iOffer - The web advertising application of tomorrow

Niks Gurins

Ed Lasauskas

Pawel Borzym

Gediminas Saparauskas

B.Sc.(Hons) in Software Development

April 15, 2017

Final Year Project

Advised by: Dr. Brian McGinley

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract For our final year project we were looking to create an industry-standard application that would help people get exposure for their unwanted items in a unique way. We as a team created a free platform for people to buy and sell their items, however, we do not handle any of the transaction part so it is more of an advertising medium. We've all been there, looking to sell an item but being unable to get enough exposure for it. We created a web application as well as a native android application with a Java back end server and MongoDB database for both. iOffer also provides a RESTful API for ease of interaction and security purposes. We've also used the Google maps API to bring a fresh new style of getting your item out there to the public.

Authors Niks Gurins - `https://github.com/Niksdope` - was in charge of the Java back end.

Ed Lasuskas - `https://github.com/edvalas` - was in charge of the database and all the different data models.

Pawel Borzym - `https://github.com/PawelBor` - was in charge of the front end development for the web-application

Gediminas Saparauskas - `https://github.com/saparasource` - was in charge of the front and back end connection for the Android application.

Chapter 1

Introduction

This is our 4th year Software Development project. The project is both a web application and a native android application. Both parts of the project will created using the following technologies.

Technologies Used:

- Hyper Text Markup Language (HTML) 5, Cascading Style Sheets (CSS) 3 & JavaScript (JS)
- Bootstrap + jQuery
- Java Enterprise Edition (JEE)
- JAX-RS, Jersey
- Apache Tomcat
- MongoDB
- JavaScript Object Notation (JSON)
- eXtensible Markup Language (XML)
- Google Maps Application Programming Interface (API)
- Facebook Developer API
- Amazon Web Services (AWS)

Our web application and the native android application are, for the most part, the same application. We originally planned to do only the web-app side of things. During the planning process we felt like the scope of the

project is too small and we are not covering enough technologies. We then decided to add in a native android version of our web application to broaden our project scope and technologies.

After some development on localhost(running and testing the app locally), we deployed our web application to an AWS virtual machine.[1] We want to simulate a real life web site that runs on a remote server rather than only working on our computers in the local host domain. Another reason why our web application is deployed to the remote server is because we need the REST (Representational State Transfer)ful API (data) to be accessible by our native android application.
We are also using a Google API for Google maps.[2] The map is used to display user advertisements as markers on the map. When browsing the map, the users are able to click on a given marker, which then displays information about the advertisement.
The purpose of the web application and the native android application is to deliver information to the user about current offers from businesses or private persons that are looking to sell their items.
The user will be able to post an item for sale, which will fall under a desired category e.g. Electronics, Car Parts or Food offers around the user's location. In order for the seller to post an item they would need to complete a registration form and log in to the system. Once logged in, the seller can post an advertisement/item for sale by providing the necessary details, like the name of the product, the price, location, phone number etc.
The shopper will have the freedom of registering or just using the application without the need to register. Without being logged in, a user that is browsing items or advertisements will not be able to comment on the product page. Sharing advertisements to Facebook can be done without being logged in.
The shopper is not actually buying anything on the web application or the native android application but is rather just accessing the advertisement and the contact details of the poster.
Users are also able to browse items for sale in the application by using search criteria or simply view all of the products for sale around their location on the map display.

1.1 Objectives of this project

When talking about the objectives or learning outcomes of this project, a few things come to mind.
Firstly, the main objective is to learn and broaden our working knowledge

of the technologies that will be used in this project. This project covers a large variety of front end and back end technologies as well as a ”schemaless” database. Some of the technologies in this project we had never used before, like MongoDB.Through this project we acquired good, hands-on experience of MongoDB as well as other technologies that we had never seen before, or had covered briefly. Working with all of these technologies and connecting the front end with the back end Java server further sharpened our programming skills and techniques. We learned more about and how to make a web application restful, which made it possible for our native android application to access the data it needs for functioning, from the server. Using a Tomcat server and deploying our web application to a remote virtual machine also gave us more experience working with a client and server architecture.
Since this project was to be developed during the course of 4 to 5 months, we gained lots of experience with building a project from start to finish and all the planning that took place in between. A lot of the smaller projects we've been working on are small in terms of scope and do not take such a long time to develop. Usually, these type of projects involve less time spent on the refinement, compared to that of iOffer.
This was a great opportunity to learn how to continually build up and work on a project over a longer period of time. This gave us more experience at becoming better at time management and incremental goal setting for the project. We managed our time wisely, by setting up sessions to work on the project together and managed our goals and deadlines by setting out a weekly development plan for the project, as to have an idea of where we are and where to focus our energy.
Our main metric, by which we measured success or failure of this project, was the ticking off of goals that we set out at the start of the week. We used our weekly plan to stay on track during the development of this project.

1.2 GitHub

The GitHub link to the project repository can be found below:
`https://github.com/PawelBor/Year-4-Offer`
In our GitHub repository we have a README and an MIT LICENSE file. The repository also contains 3 folders: Android, java and webapp.
The Android folder contains all of the back end and front end related code that the app needs to work.
The java folder contains 2 folders within, which are rest_api and service. In our rest_api folder, we have all of the java classes we need to make our applications fully restful. You will find classes and Unique Resiyrce Identifiers

(URIs) for the restful services such as getProduct, which when accessed at the correct URI, webapi/service/product/{id}, will return a JSON representation of the specific product id found in the URI, which corresponds to an item from the database, if one exists with that id. The other folder, service, contains java classes which are like supporting and utility classes. We have classes like Base64 encoder, which is used to encode images to a Base64 string. We have classes like Comment, which are just used to make our server more object oriented.
The webapi folder contains everything to do with the front end of the web application. It contains things like our Java Servlet Pages (JSPs) pages that are served as well as images, CSS other things used for the front end of the app.
Lastly our GitHub repository contains the README file. This file has information about the project, team members and supervisor. It also contains documentation that we incrementally updated over the course of the project development. Most of the documentation is brief descriptions of the technologies we are using in our projects, examples of data required, examples of data stored on our database and some diagrams, like the overall system architecture diagram found further in this document.

1.3 Sections in this document

- Methodology
- Technology Review
- System Design
- System Evaluation
- Conclusion
- References

1.3.1 Methodology

This section of the document describes how we went about developing our project. We talk about our approach to developing the web application and the native android application as well as how we carried out our research before commencing the development of the projects. We cover, in this section, how we went about meeting our supervisor, how we carried out development sessions together, how we managed our GitHub repository and the approach

we used for the development of this project. We also cover why we chose the programming languages, technologies and platforms for the project. Lastly we talk about how we dealt with and solved the problems we encountered during the development of the project.

1.3.2 Technology Review

This should be the longest and most contextual part of this dissertation. In this section we cover in detail and explain all the technologies we used in our projects. We talk about what kind of standards we used and what kind of Database Models are created and used by the system. We also talk briefly about how to set up the individual technologies, how we set them up and what we used to interact with them. We discuss our technologies under 4 headings: Front end, Back end, Database and Android.

1.3.3 System Design

This section of the dissertation provides a detailed explanation (using pictures) of our overall system. It describes each individual part of the system. Front end, Back end, database and Android are all of the individual parts of our overall system. Diagrams will aid our explanation on how the system works and interacts with each other as a whole. Diagrams like overall system architecture and navigation flow chart will help us in this section. We will also talk about how we deployed our projects to a remote AWS virtual machine and how that affects our system.

