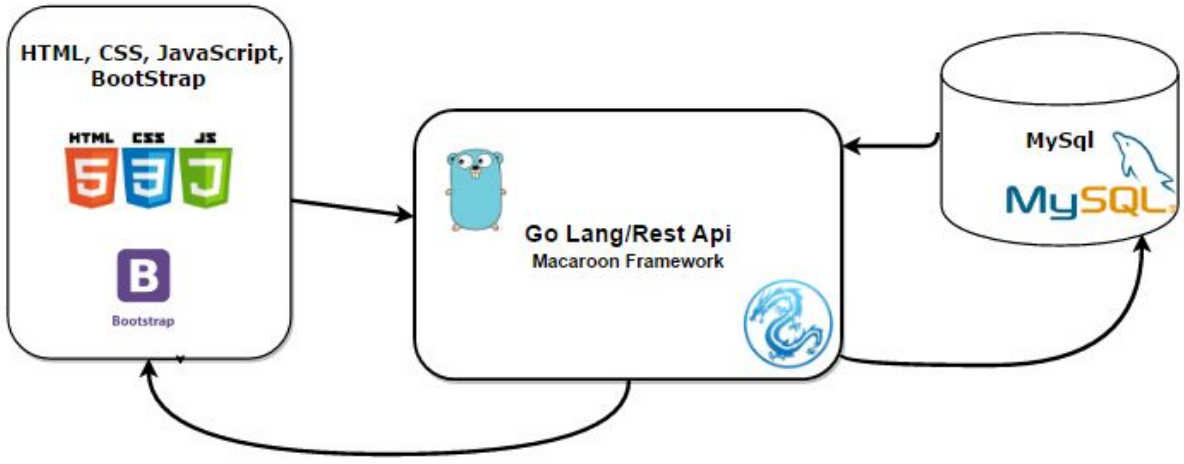


**Fig1.**

# 3 Technologies Used

**Technologies:**

- HTML , CSS, JavaScript
- Bootstrap
- jQuery
- Golang
- Macaroon Framework
- MySQL

We used HTML5, CSS (cascading style sheet) and JavaScript to develop our website as we had previous experience with this technology and it is the most common way to create a website. HTML was used to create the various controls on the website such as buttons, textboxes and Images. We then used Css to format these controls, this improved the look and feel of our web application and improved the user experience. We used JavaScript to give our website functionality and to make the page's load faster.

We used Bootstrap to help us build a better-looking user interface, Bootstrap is a HTML, CSS and JavaScript framework that is used for developing nicer looking and more responsive websites. We used Bootstrap templates that were available online to get our website up and running quickly. This saved us a lot of time as we didn't have to create the website from scratch. It also provided us with a professional looking and responsive website.

We wanted to use jQuery in our application to give the user seamless transitions between the web pages. This would make it seem to the user that although they are looking at a new page, no navigation had occurred. This would turn our website into a single web page application, unfortunately we were unable to get the jQuery fully implemented into the website as we were unable to navigate the pages using jQuery (further discussed in Issues)

As per the project instructions we used the Go programming language to create the web application. We used Go to interact with the Macaron framework for navigation (discussed in the next paragraph), we also used Go to create a REST API to interact with the MySQL database. The REST API has Select, Insert, Delete and Update functionality, this was all coded using Go.

As mentioned above, we chose to use the Macaron Framework in our web application. Macaron is a high productive and modular framework developed for Go. It is mainly used for routing and templating files. We used Macaron routing in our project to form the web pages when the user navigated to them. This meant that we only had to have one definition of the header, navigation bar and footer html sections.

We chose to use MySQL as the database in our project as the relational database schema it uses would work well with our project. We also had previous experience using MySQL from previous modules in college. We used the MySQL database to store the relevant details of books that are in the Library. This data would then be displayed back to the user using the REST API according to which category they selected.

# 4 Methodology

To develop our web app we all met as a group and brainstormed various ideas that we had. We used the github issues feature as a discussion form to add ideas and suggestions. One group member suggested we create a web application based on a library website as we would be able to implement various features.

Once we had finalised on an idea the next step was to decide which technologies we would use. To achieve this we compared a number of technologies in each tier of our application and discussed which ones we thought would be best to implement into our web app. The next step was to add the HTML templates to the application using bootstrap and then we added the Macaron framework so we could route through each page.

