
HaccpApz - Cloud based platform including Mobile and Web Applications

Martin Coleman

B.Sc.(Hons) in Software Development

APRIL 17, 2017

Final Year Project

Advised by: Mr Martin Hynes

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)



Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 5 |
| 1.1 | Client | 5 |
| 1.2 | Client's Problem | 5 |
| 1.3 | Client's Idea | 5 |
| 1.4 | Project Aim | 6 |
| 2 | Context | 7 |
| 2.1 | Objectives | 7 |
| 2.2 | Deliverables | 7 |
| 2.3 | Source Code | 8 |
| 2.4 | Scope | 8 |
| 2.5 | Relation to Degree | 8 |
| 2.6 | Minimum requirements | 9 |
| 2.7 | Enhancements | 10 |
| 2.8 | Risk | 10 |
| 2.9 | Security | 11 |
| 2.10 | Scheduling | 11 |
| 2.11 | Document Overview | 12 |
| 3 | Methodologies | 13 |
| 3.1 | Research methodology | 13 |
| 3.2 | Software Development methodology | 13 |
| 3.3 | Testing | 15 |
| 3.4 | Meetings | 16 |
| 3.5 | Project Management | 16 |
| 3.6 | Development tools | 16 |
| 4 | Technology Review | 18 |
| 4.1 | Overview | 18 |
| 4.2 | Development | 18 |
| 4.3 | MongoDb | 19 |

| | | |
|----------|--|-----------|
| 4.4 | Express | 20 |
| 4.5 | Angular | 21 |
| 4.5.1 | Model–View–Controller | 21 |
| 4.6 | Angular 2 | 23 |
| 4.6.1 | TypeScript | 23 |
| 4.7 | Node | 24 |
| 4.8 | Ionic | 24 |
| 4.8.1 | Apache Cordova | 25 |
| 4.9 | JSON (JavaScript Object Notation) | 25 |
| 4.10 | HTTP | 26 |
| 4.11 | REST (Representational State Transfer) | 27 |
| 5 | System Design | 28 |
| 5.1 | Overview | 28 |
| 5.2 | Mobile Application | 29 |
| 5.3 | Cloud Server | 30 |
| 5.3.1 | API Server | 30 |
| 5.3.2 | Database | 30 |
| 5.3.3 | Web Application Server | 31 |
| 5.4 | Web Application | 31 |
| 6 | System Evaluation | 36 |
| 6.1 | Overview | 36 |
| 6.2 | Unit Testing | 36 |
| 6.3 | Integration Testing | 37 |
| 6.4 | Functional Testing | 37 |
| 6.4.1 | Mobile Application | 37 |
| 6.4.2 | Web Application | 38 |
| 6.5 | Usability Testing | 38 |
| 6.6 | Acceptance Testing | 39 |
| 7 | Conclusion | 40 |
| 8 | Appendices | 42 |
| 8.1 | Source Code | 42 |
| 8.1.1 | Web Servers | 42 |
| 8.1.2 | Mobile Application | 42 |
| 8.1.3 | Instalation | 42 |

About this project

Abstract Brendan Keane is an experienced chef working in Galway city for many years. He identified a problem that he and many other chefs, cooks and food management staff faced daily and decided he was going to improve it. He thought of a creating a better digital based system to improve upon the current paper based system. Being former student of Galway Mayo Institute of Technology(GMIT) he returned to the computing department with his idea. The current system used by all premises that handle and prepare food is time consuming and the staff are constantly minding paperwork. These premises must legally fill these forms creating problems between management and staff if not completed. The client's idea was to reduce the amount of time wasted filling paper work. He hopes that food handling premises can adopt and use his system to manage their forms and improve their productivity. The client observed that most staff carried their phones around with them and an App could be used to complete the forms much faster and easier. The client also wanted to be able to analyse the data.

Authors The author of this project is Martin Coleman. Martin is a graduate of the BSc in Computing in Software Development in 2016. Currently he is pursuing a BSc(honours) in Computing in Software Development. Martin has an interest in computing, programming and various other technologies. Martin is always interested in improving his programming skills and knowledge.

Chapter 1

Introduction

This chapter introduces the project and the client. It discusses the origins of the project idea and the aims of the project.

1.1 Client

Brendan Keane is an experienced chef working in Galway city for many years. He identified a problem that he and many other chefs, cooks and food management staff faced daily and decided he was going to improve it.

He thought of a creating a better digital based system to improve upon the current paper based system. Being former student of Galway Mayo Institute of Technology(GMIT) he returned to the computing department with his idea.

1.2 Client's Problem

The current system used by all premises that handle and prepare food is time consuming and the staff are constantly minding paperwork. These premises must legally fill these forms creating problems between management and staff if not completed.

1.3 Client's Idea

The client's idea was to reduce the amount of time wasted filling paper work. He hopes that food handling premises can adopt and use his system to manage their forms and improve their productivity. The client observed that most staff carried their phones around with them and an App could be used

to complete the forms much faster and easier. The client also wanted to be able to analyse the data.

1.4 Project Aim

To create both Web and Mobile based Applications that connect to a comprehensive cloud based system that manages data for its users.

With knowledge of the client and the problem, Chapter two will delve into the project itself. It discusses different elements of the project in detail.

Chapter 2

Context

This chapter discusses the project. It covers areas such as deliverables that were by set by the client at the start to other areas such as risk, scheduling and scope that had to accessed after brief was received.

2.1 Objectives

The objectives were decided upon at the start of the project. They were decided upon after the initial meeting.

The objectives were as follows:

- To come to a mutual understanding of the project and understand the requirements of the project.
- To research techniques and technologies to find the best solution to the project.
- To design a system architecture that is the most suitable for the project.
- To create a system and software capable of fulfilling all the requirements.

2.2 Deliverables

The deliverables were decided upon at the start of the project. They were derived from what the client wanted.

The deliverables were as follows:

- A cross platform mobile application for Android and Apple iOS.
- A web application that allows users to view their saved forms.
- A cloud service that stores the data and runs the servers.

2.3 Source Code

All the source code used in this project is available on GitHub. At the first link below you will find the source code for the mobile application. At the second link, you will find the source code for the web server and the API server.

Both repositories contain their own README.md file that has information about the project, how to setup the required environment and how to run both servers and build the mobile application.

Links to GitHub Repositories:

<https://github.com/Martinc94/HACCPManagementApp>

<https://github.com/Martinc94/HACCPAPZ-Server>

2.4 Scope

This project is about to designing and implementing a whole system to solve satisfy the client. It contains a mobile application, a web application, an API server, a database and a cloud server to connect and host the services for all the above.