1.3.4 System Evaluation

In this section of the dissertation we cover how our end result for this project compares to what we set out to develop from the start. We will also discuss how our approach to the development of the project provided us with some opportunities based on some of the technologies we chose to use for project and also how our choices provided some limitations to the development of the project.

1.3.5 Conclusion

In this section of the dissertation we will summarise the context of the project and all the objectives we set out to do. The outcome for the project was to learn how to work with a multitude of technologies and how to get them to work together. We will talk about the actual outcomes of the project, any

tangential or even unrelated insights that we got throughout the development process, and what we'd do differently if we were to do it all again.

1.3.6 References

Here we will include all the references to third-parties (e.g. StackOverflow) that helped us overcome problems and make us understand what went wrong.

Chapter 2

Methodology

2.1 Software Methodology

Our approach to the project was mostly that of the Agile methodology.[3] At the start of the project we set out our whole project scope. After it was decided, we divided the different parts of our web application and native android application between ourselves.
Each team member was assigned to be responsible for some part of the software that we were going to produce. Pawel was assigned to manage the front end development for the web application. Niks was assigned to manage the Java server behind the web and android application. Ed was assigned to manage the database for the web and android applications. Finally Gediminas was in charge of the front end of the android application and connecting it up to the restful service and database.

2.2 Research Methodology

For our research, before commencing any development of the project, we carried out some research on in our own time. We researched our technologies and set them up to be able to work with them. We also researched the Google maps API and the Facebook API as we were going to use them later in our web application. Once we were done with our research, we met together and explained to one another what we have researched to make sure everybody is familiar with all of the technologies we were going to use for the development of this project.

2.3 Planning

In the initial week of development we spent as a group discussed potential ideas for our project. We came up with a few and eventually narrowed it down to only 2. Then we drew up the architecture of our 2 project ideas. We presented and consulted these 2 project ideas with our project supervisor. After hearing our project supervisor's feedback on our 2 project ideas, we as a group decided to go with what is iOffer now and our supervisor approved of our choice. Next week we discussed the overall scope of the project and we decided(with the help of our supervisor) that we need to add a native android application to increase the project scope and to cover more technologies.

2.4 Meetings

As a group we set up development sessions during college time in the library. We would go as a group and work in the library on our own individual parts that we initially assigned to be responsible for. Soon after, Niks and Ed begun working together on the server and database. Since MongoDB does not have any schema or tables,it was quite easy and quick to set up. After setting it up Ed moved to the server development with Niks. While Niks worked on things like handling Hyper Text Transfer Protocol (HTTP) requests with the server, Ed worked on the database related things on the server. We first got most of the server developed using Java servlets. Later on in the development process, we revisited our project as a whole and came to the conclusion that the Java web servlets just won't cut it. We realized that our Java web servlets would be impossible to connect with our web native android application so we decided to make the server RESTful, making the connection possible as well as more secure.[4] It is more secure using the RESTful API as we do not directly connect to our database. At the start we had a few problems when trying to get our server to work as a RESTful API. Eventually Gediminas got some of the URI's working. Gediminas showed the rest of the group how he got the RESTful URI working and how to work with the URI's. Then Ed and Niks took over the server development to remake the rest of the server to be fully restful. Meanwhile, Gediminas and Pawel we were busy, creating and re-creating the front ends for both platforms.

2.5 Structure

The structure of our development sessions was as follows. At the start of each session, each member had an opportunity to talk about what they have done from the previous session, what their plan is for the current session and what are the possible problems that may arise during the development.
Our group met with our project supervisor on a weekly basis. We would meet at the same time each week and discuss the weekly progress. We would show our supervisor any progress we made and would discuss our next steps in the development of the project and the project supervisor would also give feedback on the current progress. We would make any adjustments if there was any based on the feedback we received.
During the development process we would refer back to our original weekly development plan. This allowed us to track our development progress and "tick off" different goals for the project.

2.6 Testing

For testing, to make sure everything was working as intended, we carried out some black box testing. We let our friends and supervisor use the website and the native android application and by doing this we found out about bugs and parts of the project that were not working as intended. Once a bug or any abnormal activity was found, it was dealt with pronto.

2.7 Problems

During the development of the project we encountered only a small number of significant problems and plenty of minor bugs. Our significant problems were passing up images from the front end file selector using a the RESTful services, re writing our server to use RESTful URI's rather than the Java servlets and re-packaging our project as a Maven project to allow an easier to handle all of our dependencies for the server.
When we encountered our problems, to deal with them, we spent some time identifying the root cause of the issue to find a better way of finding solutions for it. We researched multiple solutions to the problem. After discussing our possibilities, we chose the best option and most efficient way for us to solve the problem. We also issued a GitHub issue on our GitHub repository and assigned certain team members to work on the problem. We selected team members whose area of responsibility was related to the issue. Once the

problem was resolved, if it had an outstanding issue on GitHub, the issue was subsequently closed and the problem was resolved.

2.8 GitHub

Since this is a group project and we have 4 team members, we needed to use some sort of a project version control manager. For this we chose to use GitHub[5]. We had been using GitHub for the last few college projects so it made perfect sense to follow the trend. Using version control tools encourages the development approach to building up projects over time by committing smaller pieces of code frequently rather than committing huge pieces of code at once. GitHub allows users to see a commit history, letting team members see exactly whats 'new'. This also allows us to go back to previous versions of the project if something goes wrong in the development process, and rollback the changes. GitHub also has an issues section for every repository, which allows users to create issues for other team member to see. We have issued a few issues on our GitHub repository and assigned a team member to solve the issue. We tried to incrementally commit small pieces of new code we added as much as we could to take advantage of our version control tool.
During the development of the project all team members were also contributing parts to the README file. This was mostly so that we don't forget what we did and how we did it. Now, when writing this document, we can refer back to the README to recall earlier stages of development for further documentation in this dissertation.

2.9 Selecting Languages & Technologies

When selecting what languages and technologies to use, we as a group decided to use some familiar and some new technologies. We chose Java for our server as we used Java a lot before but not in the JEE environment. We have not written client server application using Java beforehand. We chose Java to learn about developing server side, enterprise applications and also to improve our skills using the Java programming language. For the front end we chose to go with HTML5, CSS3, JS and JS libraries like Bootstrap and jQuery. We chose to use MongoDB as we will not need to have any of the ACID properties of databases in our project.[6] Our applications do not involve any transactions and have quite a variety of data, therefore a non Structured Query Language (SQL) document database was a good choice for us. We had also never used MongoDB prior to this project, so there was a

learning opportunity for a new technology. We chose to use Base64 encoding for our images rather than a library of MongoDB as we will not be storing files over 16 megabytes in size. We choose to use AWS for our remote virtual machine to host our web application as we got a student package account, which had free credit to run virtual machines.

Chapter 3

Technology Review

3.1 Android

We decided to create a native android application to increase the scope of our project.

Android is an operating system developed by Google.[7] Mainly used in mobile phones and other smaller hardware devices mainly using the ARM architecture.[8] The android application is developed for Android devices running version 2.2 or higher of the android OS. This is due to the Google Maps API requirements and stability issues that might be encountered by lower versions.

The application was developed in the ”Android Studio” Integrated Development Environment (IDE). We chose android studio for the simple reason that Google has stopped supporting android tools for the eclipse IDE which makes android studio the only viable IDE currently supported by Google.

