Introduction

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

For the Web application:

* HTML, CSS + JavaScript
* Bootstrap
* JSP
* J2EE
* Tomcat web server
* MongoDB
* Java, JAX RS, Jersey
* JSON
* Google Api's for the map
* Amazon web services
* Facebook Developer API

For the Android:

* Java
* XML
* Tomcat web server
* MongoDB
* JAX RS, Jersey
* JSON
* Google Api’s for the map
* Amazon web services

Our web application and the android native application are going to be the same application. We originally planned to do only the web application. 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 local host we will be deploying our web application to a remote amazon virtual machine. 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 will be deployed to the remote server is because we need the server functionally and database to be accessible by our native android application.

We will also be using a google API for google maps. The map will be 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 display 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 people that willing to sell their items.  
The user will be able to post an item for sale, which will fall under desired category eg. 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 registration form and further log in to the system. Then the seller can post an advertisement or item for sale by providing the necessary details like name of product, price, location, phone number…  
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 individual product advertisements. 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 will be able to browse items for sale in the applications by using search criteria or view all of the products for sale around their location on the map display.

Objectives of this project:

When talking about the objectives or learning outcomes of this project, a few things come to mind.

Firstly probably the main objective is to learn and broaden our working knowledge of the technologies that will be used in this project. This project covers quite a variety of front end and back end technologies and also a database. Some of the technologies in this project we have never used like MongoDB, which will be a new working experience for us all and other technologies we have covered briefly or used quite often but it will still increase our working knowledge of them. Working with all of these technologies and connecting the front end with the back end java server will further sharpen our programming skills and techniques. We will learn more about and how to make our web application restful, which will make it easier for our native android application to access the data that it needs for functioning. Using a Tomcat server and deploying our web applications to a remote virtual machine will also give us more experience working with client and server systems.

Since this project is to be development during a course of 4 to 5 months, we will be gaining quite a lot of experience with building up our projects from scratch to the finished result. A lot of the smaller project we have been working on are smaller in scope and do not take such a great amount of time to develop. Usually these type of projects involve less of the building it up part.

This is a great opportunity to further learn how to continually build up and work on a project over a longer period of time. This hand in hand gives us more experience to become better at time management and goal setting for the project.

We will manage our time by setting up sessions to work together on the project and manage our goals and deadlines by setting out a weekly development schedule of the project, so we have an idea where we are when developing compared to the original development plan for this project.

Our main metric by which we will measure success or failure of this project will have to be based our time management and goal meeting. We will use our weekly plan to stay on track during the development of this project.

Github:

Github link to the project repository: <https://github.com/PawelBor/Year-4-Offer>

In our github repository we have a readme and a basic license file. The repository also contains 3 folders: Android, java, webapp.

The android folder contains all of the back end and front end related code.

Java folder contains 2 folder 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 URI’s for the restful service such as getProduct, which when accessed on its URI webapi/service/product/{id} will return a json representation of the specific product found by the {id} part of the URI from the database if it exists.

Our other folder service contains java classes which are like supporting and utility classes. We have classes like Base64 encoder, which is used to encode our images to Base64 string. We have classes like Comment, which are just used to make are server more object oriented.

Webapi folder contains everything to do with the front end of the web application. It contains things like our JSP pages that are served and images and CSS style files and Bootstrap for the front end of our project.

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.

Sections in this document:

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

Methodology:

This section of the document is describing how we went about developing our project. We talk about our approach to developing the web application and native android application and how we carried out our research before commencing the development of the projects. We cover in this section as well how we went about meeting our supervisor, how we carried out development sessions together, how we managed our github repository and approach we used for the development of our project. We also cover why we choose the languages, technologies and platforms for our projects. Lastly we talk about how we dealt and solved problems encountered during the development of the project.

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. What kind of Standards we used or what kind of Database Model was the system implemented using. We also talk briefly how to set up our individual technologies or how we set them set and what we used to interact with them. We discuss our technologies under 4 headings: Front end, Back end, database and Android.