The scope of the project is a similar level to a project that a customer might require in industry. Its requirements were achievable within the given time. The project contains many elements such as mobile, web, databases, user authentication, managing a cloud server, data analysis, data handling, transfer and manipulation. These elements combined cover a wide range of areas that are relevant to the course. The project has great room for expansion and many possibilities to implement.

2.5 Relation to Degree

The project related to the degree on many levels as it contains a wide spectrum of different technologies, problems and tasks that are covered on the course and are commonplace in industry. The project contains mobile and

web front-ends, a back-end for handling data, an API for accessing the back-end, a database and a cloud server to handle all the parts mentioned above.

Some of the course modules that related to the project:

- Emerging Technologies.
- Data Representation and Querying.
- Mobile programming.
- Database management systems.
- Software Testing.
- Internet application development.
- Graphical User Interface and Web development.

2.6 Minimum requirements

The minimum requirements for the project were set as a worst-case situation for the project. The client was assured that he would receive a system that would satisfy these requirements. Even if a major problem slowed down development of the system it was felt that these requirements could be achieved.

The minimum requirements for the projects were:

- A cross platform mobile application for Android and Apple iOS.
- A cloud service for storing data from mobile application.
- A website to display stored data.
- A database to store the data.
- An API server to connect above services.

2.7 Enhancements

The possible enhancements to the project included some features the client wanted to be implemented in the system and some enhancements thought of with the project supervisor to improve the system. These enhancements were to be worked on after the minimum requirements were achieved.

The possible enhancements to the project were:

- Analysis of data for administrator use.
- Graphical representation of data analysis.
- Incorporation of Google maps for location services.
- Take photos on mobile application and store on server.
- Display photos alongside relevant forms.

2.8 Risk

The risks involved with the project were all the most common problems that could go wrong in the software development process. They were analysed and decided upon at the start of development so they could be monitored and avoided.

The risk involved in the project included:

- Miscommunication of requirements.
- Incorrect or incompatible technologies being chosen.
- Using new technologies.
- System architecture and design.
- Performance of various systems.
- Organisation and work management.

2.9 Security

The project requires security on each component of the system. Sensitive data such as passwords being stored in the database must be protected. Access rights to user's data and to different parts of the web application must also be protected. The API server must authorize who can access its HTTP methods.

2.10 Scheduling

The scheduling for the project had to be planned early to keep the development on schedule and make sure each deadline is met. Scheduling is an important element of software projects. Many failed projects can conclude that the main cause of failure was poor time management at early stages of development.

Early in the projects life cycle the scheduling will focus on requirements gathering.

- Requirements gathering.
- Research on project and technologies.

After the requirements are gathered, focus must change to the first milestone, the first prototype of the mobile application.

- Decision on technologies and start development.
- Develop prototype mobile application.
- Evaluate prototype and redesign if required.

When the mobile prototype is developed, the focus switches to the second milestone which is to get the server setup and communicating with the mobile application.

- Setup cloud server on azure.
- Setup database on azure.
- Start development of API server.
- Working draft of API server and mobile app working.

Once the mobile application and server are communicating, the focus will be to start development on the web application.

- Start development of web application and server.
- Prototype of web application.

When all the prototypes are working, the focus will change to creating a fully functioning version of them all.

- Working version of mobile application and API server.
- Working version of Web application.

As full functionality of the system is achieved data analysis can begin.

- Start Analysis of data.

When all the enhancements are implemented the system and relevant extras can be handed over to the client.

- Delivery of finished project including mobile packages.

2.11 Document Overview

This rest of this document contains an in depth look at the development process.

Chapter three focuses on the methodologies that were researched and implemented in to the project. It includes both research and software development methodologies. It also discusses area such as project management, meetings and development tools.

Chapter four reviews in detail the technologies and architectures that were chosen for this project. It provides an insight as to why certain choices of technologies and architectures were chosen to help produce a system that is modern, adaptable and scalable.

Chapter five looks at the system design in fine detail. It talks about how the different elements interact together and how what role each element performed to produce the final system.

Chapter six talks about how the system was evaluated. This includes how the system was tested. The various forms of testing that were looked at include Unit, Integration, Functional, Usability and Acceptance testing.

Chapter Seven concludes the whole project. It provides an overall review of how the project went and any improvements that would be implemented if project was to be redeveloped.

Chapter 3

Methodologies

This Chapter covers the different methodologies that were implemented in this project. It looks at the different types of research methodologies that were used such as Qualitative and Quantitative, different software development methodologies and why they were chosen. Other areas covered include testing, meetings, project management and development tools.

3.1 Research methodology

Firstly, the research methodology that was used in this project was Qualitative [1]. Qualitative research is useful for studies at an individual level. It was important to gain an insight from the client's perspective and to establish what requirements the client had.

The main way that research was collected was from interviews and meetings with the client. This information helped to decide what had to be completed, what technologies were best for the project and what software methodology would be most suitable to use in the project.

Later in the projects life cycle Quantitative [1] research was used. The Prototypes of the mobile and web applications were distributed by the client and he returned the feedback. This allowed us to gain valuable insight into what potential users thought about the applications. The feedback was used to make adjustments that improved the applications.

3.2 Software Development methodology

The software methodology that was used in this project was Extreme Programming (XP). Extreme Programming is a form of Agile software develop-

ment. It became clear early that Agile would be the most suitable methodology to use. Agile allows for Incremental development, changing requirements, prototyping, sustainable development and close cooperation between business and developers. As there would be weekly meetings with client and supervisors being able to show progress would be a bonus.

There are many types of Agile methodologies. The decision was narrowed down to two types, Extreme Programming and Scrum. Extreme Programming was chosen over Scrum as Extreme Programming for the following reasons:

- Extreme Programming works in shorter iterations than Scrum.
- Extreme Programming allows for continuous integration.
- Extreme Programming works in strict priority order which allows most important features to be worked on and allows priority to be changed easily.
- Extreme Programming recommends practices such as test driven development.