The IDE provides functionality to compile the code into an APK file (Android Application Package) that then can be loaded onto the devices running the Android OS. The newer android versions also contain a pre compiler which pre-compiles the apk file once it is downloaded for faster loading of the application which greatly increases the speed of the services as compared to the website.

Written in the java programming language and using XML. XML is a way of structuring items/objects into an easily readable and parsable structure. In Android development we mainly use XML files for property files, gradle

files and layouts / User Interface (UI).

I used the RESTful web API which returns JSON data on a web request. I can then parse and use this data to access and modify all data from the MongoDB database. The web API was developed in JEE using the eclipse IDE with JAX-RS.

Google Maps API was used to display the map to show all markers of each product added to the MongoDB database for the visual representation of offers around the clients GPS location. Google Maps is a web mapping service developed by Google providing a really good and easy to use API for displaying interactive maps and locations.

I also utilized the GPS location of the client to a fine location so I could get the most accurate location. The negatives of this is that it takes a longer time to find the location and this uses a lot more of the battery compared to non fine location tracking.

The Facebook API was also used for the sharing of products and the application itself to the users' Facebook wall.

3.2 Web Application

We've decided to create this web application using HTML5, CSS3 and JS for the client side simply because it can be accessed by any device. For the design, we're going to use Bootstrap. Bootstrap comes with jQuery which we can then use for ease of HTTP Requests.
We have also decided to create a native android application alongside the web-based technologies for improved user interaction and User eXperience (UX). The main development language will be Java with all necessary Google API's and data-interchange formats.

3.2.1 HTML

HTML is a standard language used for the creation of web pages and web applications. It describes the structure/architecture and overall appearance of the information on the Internet. Web browsers receive documents from web servers and render them into multimedia web pages. Keywords and tags

are used to format and demonstrate the content of Web Page. The layout of HTML follows a tree structure called the Document Object Model (DOM) and is validated on page load. [9]

3.2.2 CSS

CSS is used to define how the HTML elements are to be laid out on the screen and, in general, how the user will see it. It's highly beneficial as the CSS document can manage the layout of not just one page or just specific item on the page but every single page in the project.
For example, a website made up of 5 different pages has a button on each page. The CSS can manage the design of every button to be the same no matter what page it's on. CSS can be internal or external. Internal CSS means that the CSS will be inside the HTML document and the downside of this is that it will only affect the page containing the code rather than all the pages. External CSS on the other hand is saved as a separate file in the solution with the .css extension and it may be used by all pages in the project. [10]

3.2.3 JavaScript

JavaScript is a dynamic programming language meaning operations that would otherwise be done at compile-time can be done at run-time. In JavaScript it's no harm to modify the type of a variable or add properties or even methods to an object while the program is running during the modification. It is also one of the three core technologies of the World Wide Web (WWW) content production. Most websites use it and all modern web browsers support it without any need for extra plug-ins. It's a prototype-based language with first-class functions, making it a multi-paradigm language that supports Object-Oriented, Imperative and functional programming styles.

JavaScript is applied to a HTML document and it can provide huge, dynamic interactivity on the website. Ranging from visual effects such as fade-in or hover effects and responsive click events on html elements to properly implemented front end to server connection that will get the necessary data for the website. On top of the core JavaScript language, developers build multiple tools that unlock extra functionality with minimum effort.
[11]

These tools include

- API's which are built into web browsers, providing dynamic creation of HTML, CSS and more.
- Third-party API's allow for the incorporation of functionality from other providers, such as Google or Facebook.
- Third-party frameworks and libraries that can be applied to HTML to speed up the development process of sites and applications, such as Bootstrap.

3.2.4 Bootstrap

Bootstrap is a front end web framework used for designing web applications and web sites. It contains HTML, CSS and JavaScript based design templates for various things like design and scaling of buttons, forms and many other interface components.
Bootstrap also incorporates SASS and LESS which are two of the most popular preprocessors of CSS. Preprocessors are simply programs that take one type of data and converts it to another type of data. Bootstrap also allows for a single codebase that efficiently and easily scales websites and applications across many different platforms, meaning that websites designed using Bootstrap will work perfectly on a mobile device scaling all the way up to a large screen television.
Bootstrap is not only limited to the static design of a web page or web application. It also provides JavaScript components in the format of jQuery. They provide the developer with extra useful UI elements, like tooltips or carousels. Bootstrap can not just provide extra UI elements but extend current and existing UI elements like auto-complete function for input fields. [12]

3.2.5 Facebook Developer API

Facebook's Software Development Kit (SDK) for JavaScript, is simply a set of software development tools that provide certain functionality. The Facebook API allows for the usage of its functionality, the following are examples provided by the Facebook API.
The enabled usage of the LIKE button and other social plugins on websites or web applications. Furthermore, it allows for the usage of the Facebook login facility, the connection to the Facebook Graph API and the usage of launch dialogs that allow for sharing content on the user's personal page.
Facebook's Graph API enables the developer to read data and write data into

Facebook. It presents a simple view of Facebook's social graph, representing objects in the graph and the connections between them. Objects could be people, photos or events and connections could be relationships between users, photo tags or shared content. [13]

3.2.6 Google Maps API

The Google Maps API allows the developer for multiple ways of embedding the Google Maps into web pages or the retrieval of data from them. It is not only limited to this. The API allows for further customization.
There are plenty different offerings from Google of this service. The Web API's that include Maps JavaScript API, static maps, street view and maps embed API. There is also Web service API's like Directions, Geocoding or Roads and finally Mobile API's that include Android API, Places API for Android, Maps SDK + Places API for iOS.[2]

3.2.7 jQuery

jQuery is a cross-platform set of JavaScript libraries designed to make the client-side scripting of HTML a much easier process.
jQuery takes a lot of common tasks that require many lines of JavaScript lines and puts them into methods that the developer can call with single line of code. It also simplifies Asynchronous JavaScript And XML (AJAX) calls and DOM manipulation.
Following are the features of jQuery: HTML/DOM manipulation, CSS manipulation, HTML event methods, Effects and Animations, AJAX + Utilities. [14]

3.3 Back End

3.3.1 REST

RESTful web services have become the de facto standard for web applications that use HTTP requests. A RESTful web service uses HTTP explicitly, e.g. a DELETE HTTP request should be used to delete records from the database. A RESTful web service should be stateless, meaning that requests should contain all data needed to be processed which makes a service scalable as independence of requests means they can be processed by multiple servers without the need for state to be held locally. Furthermore services should be exposed through URIs, which are links to send your request in order to

trigger a response from the server and requests should be transferring JSON or XML data, or both.[4]

3.3.2 Jersey

For our project, we used Jersey, a Java API for Restful Web Services developed by Oracle. This library allowed us to create RESTful services inside our application with ease. Creating services meant exposing URIs to interact with our data which allows not only us to use them, but anyone that knows how. For example, to add an item to our database, the user needs to send a POST request to the URI "... /webapi/product" with all the details needed to create one. [15]

3.3.3 Java EE

Java is an object oriented programming language written in C. An object oriented programming language is one in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. .Java can be written cross-platform as it runs on a Java Virtual Machine (JVM). The JVM works a little like that of a standard Virtual Machine running on any Operating System. It provides an environment for the java to compile and execute on your computer.
The Java EE differs from the Standard Edition (SE) by allowing programmers to create much more scalable and stable software. The EE suite adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server. We chose to use Java EE because we had little prior experience developing enterprise standard applications. [16]