System Design:

This section of the dissertation provides a detailed explanation and pictures of our overall system. It describes each individual part of our system. Front end, Back end, database and Android are all of the individual parts of our overall system. Diagrams will aid our explanation for how the system works and interacts 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 amazon virtual machine and how that affects our system.

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 of this project. 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 our projects.

Conclusion:

In this section of the dissertation we will cover a summary of the context of the project and all of the set out objectives. The outcome goal for us as a group was to learn how to work with multiple technologies and how to get them to work together and to further our knowledge. We will talk about the outcomes of the project, any tangential or even any unrelated insights that we discovered throughout the development process which thought us a lot.

References:

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

Methodology

Software Methodology

Our approach to the project was mostly of agile methodology. At the start of the project we set out our whole project scope. After it was decided, we divided 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 Gedimininas was in charge of the front end of the android application and connecting it up to the restful service and database.

Research Methodology

For our research before commencing the development stage of our 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 map 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 our project.

Planning

In the initial weeks of the project, we spent a week as a group discussing potential ideas for our project. We came up with a few ideas for our project and eventually we 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 this project and our supervisor approved of our choice for the project. Next week we discussed the overall scope of the project and we decided that we need to add a native android application as well to increase the project scope and to cover more technologies.

Meetings

As a group we set up a 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 that Niks and Ed begun working together on the server and database. Since MongoDB does not have any schema or tables and is really easy and fast to set up, after setting it up Ed moved to the server development with Niks. While Niks worked on things like handling 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 a conclusion that the java web servlets will not do for us. We realized that our java web servlets would be much harder to connect with our web native android application and it makes it more secure. It is more secure using the restful api as we do not directly connect to our database. After doing some research we figured out, that our best option was if our web server was to be restful. Then we proceeded with this and begun re coding our java server to now use restful URI’s instead of the java servlets. At the start we had a few problems when trying to get our server to work as a restful api. Eventually Gediminas got some restful api 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.

Structure

The structure of our development sessions was that at 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 receive on our progress.

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.

Testing

For testing to make sure everything is working as it should 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.

Problems

During the development of the project we have encountered only a small number of significant problems and some minor bugs. Our significant problems were passing up images from the front end file selector using a restful URI, re writing our server to use restful URI’s rather than java servlets and re packaging our project as a maven project to allow maven to handle all of our dependencies for the server.

When we encountered our problems, to deal with them we identified the cause of the issue and became clear of what the exact problem is. We researched and figured out our solutions to solve the problem. After discussing our possible solutions, we chose the best option for us to solve our 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 to whose area of responsibility is related to the issue. Once the problem has been resolved, if it had an outstanding issue on Github, we then closed the issue as resolved.

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 choose to use Github. We have been using Github for the last few college projects as they require to. Using version control tools encourages the development approach of building up projects over time by committing smaller pieces of code frequently rather than committing huge pieces of code. Github allows users to see a commit history. This allows to go back to previous versions of the project if something goes wrong in the development process. Github also has an issue section for a repository, which allows users to create issues for other team member to see that need to be fixed. 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 read me file. In read me file we have information about or project, team members, supervisor and information we were incrementally writing about different parts of the project. We have documented briefly our main technologies that are used in our web application and native android application.