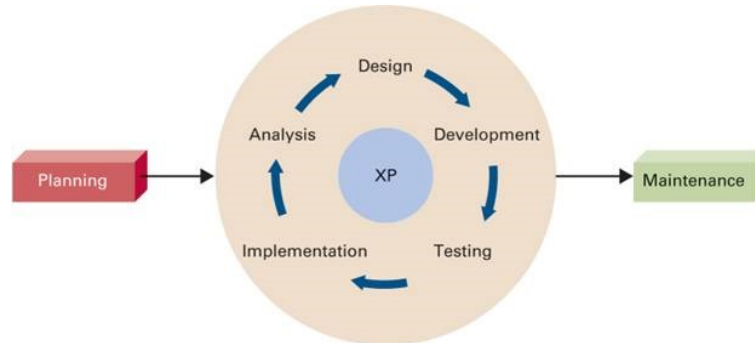
Extreme Programming focuses on adaptability rather than predictability which was important in a project like this. The values of Extreme Programming that were most relevant to the project:

- Communication between developer and client.
- Simplicity of development.
- Feedback from client/users.

Software development process as described by Extreme Programming:

- Planning.
- Analysis.
- Design.
- Development.
- Testing.
- Implementation.
- Maintenance.

XP Development Cycle [2]:

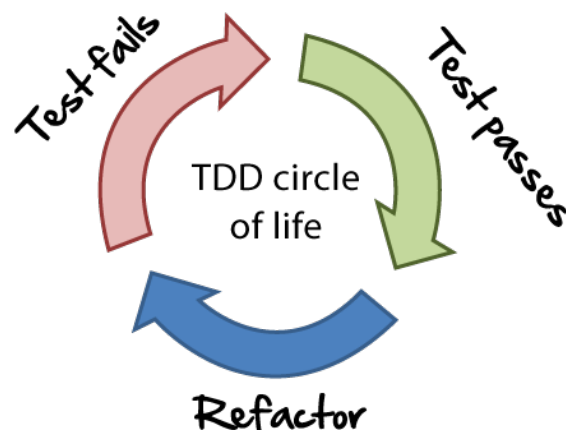


3.3 Testing

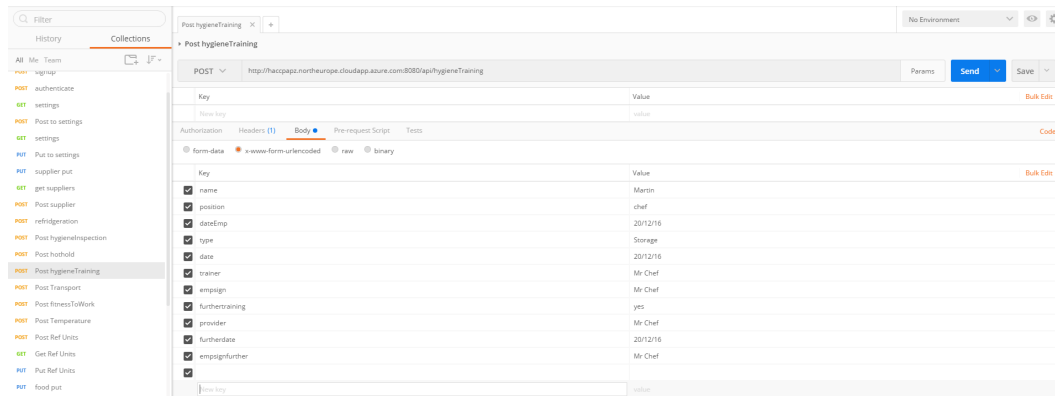
The type of development chosen allowed for Test driven development. Test driven development allow for continuous testing to happen alongside development. Tests can be written before or after new code is written. Tests can be run before and after new code is added to test to see if new code performs as expected and on old code to make sure new code hasn't effected old code.

This type of testing suited the project as there were many iterations of the project. Testing frequently made sure new and old code worked each iteration. This also prevents bugs being embedded deep into the code.

Test Driven Development Cycle [3]:



The main testing tool used during development was Postman [4]. Postman allows users to share, test, document and monitor APIs. Users can vary requests and see results in an easy to use and manage environment.



3.4 Meetings

Meeting were held weekly for the duration of the project. Meeting were with the project supervisors or the client.

Meetings consisted of:

- Progress Update.
- Feedback on progress
- Planning of next development iteration
- Q and A on various project elements if required

3.5 Project Management

GitHub was used all throughout the development of this project. GitHub allows for version control of the project, back up of code and recording of documentation. GitHub makes progress tracking easier.

3.6 Development tools

The main integrated development environment(IDE) used was Visual Studio Code. This Integrated development environment was used because its

support for typescript, HTML, JavaScript, JSON, integration of GitHub, IntelliSense and third-party extensions which provide extra support and many useful features.

Chapter 4

Technology Review

This chapter discusses in detail all the various technologies that were incorporated into this project. The chapter also covers why the technologies for each component of the system were chosen and the benefits they provide.

4.1 Overview

The following are some of the most prevalent technologies were used in the development of this project:

- MEAN Stack
- MongoDB
- Express
- Angular 1 and 2
- Node
- Ionic
- JSON
- HTTP and REST

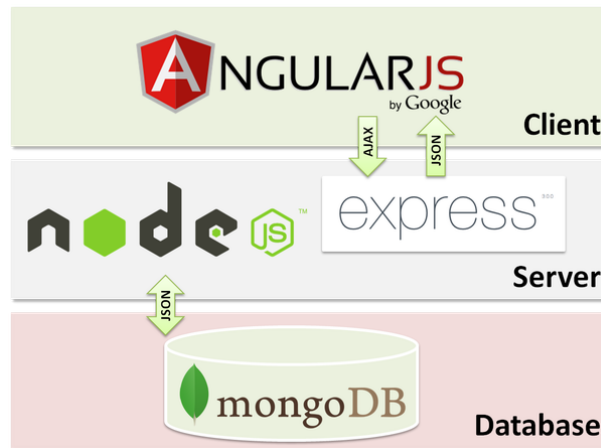
4.2 Development

The Mean stack is an open source JavaScript software stack. The MEAN stack consists of MongoDB as database, Express.js as a web application

frame, AngularJS as a front-end framework, NodeJS as a JavaScript runtime. These four technologies combine so that both the client side and the server side can be written in the same language. MEAN stack is primarily used for designing dynamic web applications. Mean supports MVC (Model View Controller) architecture. MEAN stack passes JSON between the different components.

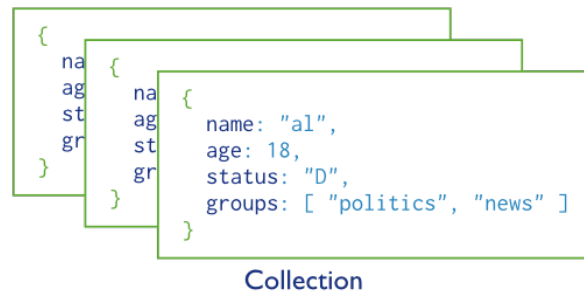
GitHub's statistics suggest that the use of components of the MEAN stack are rising rapidly. Because all the components are open-source they are free to use and are constantly being updated and improving.