3.3.4 Maven

Maven is a build automation tool that was built with Java EE in mind. It does this, by describing how the software is built, and what dependencies it has in order to run correctly. Adding a maven dependency will automatically download the library next time the project is ran, and glue everything together under the hood, saving time and effort when building a large-scale application. [17]

3.3.5 Amazon

We used Amazon web services in our project to deploy our web application server and database to a remote virtual machine. Amazon web services are a part of Amazon.com, and they provide a suite of cloud computing services such as hosting virtual machines or servers. We needed to deploy our server and database so our native android application has a remote host to send requests to retrieve the data it needs to function. As it is a mobile application it cannot request data from a localhost source. Another reason for deploying the web application on to a remote host is to simulate a real example of a client & server system. This is achieved by having our server and database up in the cloud and having the user accessing our application through a remote URL and request the data from that remote machine on our amazon virtual machine. [1]

3.3.6 Tomcat

Tomcat is an open source java servlet container. It allows to run web applications on a localhost server to access these applications though a URL. Tomcat is developed by Apache and implements Java EE specifications like servlets, JSP pages and web sockets, which allow for java web server development. We chose to use Tomcat as it is commonly used with java web applications and we needed to be able to run out web application on a server to allow server functionality for our application. Without running our application on a server, it would just be a web site without any functionality behind it. [18]

3.3.7 JSON

JSON or JavaScript object notation is a very commonly used data interchange format. JSON is a human readable and writeable data format and machine are able to parse it with ease. It is based on a subset of the JavaScript programming language and it is represented in normal text format. JSON is language independent and it uses convention that are familiar to any object oriented language. JSON is represented as an object, which contains name and pair values eg. “name” : “john”. JSON is a universal data structure and most modern day programming languages are able to supports the JSON data format as it is based them. [19]

3.4 Database - MongoDB

3.4.1 Prerequisites

To get MongoDB up and running, you must download the MongoDB installer or one of the zip folders which include portable MongoDB files, which do not require installation from their website. Once finished installing or getting the portable MongoDB folder, make a folder on your computer like this: C:\data\db. This folder will store the data from the MongoDB database. You can also set up to use MongoDB in text editor e.g. IntelliJ to interact with MongoDB. IntelliJ has a plugin called Mongo explorer, which opens a command line terminal to type Mongo commands and interact with the database.

MongoDB is a cross platform document oriented database. There are no structured tables like in relational databases. Mongo uses dynamic schemas, similar to JSON-like documents, to store data. Documents are stored in collections and are indexed by MongoDB to keep track of the order of the documents inserted. Mongo is used to handle diverse data types, fast queries and for ease of scaling the data. [20]

3.4.2 Why MongoDB

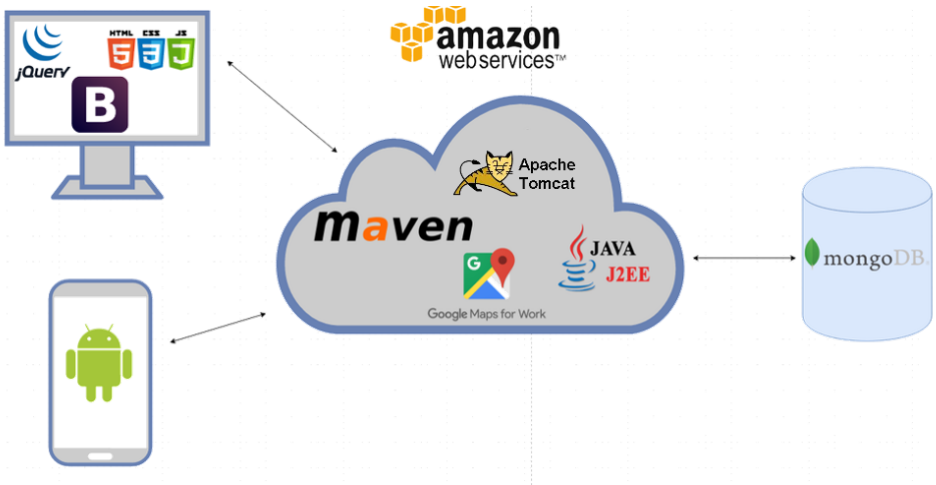
We decided to use MongoDB for our project as we will not be dealing with complex transactions on our website and in the database.
Another reason why we chose MongoDB is to learn a new technology and learn about using NoSQL databases.[21] Prior to this project, we have only worked with SQL or SQLite databases in various modules or projects for college. We have briefly covered using CouchDB, which is a JSON document oriented database like MongoDB. CouchDB uses REST and HTTP requests to interact with and has limited ACID properties, while also utilizing the use of map and reduce techniques for representing data. [22] MongoDB also seemed to be easier to use and learn, hence why we chose to go with MongoDB rather than CouchDB.

Chapter 4

System Design

4.1 Architecture

This project is based on the client & server model. We set up a remote server for our project by hosting our Java server and database on a remote virtual amazon machine. All of the data is stored on the machine and you can access our web application through this link: `34.209.10.185:8080/service/`.

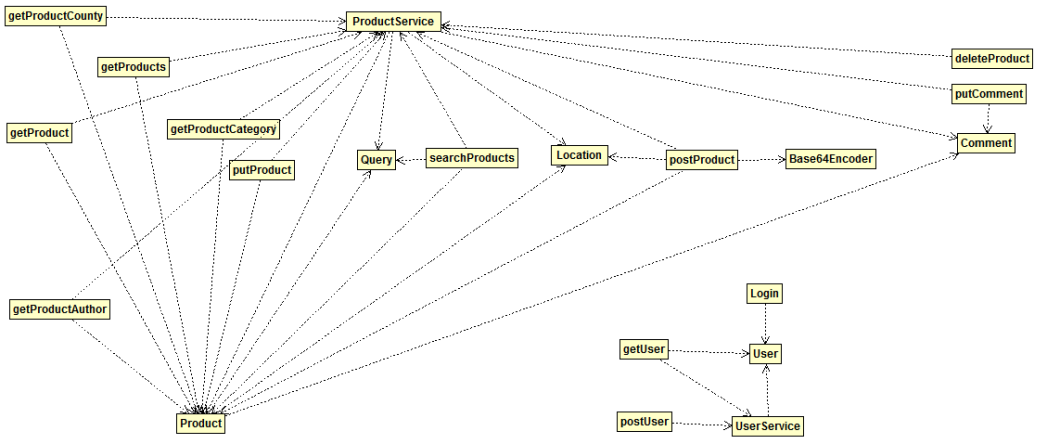


4.1.1 How it works

The user in the front end requests data or sends data to the server which is located on an AWS remote virtual machine. The server has restful URI's, which all carry out some processing on a request and execute queries against the database. The database returns the data based on the query and then the

data is sent back from the server back to the Java server, which then passes the requested data back down to the front end of the application where JavaScript parses the JSON data to display it to the user. For example when a user updates some details of a product, the JavaScript parses the input from the user for the details from the input boxes or drop-down lists. Then the data is added to a form or sent separately by requesting the update product URI and the data is sent to the server. The Java server receives the data and executes some code and adds all of the new data to a MongoDB query. Then the query is executed and the data is updated. We then reload the page and see the updates we just made to the product.