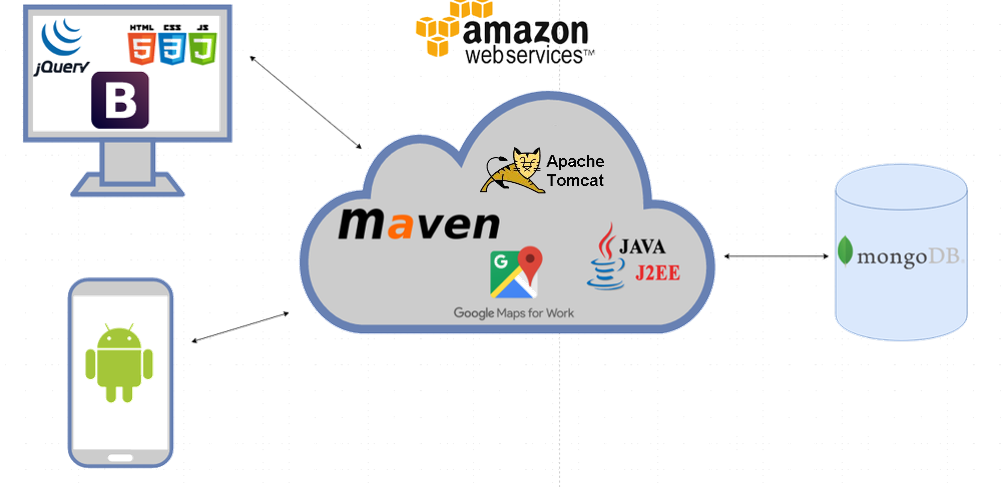
Selecting Languages & technologies

To select 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 EE environment. We have not written client server application using java beforehand. We chose java to learn about developing server side and also improve our skills using the java programming language. For the front end we chose to go with html, css, javascript and bootstrap, mostly to improve our front end development skills. We chose to use MongoDB as we will not need to have any ACID properties in our website and database. Our applications do not involve any transactions and have quite a variety of data, therefore a non sql document database was a good choice for us. We also have never used MongoDB prior to this, so this was a learning opportunity of 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 Amazon web services 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.

System Design

Architecture

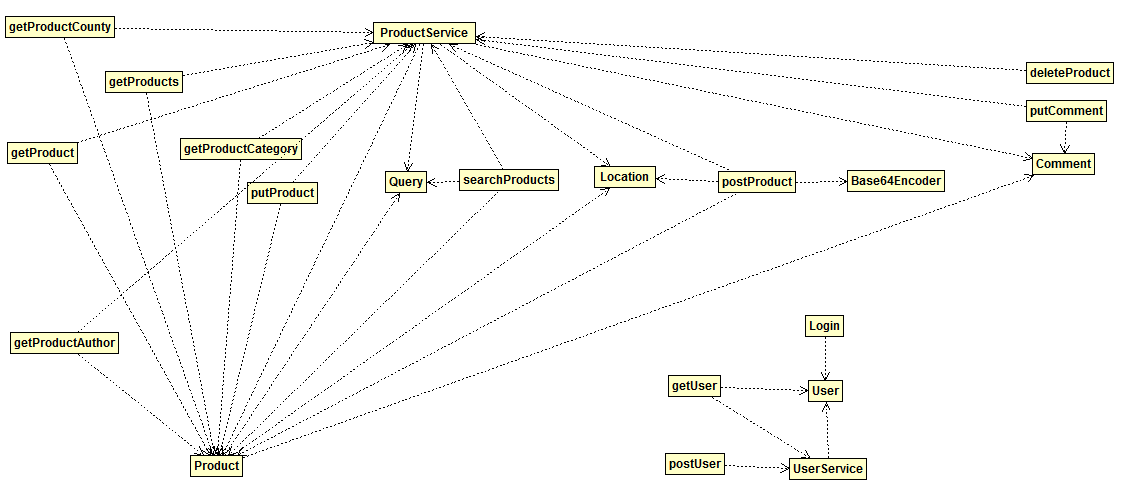
This project is based on the client & server model. We set up a remote server for our project by hosting our java server and database on a remote virtual amazon machine. All of the data is stored on the machine and you can access our web application through this link: 54.244.



How it works

The user in the front end requests data or sends data to the server which is located on an amazon 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 a new the new details from the input boxes or dropdown 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.