Once the basic template of the project was complete, one group member uploaded the file to github and made the rest of the group members collaborators. This enabled each one of us to be able to individually clone down the project and add sections to the master file.

To evenly distribute the workload we delegated the project into different tasks. As there were four of us we each took a section of the website to develop and we would meet regularly to update each other on our progress. We again took advantage of the github issues during this time to keep track of any problems we had or any bugs we encountered. We then tried to add a no page refresh aspect to our page so it would no longer be just a static web page. This is where we encountered our first problem as we

were unable to get the jQuery to work properly because of a problem with the templating. We knew this because we were able to get it working when testing on an external page.

Another group member was also trying to implement the database into the project using XAAMP, however this to proved to be quite problematic also. As we were now unable to get either the jQuery or the database functionality to work. We then contacted our lecturer for help on the issues we were having. Our lecturer suggested that we look into using MySQL locally and then using a Rest Api that would use GO to interact with the MySQL database as it would be an easier way to manage the database.

We did exactly what our lecture had advised, however, we were still unable to display the data in a useable format as it was being returned in JSON. But we were able to have a fully working Rest API with Select, Insert, Update and Delete.

For the final part of the development of the project we tried to merge together the Rest API go file and the macaron go file only to realise that the each use the run function in different ways which meant we were unable to merge them together successfully.

To finish up the project each member fixed any underlying errors that they had with their sections such as displaying images and formatting. We then created a user guide on the github repository detailing the requirements required to run the application and documentation describing the functionality of the web app.

# 5 Functionality

As mentioned in the methodology section of this paper, our web application is separated into different pages. We will discuss the functionality of each of these pages in this section.

Our web application contains various pages which offers various pieces of useful information to the Students. The web application is designed to be user friendly and professional looking; we believe that the user experience is key to a successful web application. The home page contains a sliding image feature which contains links to items like Timetables and inter library loan information. The home page also lets students access Exam Papers and EJournals. The home page contains all of the links to the items that a student would frequently look at.

The Book a room page of our web application contains information for the students about the conditions of using a study room, as well as how to book a room. The book a room page also contains a table that has information about the study rooms, this information includes the room number, the room size and what the room is equipped with. Initially we had planned on implementing a database into this page so that a user could book a room via the website. However, due to time restrictions we were unable to implement this piece of functionality.

The About Us page contains general information about GMIT and the GMIT mission statement. It also contains information about the head librarians in the Galway campus and how a student could contact them with any issues they may have. Finally it contains a map of the location of the Galway campus. The Google maps API was used to display this map.

The Opening Hours page contains four tables which contain the opening hours of the libraries for each of the GMIT campus. This provides the students with an easy way of checking what times their library is open at and when it closes. We had initially intended on putting the opening hours into the MySQL database so that they could be easily updates when the Libraries opening hours changed during exam time. However, due to time restrictions we were unable to implement this functionality.

The Contact Us page contains details of how a student can contact each GMIT campus as well as contact information for various student union members. The Contact Us page also contains links to the various social media platforms where students can keep up to date with upcoming events and important information.

The web application also contains a sign in pop up box which can be accessed from any page. We had intended on implementing login functionality into the application using the MySQL database. However, due to time restrictions we were unable to complete this.

The final section of our web application is the Books section. The books section is split up into various book categories. This would allow the student to easily find the relevant books that they are looking for. This section would provide an easy way for a student to see how many copies of a certain book was left and how long they could loan that book for. The books were stored in a MySQL database and were retrieved by the Rest API. Unfortunately, we were unable to get the data into the table as we could not parse the JSON into a usable format. We then created tables to show what we had intended on doing in these pages. The Add Books part of this section contains a form that allows a

member of staff to add a new book to the database, a php script is used to do this. As the database is not hosted on a server we could not execute the script to implement this functionality.

# 6 Issues

We encountered many issues throughout the development of this project while using new technologies such as Go and The Macaron Framework. We documented all the issues and problems we encountered using the Github Issue's feature in our Github repository. We will discuss the main issues we ran into in this project in this section.