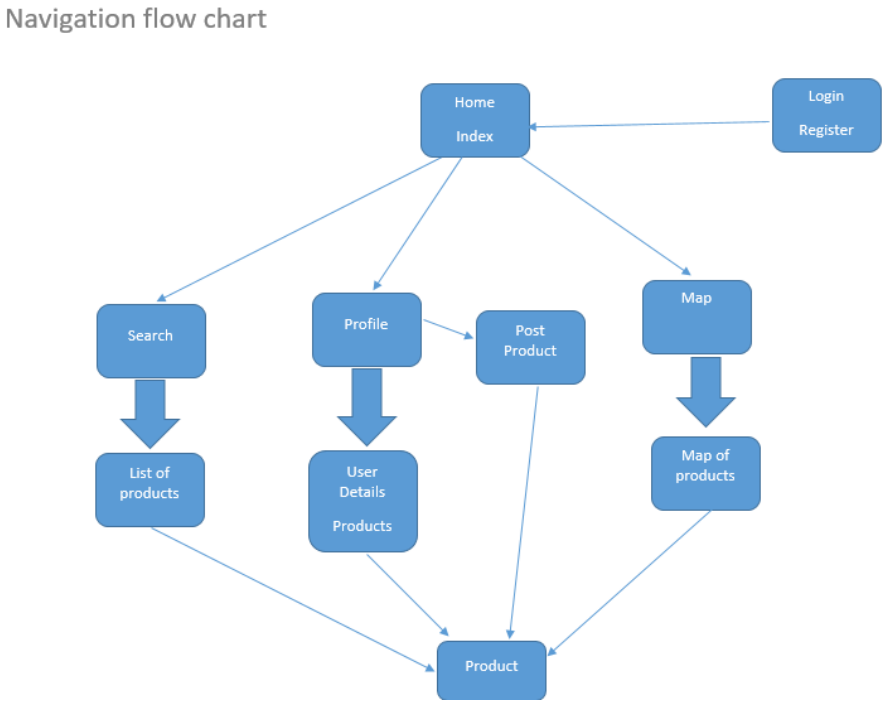
4.1.2 Class Diagram



In our server we have 2 main services, a Product service and a User service as seen above in the class diagram. The user service deals with all of the user related requests like registering, logging in and retrieving users details. User class is used to create an object oriented design of our server, which makes it easier to work with many different variables by storing them all in objects and accessing those fields with get and set methods.
The product service contains the most dependencies in the diagram. Most of the classes are using the product class which, just like the user class, stores and represents data for us in a object oriented fashion to make it easier to work with. There are many different RESTful URI's, which use classes such as getProduct, which returns a product by its Id. getProductAuthor, getProductCategory and getProductCounty classes are used to retrieve data from the database based on the author of the product, a product category or product county. deleteProduct deletes a product from the database using the product id. searchProducts class uses a query class, which create query

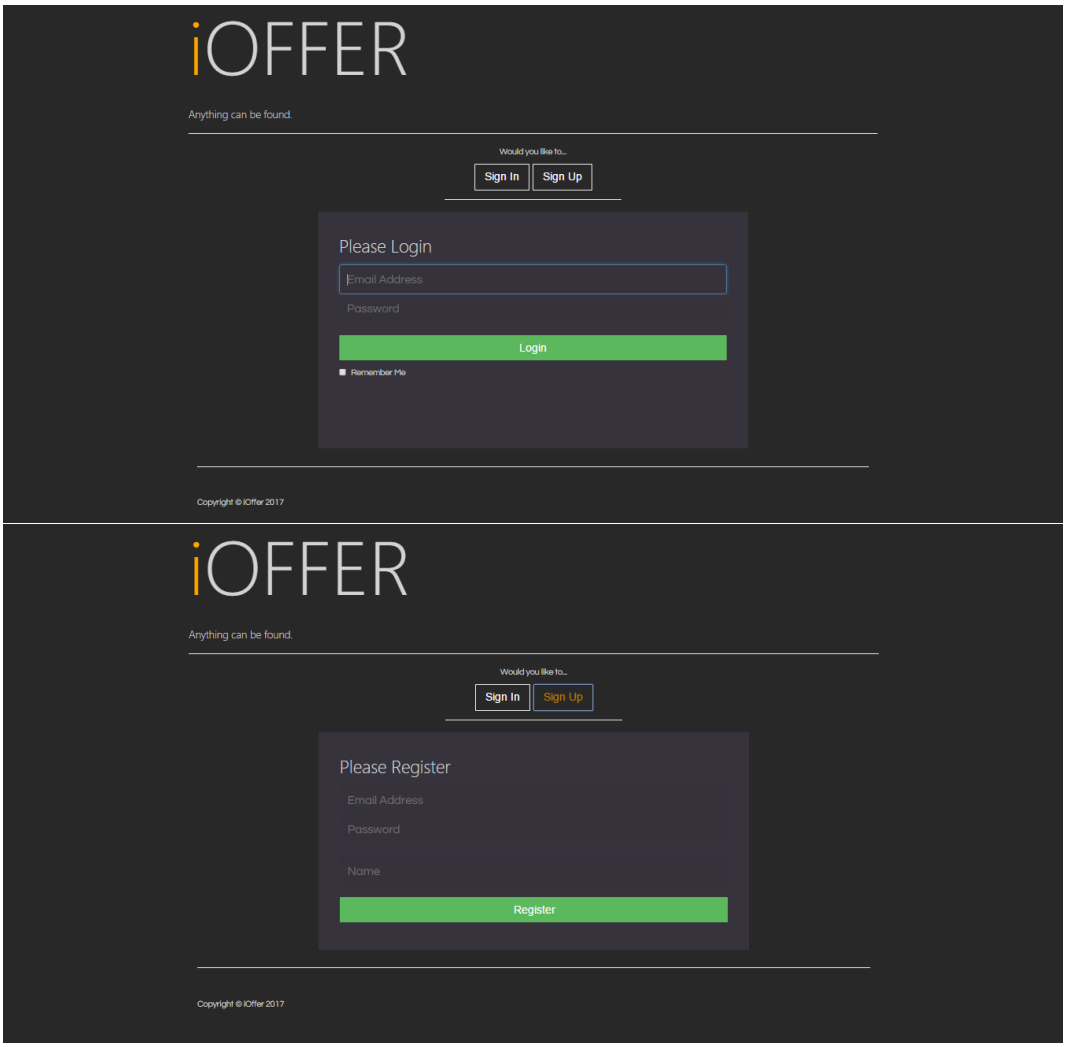
objects that contain information received from the user in the front end like name of the product, product category, product county, minimum and maximum price of a product. Then the query object is parsed on the server into a MongoDB query to execute it against the database and return the list of results or one result or no results.
The putComment and comment classes insert comments posted about a product into the database under the comment field of a product. postProduct uses the location class, which creates an location object to store longitude and latitude values, which are used to display product markers on the map page. It also uses the Base64Encoder class, which provides are the methods needed to convert images we receive from posting a product into base64 strings to them on our database.