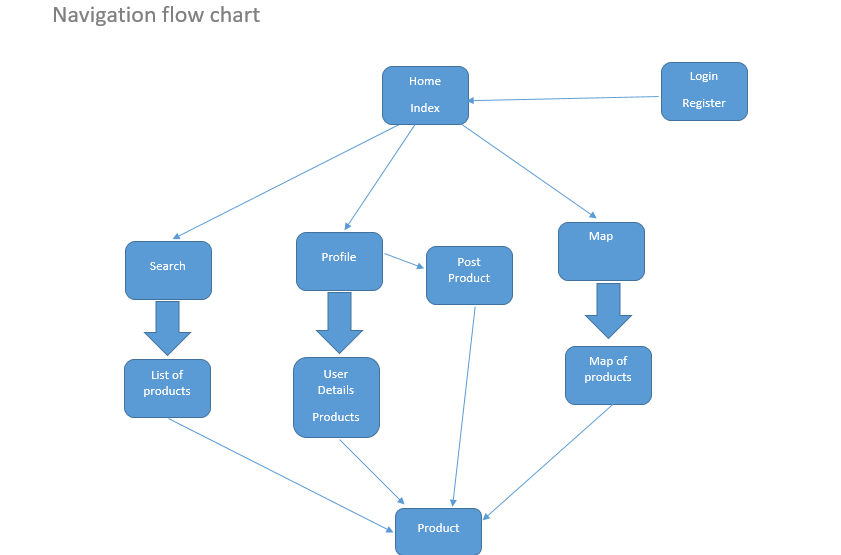
Class Diagram



In our server we have 2 main services, Product service and User service as seen above in the class diagram. User services deals with all of the user related requests like registering, login 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.

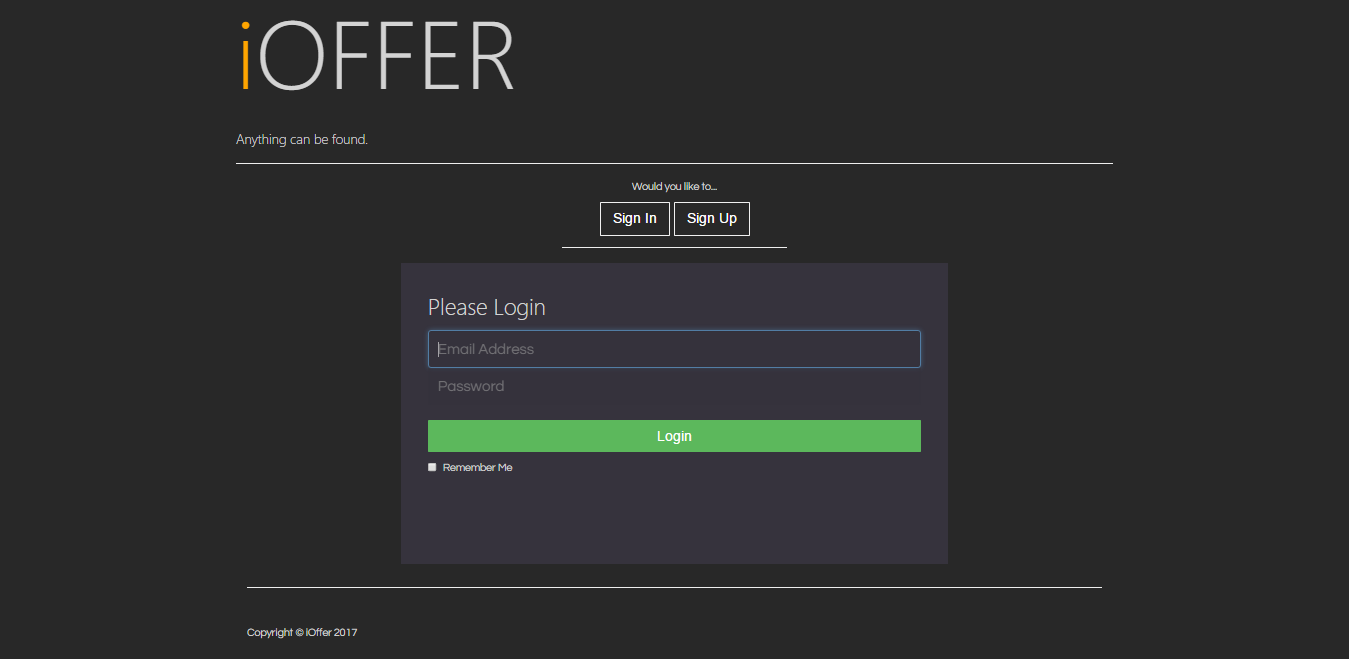
Product service contains the most dependencies in the diagram. Most of the classes are using the product class, which just like user stores and represents data for us in a object oriented fashion to make it easier to work with. There are many different restful URI’s, which use classes such as getProduct, 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.