One of the biggest problems we had was using templates in our project, we did a lot of research online about templating in the macaron framework. However, we struggled to understand the appropriate way to template the project. We eventually templated the html into sections and used macaron to use various templates according to the page the user was on. We believe that this was a misjudgement on our part as it further complicated things down the line such as jQuery and using the Rest API.

We were also unfamiliar with the go programming language and due to it being a relatively new language we sometimes struggled to find information online about specific issues we were having, such as the templating and implementing the Rest API. This meant that we had to use our problem solving skills to come up with an effective solution to the problems encountered.

An issue which we experienced while making this project was implementing jQuery into our website. Having templated the html up we found it difficult to navigate through the page using jQuery. We were able to detect a click on the navigation bar and give a javascript alert to indicate the click but were unable to navigate from that click. We used similar code to navigate from a much more basic website as a test. We were then able to identify our previous error in templating the website as the cause of the jQuery not working. We then contacted our lecturer and spoke to him during our scheduled lab times about the issue. Our lecturer advised us to leave the jQuery out as the deadline was approaching and we still had other functionality to implement. This meant that it was too late to change the project when we realised the error we made while templating the website.

Another issue that we encountered during the development of this project was with the MySQL database. We attempted to set up the database using XAAMP as we had

experience with this technology from previous classes. We then realised that we were unable to get the application to communicate with the database using this technology as external files are not allowed to communicate with this technology. We then contacted our lecturer and informed him of the issue. He advised that we run MySQL locally on our machines and develop a Rest API that would use GO to interact with the MySQL database. This proved to be a more straight forward way of using a database in the application.

Having implemented the MySQL database and a Rest API to interact with it, we encountered another problem. The data returned from the Rest API was in a JSON format. We were unable to parse this data into a usable format. We had intended on displaying the data in a html table from the database according to category (as shown in the books section of the webpage). We attempted to parse the JSON into a html table using a for loop but were unable to get the data into the format we wanted. This meant that the data we got back from the database is displayed as a block of JSON. We were able to have a fully working Rest API with Select, Insert, Update and Delete but were unable to get the data in a usable format.

The final issue that we encountered during the development of this project was with the Run function in GO. We discovered late on in the development of the project that the Rest API go file and the macaron/navigation go file use the Run function differently. We tried to merge the go files together to get the application to run from a single go file, this proved unsuccessful. This was due to the different ways that Run is used in the application (m.run(4001) <- for navigation) and (router.run(“:4001”) <- for Rest API). This conflict means that data cannot be gotten from the database while on the website. The files must be run separately.

# 7 Conclusion

In conclusion we were extremely disappointed in how the project turned out. We realise that we spent far too long on the design aspect of the web app. The main reason for this was down to the fact that we were already quite familiar with these technologies and perhaps we thought that we should demonstrate our skills in these areas. However the aim of the project was to learn an emerging technology, in our case the Go programming language. When we began to start implementing more Go into the project this is where we began having problems but at this stage it was already too late fix.

We feel we didn't meet the goals we had set out at the beginning of the project. We unfortunately didn't get to see our web applications full potential. There was a lot of features that would have really improved our project, such as, getting data returned from the Rest API in a usable format, or being able to navigate through the pages using jQuery. We were very close to having these features working, and lacking them meant our web applications functionality felt limited, this is the main reason we are disappointed with our project.

As a whole we feel the design aspect of the website looked very professional. Our intention was to design a user friendly, aesthetically pleasing web page. We feel we achieved a layout that was well structured and allowed for easy navigation. We added Bootstrap templates, which incorporated a simple design and user friendly feel throughout the web application. In the end we were pleased with the front end design of our project, despite this however, we feel that the functionality was lacking.

If we were to do the project all over again we would have made a lot of changes. We would have put less emphasis on the design aspect, managed our time more efficiently, and adhered to a more effective plan. We also would have approached the templating quite differently. We wouldn’t have used the templating in the way we did we as we were unable to use jQuery because of this. We also would have focused on the REST API more to get data going to and from the website.

Although we did not meet the goals we set out at the beginning of the project and were disappointed with the lack of features in the project, we can confidently say that this has been a great learning experience for us as a group. We now know that we should try and do the most difficult parts of a project first and the parts that we are unfamiliar with. We feel as though this learning experience will stand to us in future projects throughout college and in future with our professional work.