4.1.3 Navigation in the application



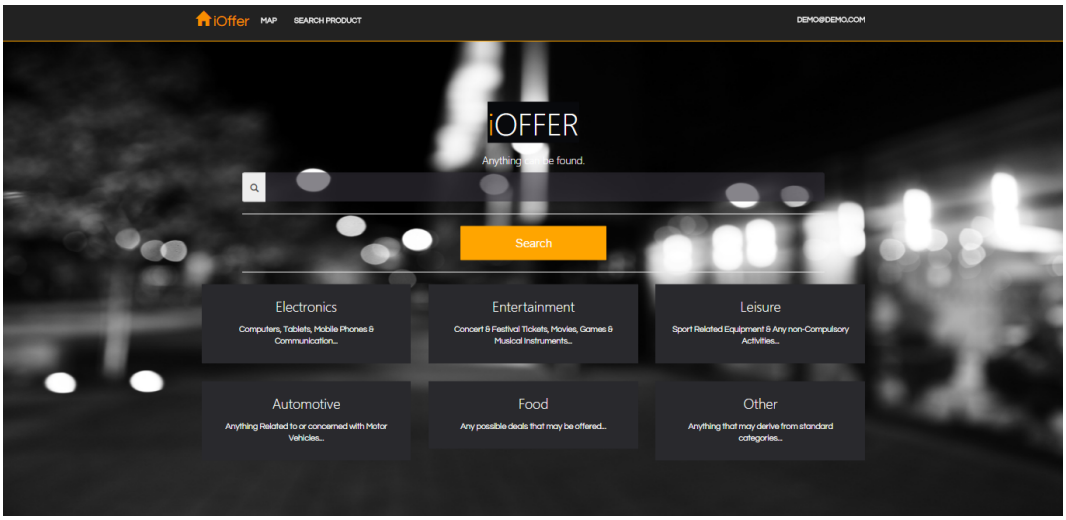
This flow chart above represents the navigation paths in our web application. The application can start on the home page. If a user tries to access pages such as the profile page or the post product page and they are not logged in, then the application will redirect them to the login & register page. We use cookies to check if a user is logged in or not.[23]

4.1.4 Login and Register pages



When navigated to the login & register page, the user can log in to the website with their email and password or use the register tab to create a new user account by providing an email address, password and their name. The password is sent up and stored as a ”SHA-256” encrypted string. The web application does not allow to register an email address more than once.

4.1.5 Home Page

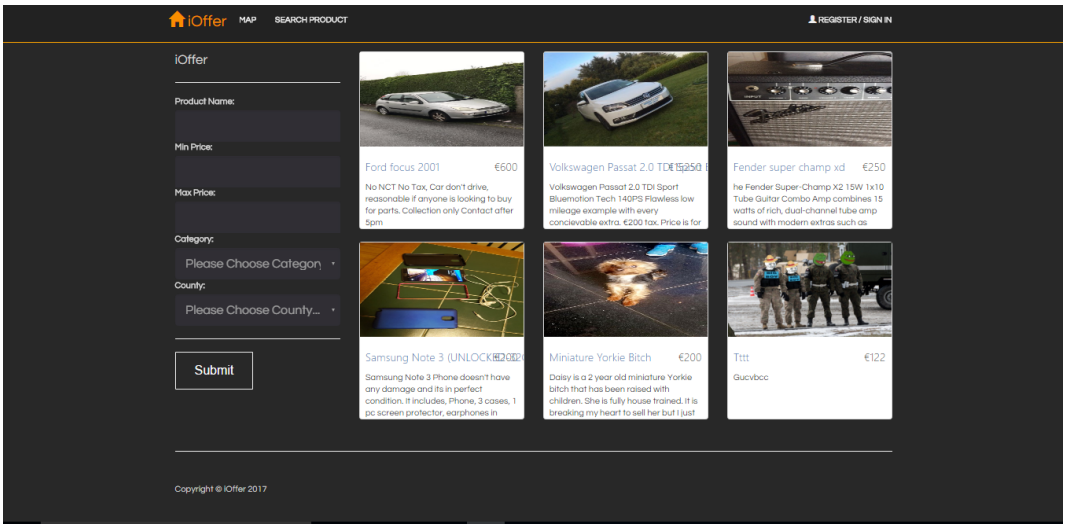


On the home page a user can navigate to a variety of pages. They can click the map link on the navigation bar to go to the map page to view all the products around their location. The user can navigate to the search page using the search product link the navigation bar. The user can click their email which is displayed on the right hand side of the navigation bar to access their profile page.

In the center of the page we have a search bar which navigates the user to the search page when they click the search button and it retrieves products from the database based on what they typed in the search bar using a regular expression to search for the products.

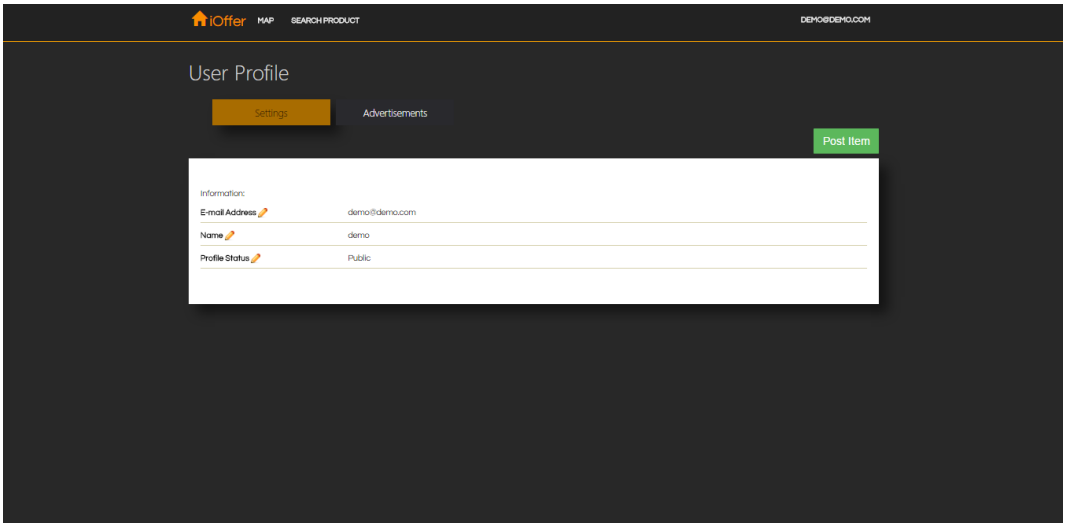
We have 6 buttons below the search bar which represent the 6 possible category choices when posting a product. Depending on which category button a user clicks, they are then navigated to the search page and presented with the products of the specific category which they clicked on in the home page.

4.1.6 Search Page



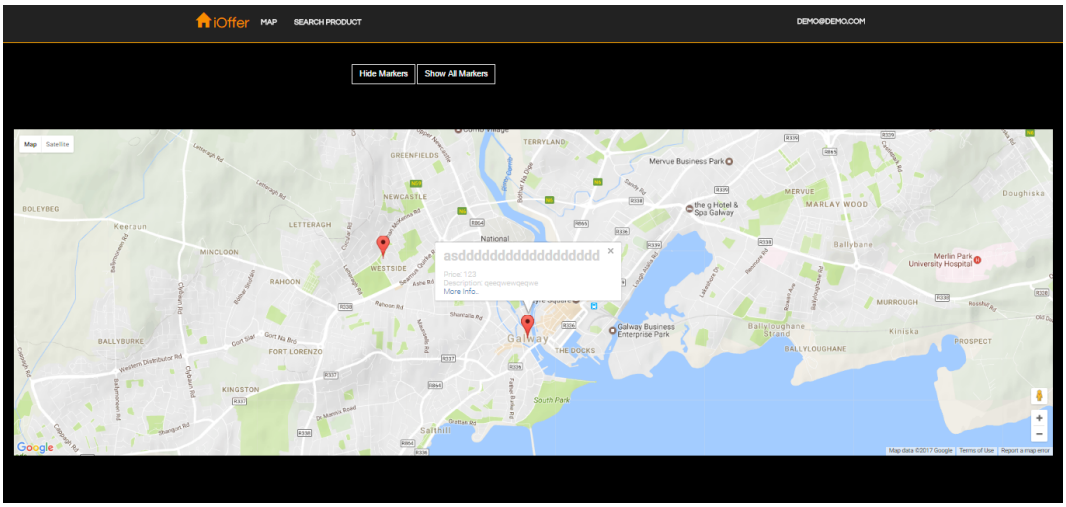
From the home page a user can navigate to the search page, which retrieves products from the database based on some search criteria such as product name, category, county minimum or maximum price ranges. The products returned are displayed in a list of cards and each product can be clicked to navigate to the individual product page. As seen in the screenshot there are 5 fields to use for search criteria on the left side of the page when searching for products.