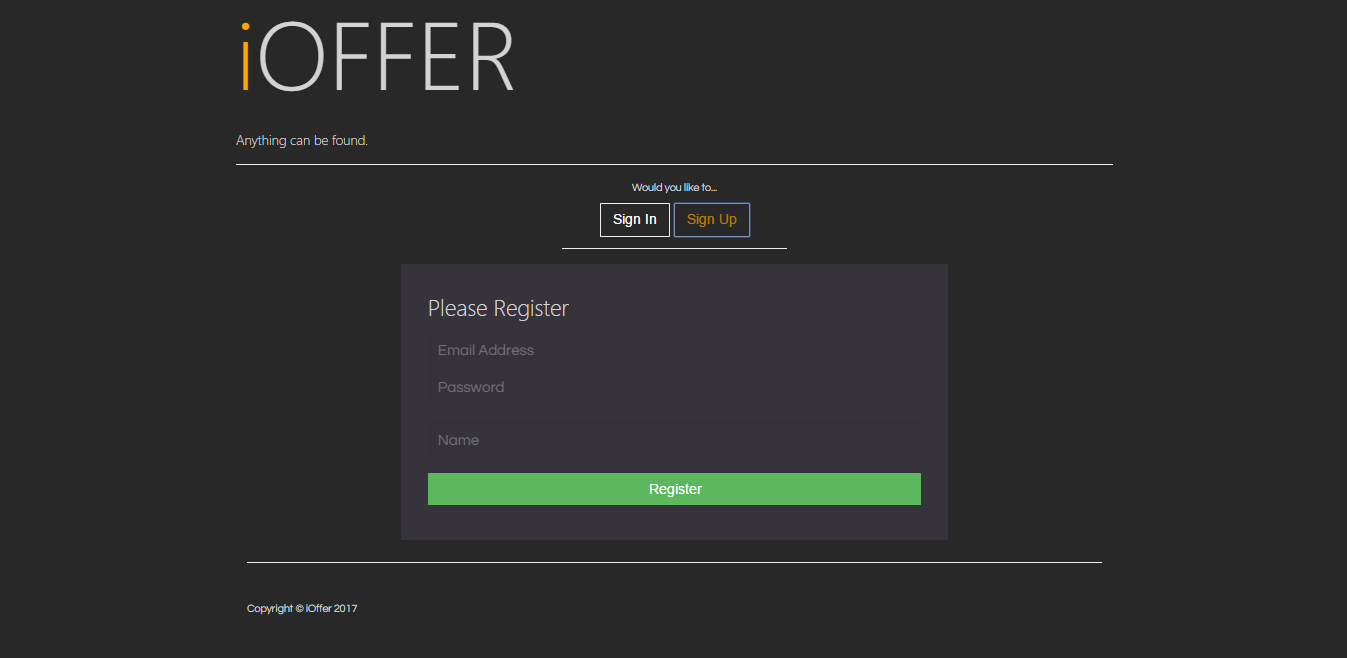
Navigation in the application



This flow chart above represents the navigation paths in our web application. The application can starts on the home page. If a user tries to access pages such as Profile or post product 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.