MEAN Stack [5]:



4.3 MongoDB

MongoDB [6] is an open-source NoSQL database. It uses a document-based model to store data. The document based model it uses is JSON [7]. JSON uses key-value pairing to identify data. MongoDB uses collections and documents instead of using tables and rows like MySQL databases. Each MongoDB database is composed of collections and each collection is composed of documents.



Mongoose [8] provides an interface that connects MongoDB to NodeJS. Mongoose uses a schema based model. Mongoose handles query building, type casting, validation and business logic.

Example of Mongoose connecting to database, creating and saving a schema.

```

var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var Cat = mongoose.model('Cat', { name: String });

var kitty = new Cat({ name: 'Zildjian' });
kitty.save(function (err) {
  if (err) {
    console.log(err);
  } else {
    console.log('meow');
  }
});

```

This allows for fast saving of data. Each data model can be assigned its own schema and easily saved and retrieved. This allows for type safety of data if required.

4.4 Express

Express or Express.js [9] is an open source web application framework for NodeJS. It provides set of features for web and mobile applications such as creating API's. Express is a lightweight framework yet performs well.

With Express you can get HTTP endpoints up and running fast and easy. Methods such as GET, POST, PUT, DELETE from the NodeJS HTTP module are all supported by express.

Example of how HTTP routes Can easily be setup:

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage')
})

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage')
})
```

This simple minimalistic design means that API routes can be setup and modified easily. This helps speed up development.

4.5 Angular

Angular or AngularJS [10] is an open source JavaScript web and mobile application framework from Google. Angular lets developers to develop in a single language on multiple platforms, improves code reuse and performance. Angular is designed to be test friendly as components are separated from each other.

Angular applications use dependency injection which means there is no main method to be managed. Dependency Injection lets you ask angular for an instance of a component and Angular will provide it for you. This makes angular more maintainable and because the components are injected they can be easily be removed or replaced.

Angular is designed to improve upon HTML (Hypertext Markup Language). Angular improves HTML by adding a dynamic element to it. Angular does this using client side architectures and data binding. The main architecture Angular implements is Model–View–Controller (MVC).

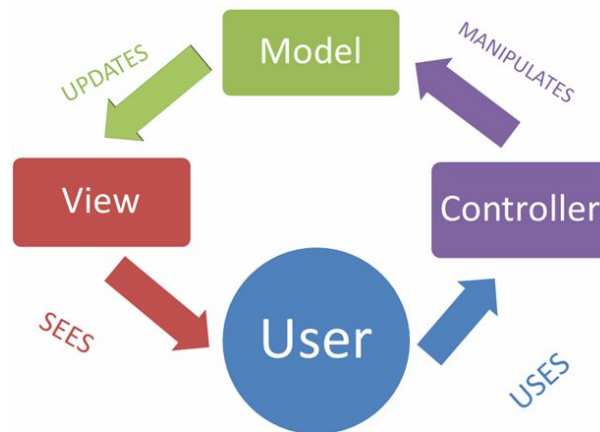
4.5.1 Model–View–Controller

Model–View–Controller is a software architectural pattern used for designing user interfaces. It splits the software into three separate components, the View which is what the user sees, the Model which handles the data and updates the view and the controller which the user uses to manipulates the data in the Model. Each component should be loosely coupled to other components in the software. This allows for parallel development of components

and components to be swapped in and out without affecting other parts of the system.

For example, a class or object in the model gets information from database and renders it for the View to display to the user. The user uses the Controller to edit the data, the Controller modifies the data in the Model and the View Updates from the Model displaying the edited data to the user.

Model–View–Controller example [11]:



Model

The Model is responsible for managing all the data and logic for the application such as manipulating data and getting data from external sources such as a server. It is also responsible for updating what the user sees in the View Component.

View

The View is responsible for all the User Interface logic. It handles everything the user interacts with such as text boxes, forms etc. It updates itself with data from the Model.

Controller

The Controller acts a medium between the Model and the View. It lets the user interact with the Model to manipulate the data it contains.

4.6 Angular 2

Angular 2 is an improved version of Angular. It replaces JavaScript with Typescript. Angular 2 has all the benefits of Angular 1.

Angular 2 made the following improvements:

- Code structure is simplified and it easier to learn as there are less concepts.
- Improves server-side speed.
- Improved client-side performance as it uses server side rendering for faster viewing.
- Provides support for newer and older browsers.

4.6.1 TypeScript

TypeScript [12] is a strict subset of JavaScript that is developed and maintained by Microsoft. Because it is derived from JavaScript, TypeScript can run any JavaScript code. TypeScript adds type safety and an object-oriented structure to the language. It uses many structures that common object-oriented based languages use such as Classes, constructors and Interfaces. TypeScript provides static typing through type annotations that enable type checking at runtime. Typescript is compiled to JavaScript which means it can run in any environment that supports JavaScript.

Example of Typescript class, interface, function and declaration of a class

```
class Student {
    fullName: string;
    constructor(public firstName, public middleInitial, public lastName) {
        this.fullName = firstName + " " + middleInitial + " " + lastName;
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person : Person) {
    return "Hello, " + person.firstName + " " + person.lastName;
}

var user = new Student("Jane", "M.", "User");
```

4.7 Node

NodeJS [13] or Node is an open-source, asynchronous event driven JavaScript runtime that is designed to build network applications. Node is highly scalable and can handle many connections concurrently. Node interprets JavaScript using Google's V8 JavaScript engine. Node is lightweight yet powerful and is quickly growing in industry with large companies such as IBM, Microsoft, Netflix, PayPal, SAP, Yahoo and Cisco using it. As Node was designed with network applications in mind it implements HTTP. The use of HTTP means Node is suitable for building web frameworks on.

Example of a simple server in Node.

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

4.8 Ionic

The Ionic Framework [14] is an Open-source mobile application software development kit. It is built on top of Angular and Cordova. It builds mobile applications using web technologies like HTML and CSS. Ionic applications use the mobile devices web browser to run. Because of this, applications don't

have to be written in the devices native code meaning that Ionic applications can use the same code to run on multiple devices and operating systems such as Android and IOS.

4.8.1 Apache Cordova

Apache Cordova [15] is a mobile application development framework that lets the native features of devices such as camera and location work with HTML. Cordova provides a wrapper to native APIs. These plugins can be installed by developers as required and allow developers to access the device's native accelerometer, geolocation, camera, file system and microphone etc. This greatly speeds up development as no knowledge of the native APIs are needed. Cordova supports a wide range of devices such as Android, Apple, Blackberry and Windows phone.

4.9 JSON (JavaScript Object Notation)

JavaScript Object Notation [16] or JSON as it is more commonly known is a syntax for storing and exchanging data. JSON is fast becoming industries most used data interchange format. JSON is a text-based format which makes it easy for humans to both read and write.

JSON is lightweight which makes it easy to parse and generate for machines. It is preferred in most network applications as it is less verbose than traditional methods such as XML (eXtensible Markup Language). JSON is based on JavaScript objects but because it is text-based it is language independent. Because it is so easy to use and generate most common languages have adopted JSON.

JSON is designed from two structures:

- A name/value pair
- Lists of pairs

These are common data structures. Most common languages represent name/value pairs as objects or structs and represent lists as Arrays or sequences. This makes JSON interchangeable easily between languages and allows almost any language to be able to communicate to another language.

The values that JSON supports are:

- Strings

- Numbers
- Objects
- Arrays
- true
- false
- null

Example of JSON name/value pairs:

```
{|
  "Fridge1": "Backup Fridge",
  "Fridge2": "Meat Fridge",
  "Fridge3": "Storage Fridge",
  "Fridge4": "Kitchen Fridge",
  "Fridge5": "New Fridge",
  "Fridge6": "Old Fridge"
}
```

JSON is used in many databases as its quick to store, retrieve and parse/load into applications. It reduces time taken by Database as it doesn't have to manipulate data, less verbose data can be sent and received faster and applications can easily parse the JSON data.

4.10 HTTP

HTTP [17] (Hypertext Transfer Protocol) is the set of rules followed for transferring files over the internet. HTTP runs on top of the TCP/IP suite. TCP (Transmission Control Protocol) is the main internet protocol for data transfer. It combines well with the IP (Internet Protocol) to form the TCP/IP suite.

HTTP allows for the transfer of files such as text, images, videos and sound.

HTTP uses the following methods:

- GET
- POST

- PUT
- DELETE
- HEAD
- OPTIONS
- CONNECT

These methods can be used to define what the request wants. The most commonly used Methods are GET and POST. GET requests data from a specified resource and POST submits data to a specified resource. e.g. Request wants to GET data or Request wants to Post image to server.

4.11 REST (Representational State Transfer)

REST [18] (Representational State Transfer) or RESTful is a method of communication for computers systems over the internet. REST is an implementation of HTTP methods. REST uses the same methods of that HTTP uses such as GET, POST, PUT, DELETE etc. This allows applications to benefit from the same style of communication the internet uses. REST is stateless. This means it doesn't keep track of sessions improving performance reliability and scalability. This allows servers to respond to requests quickly and to forget about it after. Because REST is web based, responses can be cached to speed them up.

Chapter five discusses how the above discussed technologies combine to create the system. The chapter discusses the role of each component and how the chosen technologies benefit the system.

Chapter 5

System Design

This chapter discusses the system design in detail. It talks about how the various components of the system such as mobile application and API server link together to create the system design. The chapter also covers the roles each component performs for the system.

5.1 Overview

The system consists of three elements working together, a Mobile application, a Web application and an Application Programming Interface. To get these separate elements working together a software architectural pattern had to be followed. After researching many different frameworks and technologies the decision was made to base the design on the MEAN stack.

This decision was largely influenced by the decision to use the Ionic framework. The Ionic framework uses Angular.js which is one element of the MEAN stack. The Ionic framework was chosen for the mobile application as it allowed the application to be developed once and be deployed on various mobile operating systems.

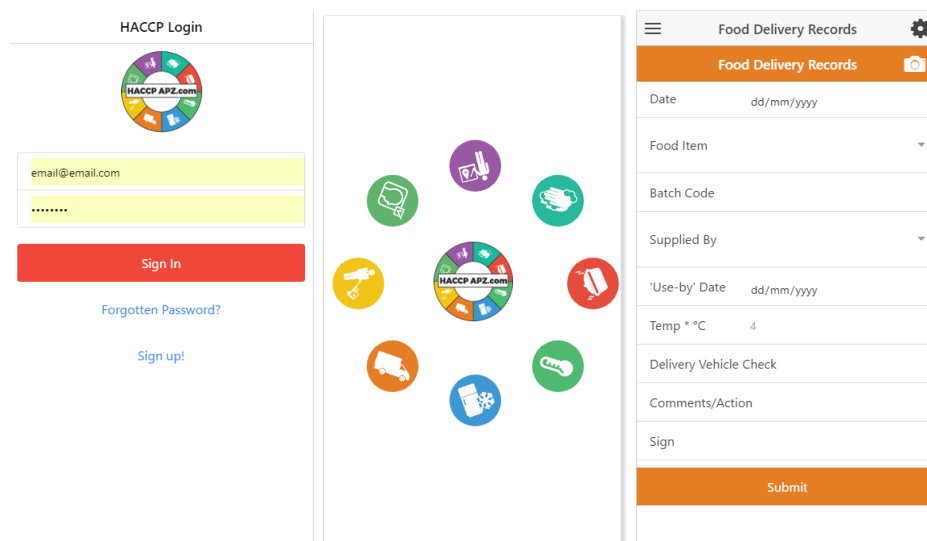
Applying the MEAN design to the rest of the project was a good idea as it meant that most of the technologies were JavaScript based. This meant that once a strong understanding of JavaScript frameworks was gained the expertise was passed onto the next element of the project speeding up development as new technologies hadn't to be learned from scratch. Because all the technologies are of a similar origin there are less compatibility issues than if other technologies were used.

5.2 Mobile Application

Firstly the mobile application was designed as this was what the client felt was most important part of the system for him. The main objectives of the mobile application were to create text forms, take and add pictures to relevant forms, store pre-programmable user settings and be cross platform for Apple IOS and Android. The decision to use ionic was made as research showed it was cross platform and it supported plugins for working with native camera.

The mobile application consists of login and sign up pages, home page, settings page and various form pages.

- The login and sign up pages handle Authentication and signing up to use the system.
- The home page links to other pages when logged in.
- The settings page handles editing and saving user defined settings such as foods and suppliers.
- The various form pages handle creating and handling form objects.



The main role of the mobile application is to manage the forms the user creates and to communicate with the server. The applications roles include form adding a geolocation if relevant to forms, pushing data to and pulling data from the server, saving form if no data or Internet connection is available and handling Authentication. The mobile application is the main provider of data to the server. The data generated is in JSON format.

5.3 Cloud Server