4.1.7 Profile Page



From the home page a user can navigate to their profile page. On the profile page we have two tabs. The profile tab displays the user name and email. The advertisement tab shows the user a list of the products which they posted. The user is able to access each of their product individually by clicking on any of the products from the list. Lastly a post product button is also present on the profile page, which navigates the user to the post product page.

4.1.8 Map Page



From the home page a user can navigate to their map page. The map page uses a google map API to display a map and markers for each product. By clicking a marker a window opens up which display some information about the product and by clicking a link in the window the user can navigate to the product page. The user is also able to use the hide markers button which hides all of the markers on the map and then show all markers button to make the markers visible on the map again.

4.2 Android

4.2.1 Main Layout

The main page has functionality for filtering products by name - typing in the item name. There is also a filter by category feature which lets you select one out of the pre selected six categories. Once the filter is applied, the client is transferred to a full list of items returned from the web API.

4.2.2 Pull out navigation Layout

The pull out navigation can be used as a shortcut to any of the pages on the application. It is also available from all pages of the application. It can also be used to share the entire application to your friends.

4.2.3 Profile Layout

The account page contains all details of the account including the name of the client, email and the status of the account. Your account also has all the products posted by the client with the ability to edit and delete products posted once accessed.

4.2.4 Liked Layout

This page contains a list of items that the user liked and want to keep an eye on for later contact. This is done by saving a list of products and their id's which are then returned and placed into the custom scrollable list and displayed.

4.2.5 Create Product Layout

This page contains a form which must be filled out to post an item to iOffer. Once the product has been successfully created, the client is redirected to the product page where the client has the ability to alter the product, remove it or respond to comments.

4.2.6 Product Layout

A custom list was created in java to display items which are filtered, displaying a small preview image of the item alongside its title, county and price. It also contains each item within a scrollable list. This provides a convenient way of showing all the relevant products which have a constant limit. Each product has its own onclick event listener which redirects to the specific item page which contains a full description of the product along with all of its images and the price. The specific item page also contains a call button which opens up the clients dialer with the phone number inserted so the user can choose to call or message the seller. Each product has its own comment section where any logged in user can comment and make offers on the product to agree on a price and collection. There is also a share button for sharing the specific product to the clients Facebook wall.

4.2.7 Map Layout

This page contains a full size map which has all the products with markers containing each product with the title of the product, the price of the product and a redirect link to the full specific product page once the redirect link is selected.

4.2.8 Register Layout

A user can create a profile to post products on the site and comment on products on iOffer. This layout contains a textblock for the users full name, email address and a new password.

4.3 Database

4.3.1 Images with MongoDB

We decided to use images that are stored on our MongoDB database. When a user is creating a product to advertise on our website, the user is required to upload images, so other users browsing products can see them. This is necessary for our website, as not using images would make the website not viable for use.

A common library is available when using MongoDB. The library GridFS, allows to store various files on MongoDB. GridFS is most commonly used when dealing with large file sizes, storing and retrieving files that exceed the BSON document size, which is over 16 megabytes. GridFS does not store the whole file into one document, instead it divides the file into smaller chunks. The usual chunk size is 255 kilobytes. All of the file chunks are stored as separate documents on the database itself. One main document is stored which contains a pointer to all of the other chunks of the particular file.

Instead we decided to use something we have done in class one time, which is the Base64 encoding format.

We wanted to use and put what we learned into practice. We select images in product creation using a file picker which filters file input only to .jpg, .jpeg and .png files. The files selected are sent as part of a form using a Restful POST URI. On the server side the form is then parsed into its components and they are processed to be stored on the database. Images are converted to Base64 strings using a Base64 encoder class. The Base64

strings once converted, are concatenated to an image string and each individual image string is separated by a comma. The final result string is then stored in the database.

To retrieve the images, we carry out the opposite of what we do to store them. We use a Restful URI to retrieve the data. The data is passed down together with the Base64 image string to javascript on the front end. JavaScript then parses the data and it splits the image string into the separate base64 strings. It then decodes the strings back to image format for display on the web site.

4.3.2 Data Model

We only need to have 2 collections in our database, which include:

- User collection for log in details. Eg. UserName/Password
- Offer/Advert collection which contains users posts

Example document of a product stored in the database:

```
{
        "_id":123123123,
        "name":"Samsung Galaxy S5",
        "price":"400",
        "description":"mobile phone",
        "images":"Base64 string",
        "location" : "53.27, -9.05",
        "county" : "Galway",
        "author" : "someone@gmail.com",
        "category" : "electronics",
        "mobileNo" : "085123123",
        "comments" : "Json array of comments containing: author, date, comment",
}
```

4.3.3 Data

The product location is retrieved by using a geo locator in javascript when creating the product advertisement. These geo coordinates are used to place markers on the map display of our products.

Each product contains internal _id used to view and query individual products.

Comments array starts out as an empty json array. When other users who are viewing products post comments, the array gets updated by inserting the comment into it, which contains: Author of the comment, date and time of the comment, and the comment itself.

4.3.4 Login

For Login functionality the user will be asked to provide user information such as:

- E-Mail
- Password (SHA-256)

4.3.5 Registration

For Registration the user will be asked to provide:

- E-Mail - which will be used for logging in
- Password (SHA-256 Encrypted)
- Full Name
- Phone Number - Make sure that the seller will be accessible if no other information is provided.

4.3.6 Advertisement

For posting advertisements, these are the following requirements to post advertisements on our website:

- Advertisement Image (Optional)
- Advertisement Name
- Advertisement Description
- Phone Number (Provided from Registration)
- Seller ID/Name

- Position (latitude/longitude) - Will be used to post on map. The Seller will have a choice to use their location or by placing a marker on the map.
- Advertisement Category
- Advertisement Price

The server provides and supports operations on products such as create product, update product and delete product. Each of these are used when accessing different Restful URI's.

The server also provides detailed searching functionality to search for products on the database. Searching is available with product name, category, county and minimum & maximum price ranges. Searching can be carried out using any of the fields or all of them at the same time. Products are returned, which match the search criteria.

Chapter 5

System Evaluation

5.1 Evaluation

From the initial planning of this project, we can say that we have met all of the initial objectives we have set out for this project. In some cases we have added more to the project functionality and improved the overall system.

We have met all of our set out requirements of the web application and android application by steadily ticking off all of our set out objectives. We incorporated all of the functionality that we set out to implement and towards the end of the development we have added additional things that were not on the initial specification of our project.

The application allows users to have full login registration functionality, full functionality for posting item to advertise for sale, the google map api implementation to display products for sale and all of the related searching for product on the web application. Throughout the development process we eventually made our web server restful.