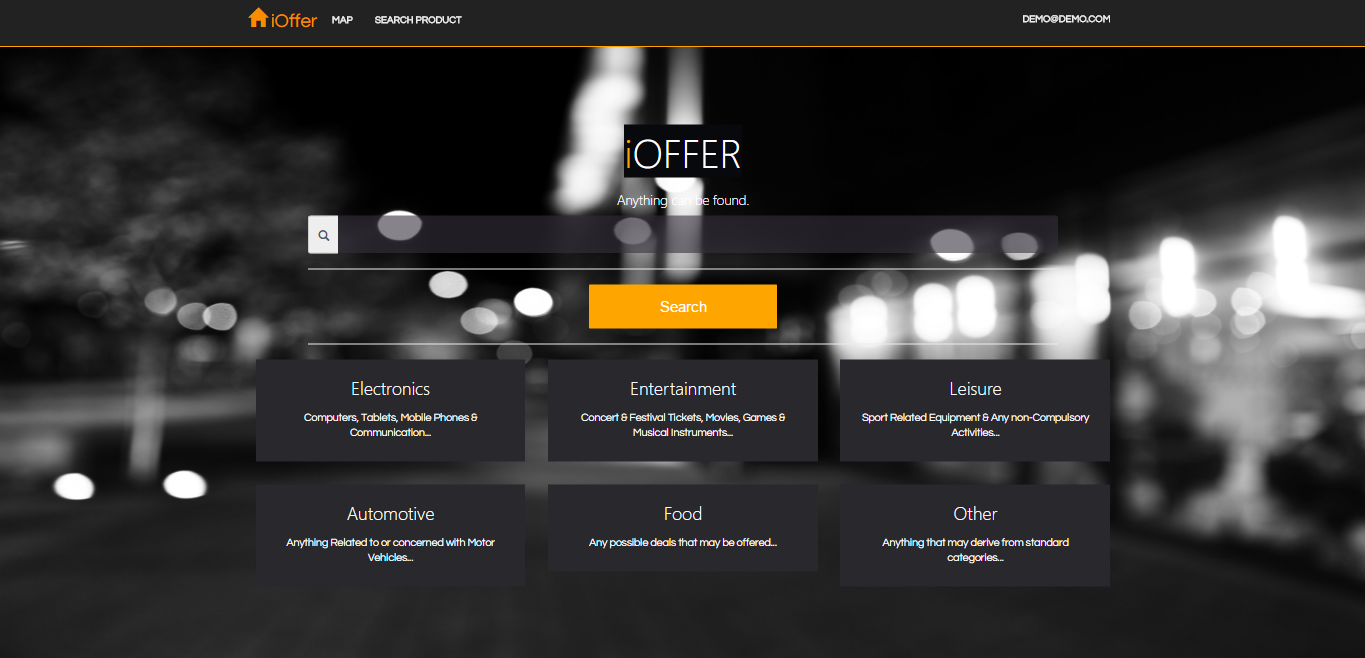
Login and Register pages





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 web application does not allow to register more than once with a given email.