The server is the core of the system. It's the link between the mobile and web applications. Its main role is hosting the services that the other elements of the project depend upon. It was decided to use Windows Server to speed up the development process as it was previously taught in the course and because the local development work was done on a Windows 10 machine it would minimise compatibility issues when moving from local test server to the cloud. The system is hosted on the Windows Azure platform as this was provided by the client.

5.3.1 API Server

The API server handles communication throughout the server. It is responsible for handling Authentication and data. The API handles storing data sent to it and sending data when requested. The server also handles any manipulation of data that is required.

NodeJS was chosen to run the API server as it is lightweight, scalable and reliable. Node can handle requests with speed and is easily testable with the Postman application.

The API server also provides data analysis to the web application. It is responsible for analysing and manipulating the data according to the request's parameters. Node was a good choice for this as it's a JavaScript runtime it was designed to work with JavaScript and JSON. The server could receive the request, manipulate the data and return the request in a fast time as it wasn't depending or waiting on an external program to do the work.

Using a RESTful design meant that the server didn't have to worry about managing session. This allowed the server to perform faster. It also provided API routing using HTTP method calls. All any application needs to access the API is standard HTTP. This allowed for simpler testing and design of web and mobile applications as they didn't require complicated setup to communicate with the API.

5.3.2 Database

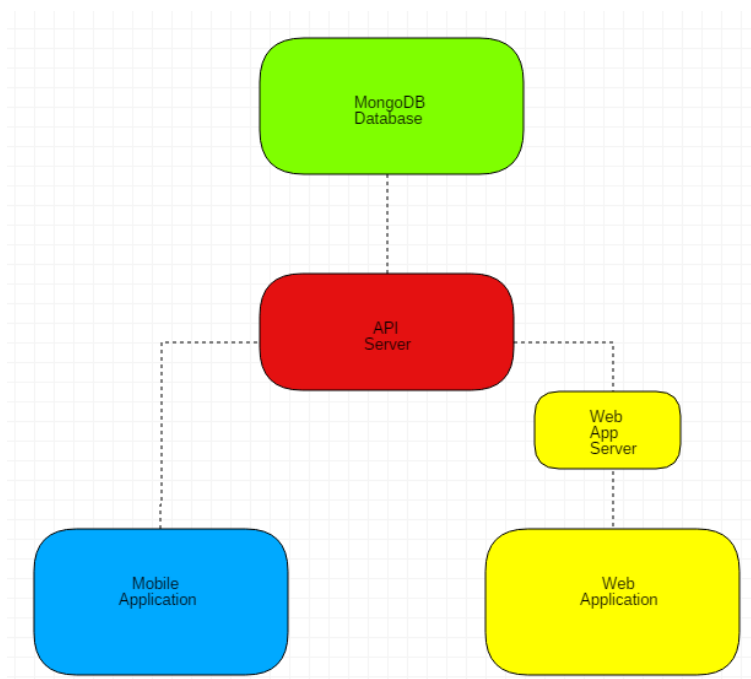
The chosen database was MongoDB. MongoDB was chosen because it stores data in JSON format and because of Mongoose. Mongoose allows Node to query the database easily and add structure to the data in the form of schemas. This is particularly useful if data is required to be in an exact format. MongoDB also allowed the storage of photos which meant MongoDB was the only database that was needed.

5.3.3 Web Application Server

The cloud server also has the job of hosting the web application server. The web application is written in Angular 2. Angular adds makes HTML pages dynamic. Angular can communicate with the API to get the information its pages require.

The application is running on the same cloud to reduce any latency but if required the server is designed to run from any IP and contact the API server remotely. This could be useful if multiple instances of the application server are needed to be served in different locations in the future adding scalability.

System Architecture:



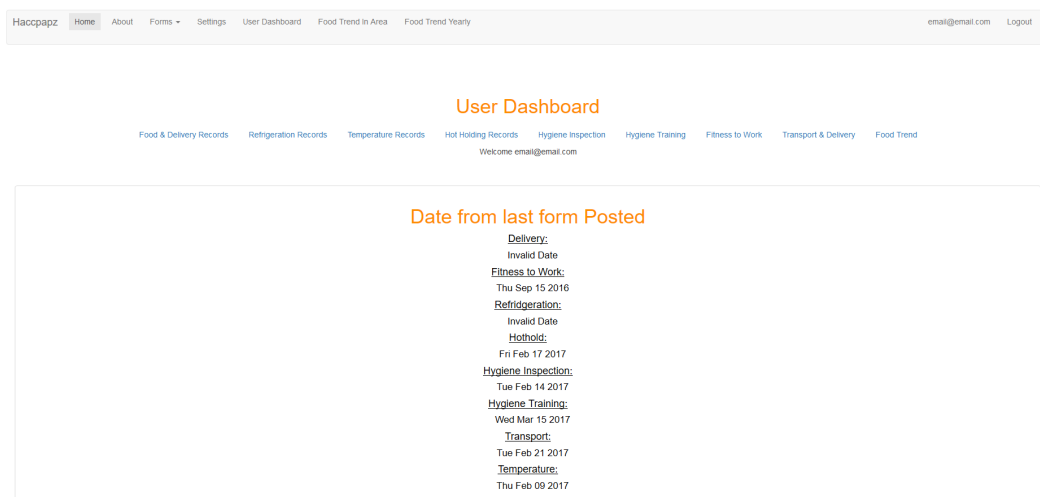
5.4 Web Application

The web application was the last part of the application designed. This was useful as an understanding of all the data that was going to be displayed was had. The application could be designed to work around the data rather than changing the data to suit the website.

The main objectives of the web application were to display the gathered data stored in the database from the mobile application back to the user and to create queries and display the result of the analysis of the data to the user.

The web application consists of login page, home page, about page, settings page, user dashboard, data analysis pages and various form pages.

- The login page handles Authentication of users.
- The user dashboard links to other pages when logged in and displays user information.
- The settings page displays the user defined settings such as foods and suppliers.
- The various form pages display the data from the database.
- The data analysis pages let Administrators query and view analysis of data.



The choice to use Angular 2 was useful as Angular could manage requests, responses, Authentication of users and manipulation of data.

Angular could provide and revoke access of different users to different pages. Users can only see their own data and cannot access pages without being logged in. Only administrators can perform analysis on the data. If an unauthorized user attempts to access a page they don't have permission for they will be redirected to the homepage or the login page.

This application uses the Observables from the RxJS (Reactive Extensions for JavaScript) library in combination with the HTTP requests to get and handle data from the API. Observables improve on Promises which are commonly used in angular to get data. While Promises are only called once

and return a single value Observables can handle multiple values over varied times. Observables can also be cancelled while Promises cannot be cancelled.

Example of an observable getting a collection of forms of type Hothold:

```
getHotholdForms (): Observable<Hothold[]> {
    return this.http.get(this.hotholdUrl, ({ headers: this.authHeader}))
        .map(this.extractData)
        .catch(this.handleError);
}
```

Angular also manages many powerful features such as Geo-location and Maps very well. The data analysis pages had to return the location of the search in its query. Angular gets the location of the user from the browser.

Example of code getting location when page initialises:

```
//on load check for navigator and get Geolocation if possible
ngOnInit() {
    if (!!navigator.geolocation) {
        //gets GetLocation
        navigator.geolocation.getCurrentPosition(this.setPosition.bind(this));
    } else {
        console.log("No support");// No support
    }
}
```

To allow the user to search from a different location Google maps was incorporated into the project. A map and location marker was added to the page. When used in combination with the browser geolocation the marker was placed at the user's current location if location was allowed and the user could move the marker to select the location to query from.

Example of Google Map embedded in HTML:

```
<div id="selectmap" class="box_div">
    <h2>Select your location</h2>
    <sebm-google-map [latitude]="lat" [longitude]="lng">
        <sebm-google-map-marker [latitude]="lat" [longitude]="lng" [markerDraggable]=true (dragEnd)="updateLocation($event)"></sebm-google-map-marker>
    </sebm-google-map>
</div>
```

Food Trend page:

Haccpapz

[Home](#) [About](#) [Forms](#) [Settings](#) [User Dashboard](#) [Food Trend In Area](#) [Food Trend Yearly](#)

email@gmail.com [Logout](#)

Food Trend

Search for Food Trends

Food:

Beef

Within how many Months:


2

Within how many Kilometres:

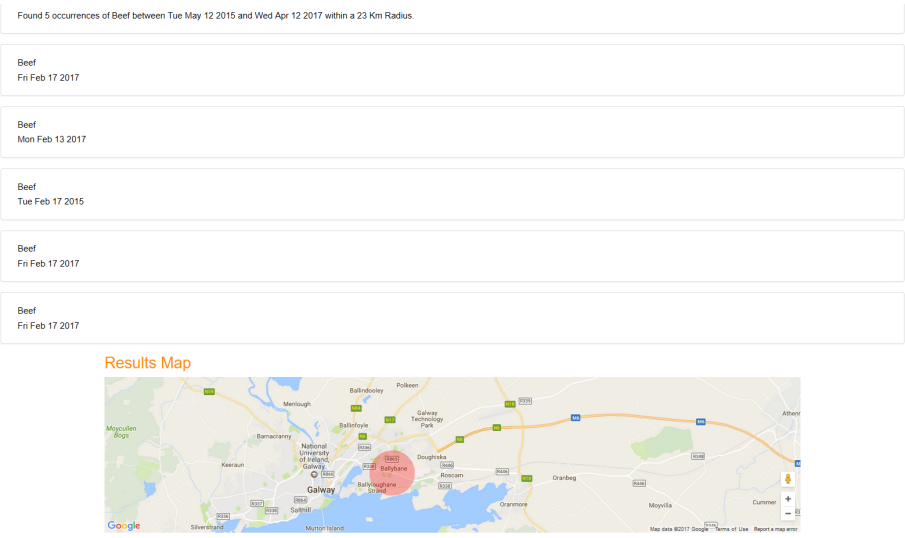
20

Submit

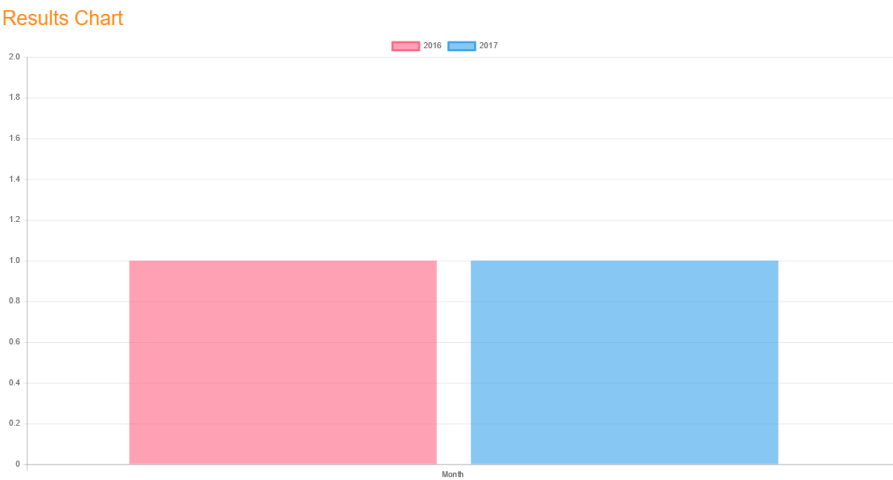
Select your location



Results map:



Results bar chart:



Chapter six discusses how the system was evaluated. It discusses the various types of testing that was performed to make sure the system met requirements.

Chapter 6

System Evaluation

This chapter discusses the various types of tests that were performed throughout the development life cycle to ensure software quality and that the requirement for the project were met.

6.1 Overview

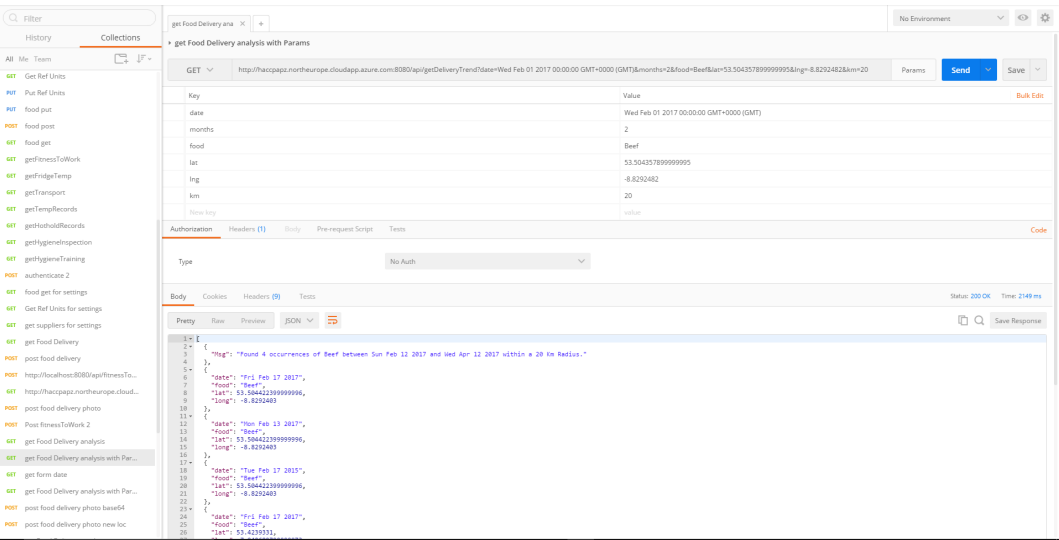
The system was designed with test driven development in mind. As each unit of code as written relevant test were ran or new tests written if needed. Because test driven development was used during this project a lot of bugs were caught at an early stage and weren't built into the system.

The types of testing used included:

- Unit testing
- Integration testing
- Functional testing
- Usability testing
- Acceptance testing

6.2 Unit Testing

As each unit of code such as a HTTP Method was finished it was tested using the Postman [4] application. Postman could send a request and view the result. The results could be examined to see if it passed test.



6.3 Integration Testing

When new components were added to the system tests were ran to see if any other components were affected by the changes. The whole system was tested to see how well the different components worked together.

6.4 Functional Testing

Each section of the system had its own functional tests to see if it was running as intended.

6.4.1 Mobile Application

The tests for the mobile application were:

- User can login.
- User can sign-up.
- User can access each form page from menu.
- Each form page submits data when submit button clicked.
- User can view, edit and submit settings to server from settings page.

- User can take photo and submit photos with form if relevant to form.
- User can submit forms saved in local storage from settings page.

6.4.2 Web Application

The tests for the web application were:

- User can login.
- User will be redirected if not logged in.
- Only administrators can access Data analysis pages.
- Each form page can view its relevant forms from the logged in user.
- User dashboard displays last date for each form submitted.
- Administrator can enter food, date and range on food analysis page.
- Food location shows up on results map on food analysis page after results returns.
- Administrator can select location using Google maps on both analysis pages.
- Administrator can enter food, range, year one, year two and month on food analysis yearly page.
- Results bar chart displays and updates on food analysis yearly page.

6.5 Usability Testing

During the weekly meeting and emails the client gave his opinions about the latest additions to the project. If the client wanted something changed it was discussed and an agreement was made on the steps to undertaking to improve the issue.

6.6 Acceptance Testing

Prototypes of the system were distributed by the client so potential customers could test out the system and give feedback about how useful and user friendly the system was to use. This feedback led to changes to increase the friendliness of the system for the user.