From the start we did not plan of doing this but this is a huge improvement for our overall system functionality as it made connecting our native android application to the server a lot easier. We also implemented a facebook api, which allows us to share products on our web application to facebook. Lastly we added comments to all of our products which allows other users to comment on the product which are listed on our web application. This allows for a more real world feeling application and adds more to our web applications functionality.

We as a team are happy that we have met all of the set out requirements for our project. Other set out objectives for this project were for us all to learn and improve our coding skills and time management skills. We have improved our coding abilities during the course of the development of the project in the specific areas which are all of the technologies we used in this project such as java, JavaScript, HTML, Android and MongoDB.

We have also gained a lot of experience in time management, meeting goals and deadlines and managing software on Github. As we followed a weekly development plan for our project we made sure that we stay on track and meet all of our weekly plans. We also gained some additional knowledge with using software managing tools such as Github.

5.2 Limitations & Opportunities

Based on our choice of technologies and programming paradigms we had a few limitations and opportunities.

Because of how we managed our development process, working together during development sessions, it allowed us to predict what problems could arise. This is because we had the developers for every part of the application and the full scope of the application was being worked on all the time.

Besause we encrypt the images for an item, we did not have to store them on our server and could afford to get a virtual machine with less memory. This saved us costs. Because we created a restful web api, our application promotes scalability and allows others to query our database using the restful API.

Our lack of previous experience working as a group on Github, made it difficult to use. This was because we were working on the same branch and had to merge code if one team member was working on a page and another edited it at the same time and committed the changes, the first member would have to merge his code with the Github version of the code. Initially we began our project on a netbeans/glassfish environment, however a few weeks into development we were posed with problems. We could not find satisfactory documentation for glassfish and decided to change our development environment to eclipse and tomcat. This proposed an opportunity which allowed us to get more done as less time was spent looking through glassfish documentation.

Chapter 6

Conclusion

Overall, the main objectives of the projects have been successfully implemented without major problems.

Main goals of the project included many different functionalities that were implemented over time as said previously in the System Evaluation section. First of all the web version of iOffer was planned for development and then it was decided that Android application should be implemented to increase the scope of the project.

Both Web version and Android application would have same functionalities but wih slightly different implementation and design, each being tailored for the user needs. Goals were not solely limited to this but main goal was to gain a deeper knowledge of how to implement system like this, learn new technologies and how to work with them efficiently.

During the development it was discovered that, as previously mentioned above the hands-on experience with multiple technologies has improved our proficiency using Java, HTML, CSS and JavaScript.

6.0.1 Java

Working on this project resulted in each member gaining a better, in-depth knowledge of working with Java as main language that has been used for server implementation, a body working between the front end and the database. Before the beginning of this project each member was familiar with the language and proficient at that but over time and through development many things were encountered that we have never worked with before and this only increased the confidence of each member.

6.0.2 Android

During the four years of education at GMIT no module required the use of Android development in native format, there was plenty of mobile development but nothing with native Android which was one of reasons for choosing this platform for development. From little to no interaction with android development it was great to learn how it works and the unlimited possibilities related with it. Leaving each member confident in development using this platform.

6.0.3 Various API's

Described in previous chapter was the Application Programming Interfaces (API's) which were used during the development and without any of them much functionality would be lost or it would lead to much more difficult implementation with possibility of multiple bugs being there hidden resulting in much less friendly environment for user and the developer.
Overall, it was great to get a chance and work with those API's, their implementation in the project resulted in huge boost of time required to develop both Website and Android application.

6.0.4 Database

Working with MongoDB was great choice, each member stated their positive view on choosing this as the database being used. Documentation was very impressive and full of information and the general structure of MongoDB was easy to follow and interact with. The capability of document to represent complex object's data structure using JSON syntax was warmly appreciated.

6.0.5 Maven

Working using Maven proved to be useful and easy, it simply let's the developer get package dependencies more easily and forces to have standard directory structure making the whole project much more view friendly. No complicated build.xml files, just Project Object Model which is XML file containing information and project and its configuration details used by Maven to build the project. Other benefit was the centralized POM resulting in all developers in project using same jar dependencies.
Overall Maven cut down on difficulties with dependencies and was great to work with.

6.0.6 Tomcat

Described in previous chapter that Tomcat was used for the deployment of Java Servlets and JSP's it's no surprise to why it was chosen. The easily operable, rich in documentation, online resources and previous experience with it forced for the selection of this web server.

6.0.7 Amazon

Easy to set-up, user friendly interface, unlimited possibilities and the student discount was the key to using it. Minimal problems encountered during the process of using it made it just the perfect webservice to choose from.

6.1 Recommendations for future investigations

If both, the Website and Android application were to be further developed in the future these could be the potential recommendations. Firstly, the time required for the development of this project to its full potential. It was very time consuming for the size of our group and the scope of the project was too great and many functions were not implemented due to time limitations.

It would be worth mentioning that the size of the group and the separation of work would greatly improve the end product and its functionality. More team members would allow for easier division of work i.e One member could work on Login and Registration functionality only while other could work on Posting advertisement and getting both polished to perfection rather than dividing the group into Front End, Server, Database and Android segments and each member jumping from one to other.

Overall, working on this project was a great experience to each member of the group, it gave each of us a greater feel of how it feels to work in real life situation where the developers are responsible for their work and how it may affect other group members if only one member would slow down or not work as hard as others. Concluding that a healthy team is very important and it is great to work with different people with different views on how certain things should or could be done.

Chapter 7

Appendices

7.1 GitHub

The GitHub link to the project repository can be found below:
`https://github.com/PawelBor/Year-4-Offer`

Bibliography

[1] "Amazon web services (aws) - cloud computing services." https://aws.amazon.com/.

[2] "Google maps apis - google developers." https://developers.google.com/maps/.

[3] "The agile movement." http://agilemethodology.org/.

[4] "Representational state transfer - wikipedia." https://en.wikipedia.org/wiki/Representational_state_transfer.

[5] "Github." https://github.com/.

[6] "Acid - wikipedia." https://en.wikipedia.org/wiki/ACID.

[7] "Android." https://www.android.com/.

[8] "Home – arm dynamiq technology." https://www.arm.com/.

[9] "W3c html." https://www.w3.org/html/.

[10] "Cascading style sheets." https://www.w3.org/Style/CSS/Overview.en.html.

[11] "Javascript web apis - w3c." https://www.w3.org/standards/webdesign/script.

[12] "Bootstrap · the world's most popular mobile-first and responsive front-end framework.." http://getbootstrap.com/.

[13] "Facebook for developers." https://developers.facebook.com/.

[14] "jquery." https://jquery.com/.

[15] "Jersey." https://jersey.java.net/.

[16] "Java platform, enterprise edition (java ee) oracle technology network oracle." http://www.oracle.com/technetwork/java/javaee/overview/index.html.

[17] "Maven – welcome to apache maven." https://maven.apache.org/.

[18] "Apache tomcat® - welcome!." http://tomcat.apache.org/.

[19] "Json." http://www.json.org/.

[20] "Mongodb for giant ideas — mongodb." https://www.mongodb.com/.

[21] "Nosql - wikipedia." https://en.wikipedia.org/wiki/NoSQL.

[22] "Apache couchdb." http://couchdb.apache.org/.

[23] "Http cookie - wikipedia." https://en.wikipedia.org/wiki/HTTP_cookie.