Home page

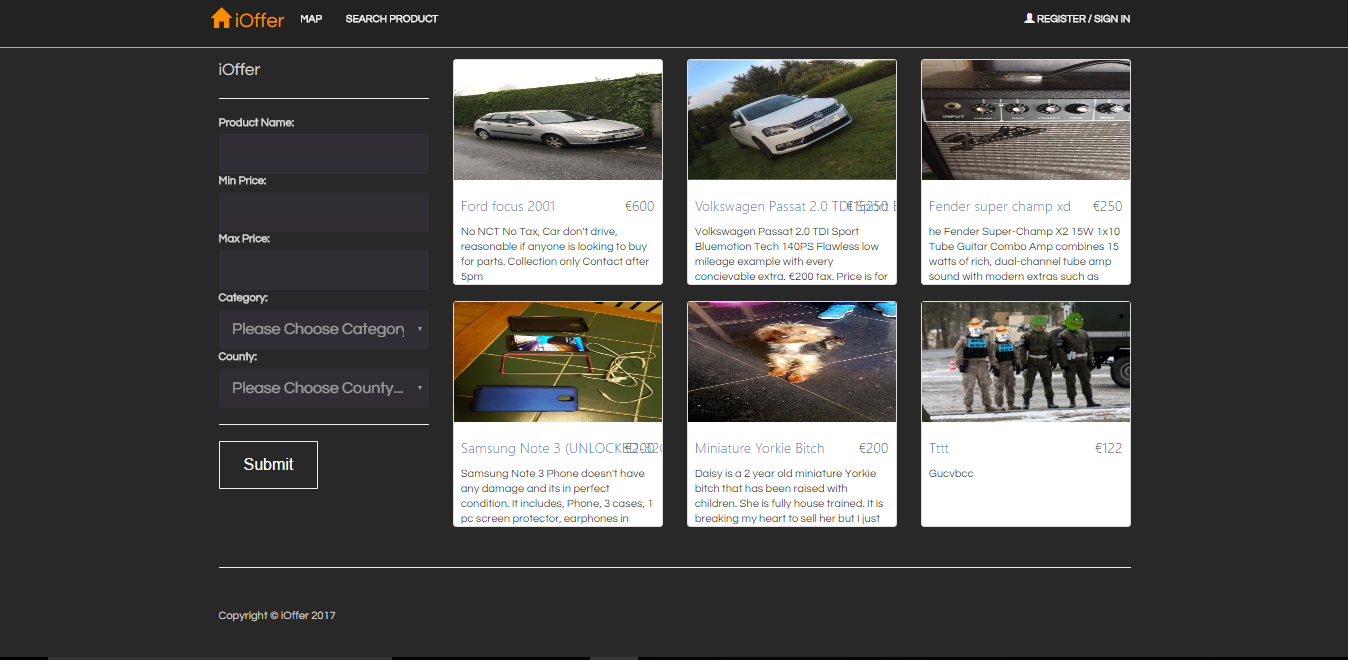


On the home page a user can navigate a variety of pages. They can click the map link on the navigation bar to go to the map page to view all the products around their location. The user can navigate to the search page using the search product link the navigation bar. The user can click their email which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

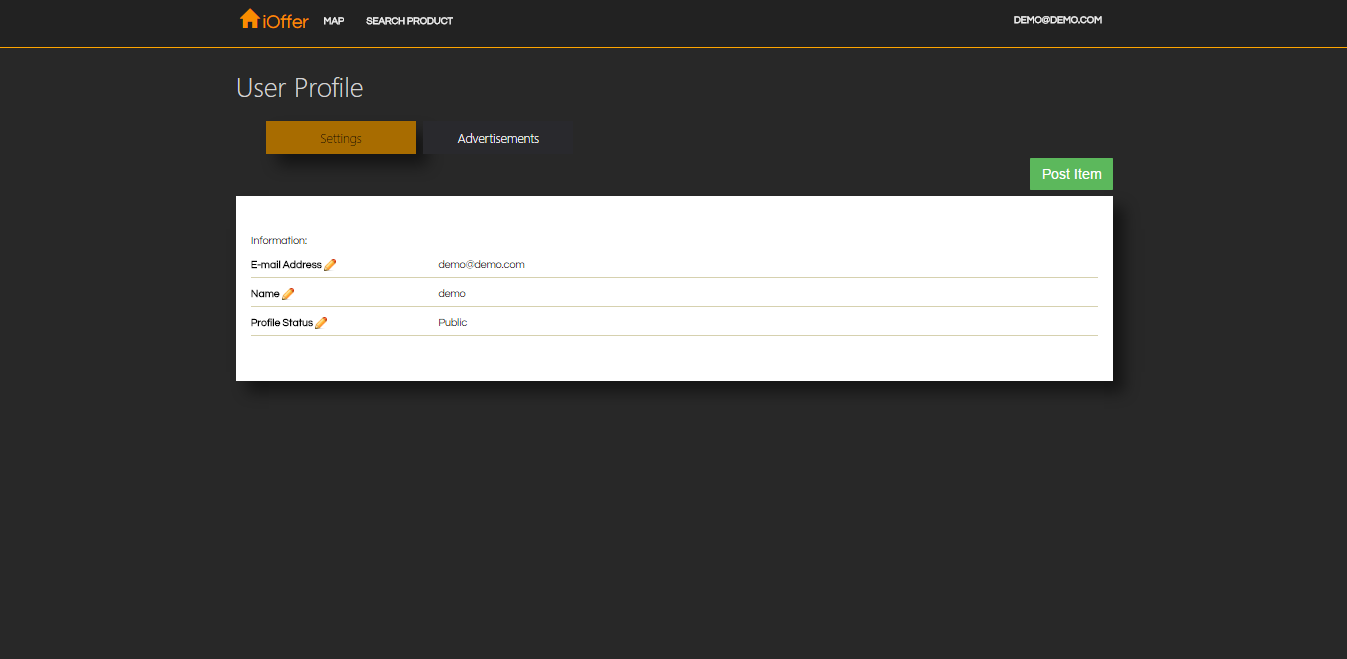
We have 6 buttons below the search bar which represent the 6 possible category choices when posting a product. Depending on which category button a user clicks, they are then navigated to the search page and presented with the products of the specific category which they clicked on in the home page.