Chapter seven contains the conclusion of the project. The chapter reviews and evaluates the overall project.

Chapter 7

Conclusion

The Objectives of this project were to create a system that digitalised a paper based system and saved all the data on the cloud. The system lets users save their forms on their mobile devices and view them from the web using the web application and for the client to be able to perform analysis of the data.

One of the client's main aims was to get a system up and running so he could start to get his idea off the ground. The current system is designed with scalability in mind to grow with the business.

The current system completes the objectives set at the beginning and has incorporated a lot of the features and ideas the client and supervisors had suggested throughout the development process.

The client has a system from which he can use to grow the idea. Currently the system can provide analysis for food deliveries. The client can search deliveries of a given food within a given range within a given time e.g. Amount of beef deliveries within 2 months and within 20 kilometre range from a given location. The client can also compare the amount of a given food on yearly basis by a given month e.g. Amount of beef deliveries in area in January 2016 vs January 2017. This information can be used or sold by the client once the system grows in popularity.

As this was a new system it was hard to perform analysis and to know what trends in the data could be uncovered. Once more customers start using the system more useful and interesting analysis can be performed on the data.

If this project was to be repeated Docker could be incorporated. Recently Docker was covered on the course. The benefits that Docker would add to this project would be the ability to setup services faster and create multiple servers that work together to provide load balancing and increase scalability. As needed new server instances could be easily created and connected to a swarm of servers. This swarm design means that if one server went down

the other servers would compensate for it until it came back online. This design would mean that it would be very rare that the system would be offline increasing reliability.

The overall look of the web application could be improved but the main objective was to get the system working and the client has budgeted for a front-end designer to come in and improve on the look and feel of the web application. Improving the look of the website would be added as an enhancement if repeating the project.

This was an enjoyable project to work on. It provided great experience like learning how to work with a client as well as learning new technologies and implementing a system architecture. The project showed how beneficial it is to use a development methodology and how to work with deadlines. It also provided experience on how to manage a project from start to finish and that projects are always open to change and software should be developed so that it can accommodate change. This project has provided an insight on what it is like to work on industry level projects.

Chapter 8

Appendices

8.1 Source Code

All the source code used in this project is available on GitHub. At the first link below you will find the source code for the mobile application. At the second link, you will find the source code for the web server and the API server.

8.1.1 Web Servers

<https://github.com/Martinc94/HACCPAPZ-Server>

8.1.2 Mobile Application

<https://github.com/Martinc94/HACCPManagementApp>

8.1.3 Instalation

Both repositories contain their own README.md file that has information about the project, how to setup the required environment and how to run both servers and build the mobile application.

Bibliography

- [1] I. Newman and C. Benz, *Qualitative-quantitative Research Methodology: Exploring the Interactive Continuum*. Southern Illinois University Press, 1998.
- [2] “<http://www.kean.edu/~tabraham/mis/development/developnotes.html>.”
- [3] “<http://www.agilenutshell.com/>.”
- [4] “<https://www.getpostman.com/>.”
- [5] “<https://www.quora.com/what-are-the-advantages-of-developing-with-the-mean-stack-mongodb-express-js-angular-js-node-js>.”
- [6] “<https://www.mongodb.com/>.”
- [7] “<https://www.mongodb.com/json-and-bson>.”
- [8] “<http://mongoosejs.com/>.”
- [9] “<https://expressjs.com/>.”
- [10] “<https://angularjs.org/>.”
- [11] “<http://csharpcorner.mindcrackerinc.netdna-cdn.com/uploadfile/3d39b4/asp-net-mvc-introduction/>.”
- [12] “<https://www.typescriptlang.org/>.”
- [13] “<https://nodejs.org/en/>.”
- [14] “<https://ionicframework.com/>.”
- [15] “<https://cordova.apache.org/>.”
- [16] “<http://www.json.org/>.”
- [17] “<http://searchwindevelopment.techtarget.com/definition/http>.”

- [18] “https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.”