Search Page



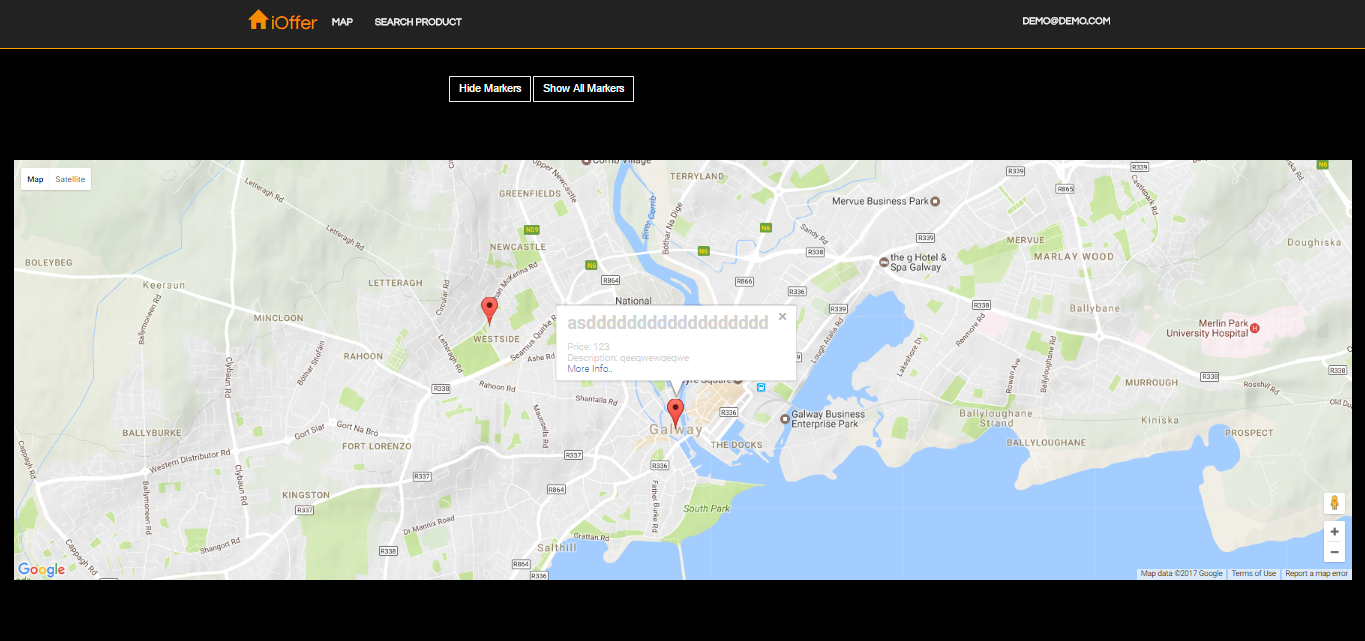
From the home page a user can navigate to the search page, which retrieves products from the database based on some search criteria such as product name, category, county minimum or maximum price ranges. The products returned are displayed in a list of cards and each product can be clicked to navigate 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.

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.

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.

Technology Review

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.

Tomcat

Tomcat is an open source java servlet container. It allows to run web applications on a localhost server to access these applications though an 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.

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.

Database

MongoDB

Prerequisites

To get MongoDB up and running, 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 eg. Intellij to interact with MongoDB. Intellij has a plug in called mongo explorer, which allows to open 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 and similar to JSON-like documents to store data. Documents are stored in collections and the documents 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 scalling the data.

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. Prior 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.

MongoDB also seemed to be easier to use and learn, hence why we chose to go with MongoDB rather than CouchDB.

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 without 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 an image string and each individual image string is separated by a comma. The final result string is hen 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.

Collections within the database

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

* User collection for log in details. Eg. UserName/Password
* Offer/Ad 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",

}

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.

Login

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

* E-Mail
* Password (SHA-256)

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.

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.