University of Glasgow | School of Computing Science

Level 3 Project Case Study Dissertation

AMIS - A Marketplace for Industrial Symbiosis

Waqar-un-nisa Nabi
Jake Aldred
Luke Doleman
Alex Leet
Anton Antonov Petrov
Mariusz Szmajduch

10 April 2016

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format.

1 Introduction

This document is the dissertation of Team C, a team consisting of 6 third year Computer Science students at The University of Glasgow. We worked with the Crichton Carbon Center to create a web application which would be an online marketplace to buy and sell industrial waste.

Our dissertation will explain the project we worked on in greater detail. It will also explain some of the greater challenges we faced at different points of the agile software development process, how we overcame them and what we learned from those experiences.

The structure of the dissertation is delineated as follows.

Section 2 presents the background of the case study. It also describes the customer and project context. It outlines the original core requirements we discussed with our customer from our first meeting and which of these we achieved.

Section 3 details out team dynamics and which roles we felt were most needed in the group and how over time these roles developed. It details the challenges we faced with the Agile development process and what we learned from it. The testing stage is also explained here as the challenges we faced with it were part of the software development process.

Section 4 outlines the meetings that took place with potential users who were working in companies that are part of the manufacturing industry. We found this to be very valuable to our project as that is where we got most of our information on how companies currently trade waste, as well as what features they would want to see in a website that could help them expand their waste trading.

Section 5 details the process of front end development from creating the wireframes to the many changes that took place before the final product was delivered. It also explained some of the integration and communication challenges we faced as a team when connecting the front and back end together.

Section 6 explains the process of back end development. We began here from our first team meeting and much of the site has changed from the original plan, this process is explained in greater detail here. The greatest challenges we had were trying to implement some of the features and these are explained here as well as other challenges we faced and what we learned from these experiences.

2 Case Study Background

2.1 Project customer

Crichton Carbon Center are an environmental not for profit organisation. They are based in Dumfries and since 2007 have been working to create a low carbon society working with both academic and applied programs. [?] The CEO Mike Bonaventura and researcher Liz Brooks were our customers for the project.

2.2 Project

The project was to create a marketplace for industrial symbiosis (AMIS). The idea behind the project was to enable the creation of a circular economy.

"A circular economy is one that is restorative and regenerative by design, and which aims to keep products, components and materials at their highest utility and value at all times, distinguishing between technical and biological cycles."[?]

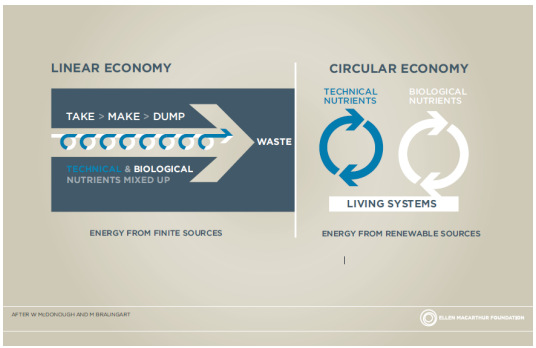

Figure 1: Linear economy and circular economy

The current process is a linear one which has a "take, make, dispose" process whereas the circular economy has a recycle, reuse, reduce process. Disposal of waste still exists in the circular economy but it is at the bottom of the hierarchy (figure 2). The online marketplace would work from the reuse stage of the hierarchy facilitating the reuse of waste and bi-products from industry.

In Scotland there was 2.5 million tonnes of waste sent straight to landfill from the 8.6 million tonnes generated by industry in 2010.
This represents a large loss of materials from the economy and also forfeits an opportunity to reduce emission through developing the circular economy. There is also a lot of money wasted by businesses by doing this. In 2008 the manufacturing sector produced 22.7 million tonnes of waste costing businesses € 1.8 Billion and this cost will rise by 50% by 2020.
Businesses have to pay landfill taxes and logistical costs when it comes to disposing waste which means they lose money from these materials.

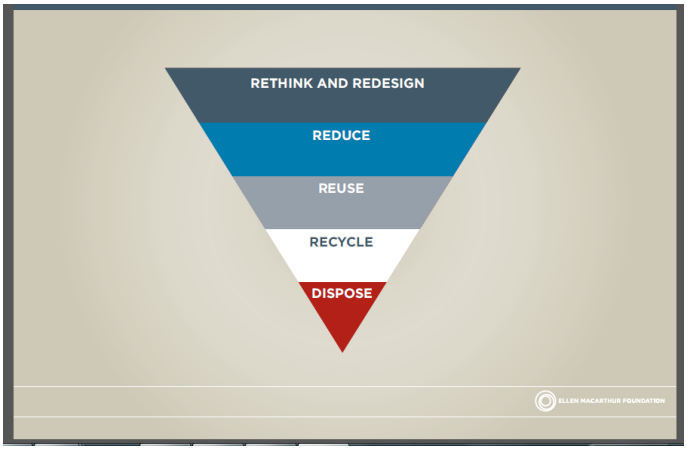

Figure 2: Reusing waste

2.3 Objectives

During the first meeting with our customer we found out the core requirements.

- The site needed to be a global marketplace, without any currency constraints, and there needed to be some way of dealing with all the legislation tied to trading waste for each type of waste in all the different countries.
- We needed to decide what types of waste we would deal with and if we would specialise in just a few types since the legislation that would need to be looked at would then greatly decrease.
- We also needed to plan a way to monetise the site so it could it at least support itself in terms of costs.
- And what types of users there would be and the general design of the site and what they could do.

The site is a global marketplace, with no legislation constraints as well as no currency constraints. This meant we can deal with all types of waste and create the greatest green impact. After speaking to some companies who could potentially use the site in the future we decided to have a single user type, as the companies would want to both buy and sell and there was no need to separate these accounts.

They also mentioned the idea of having a "distributor" or "logistics" account, this account type is for users who have no waste to sell and dont need to buy any waste but can cover the logistics of transporting waste. We also implemented a statistics page which had become a "Should have" requirement later on in the development process.

This included statistics for the site and a live twitter feed for the site that can raise awareness of the site and the financial and green benefits it offers. We also

planned a way to monetise the site, although did not implement it as we did not implement a payment system and this will be explained in further detail in Section 6.

3 Agile programming and teamwork

3.1 The team

To find suitable roles for everyone we discussed what experiences we all had and what we wanted to get out of the project. Everyone was then assigned suitable roles that would make the most of their experiences.
Our team consisted of:

- Project manager + front end developer
- Customer liaison + front end developer
- Front end developer
- Tool smith
- Back end developer
- Database engineer + back end developer

After the first few meetings we realised we did not have the need for a tool smith as the tools we would be using had been setup and configured and managing them was not an issue. However, as the code grew, it became duplicated and uncommented. There was no specific programming style being used for naming variables and the code became increasingly difficult to understand. These are the known risks of following the agile methods.

In retrospect, we know we should have used the Extreme Programming strategy from the beginning with its test-first software development model which could have assisted us in quality assurance and design activities. By doing this some of the challenges we faced could have been avoided because we would have created the tests before implementation and lower the growing cost of debugging in the future which we experienced. This is shown in figure 3.

In order to address these problems the tool smith was assigned the role of quality assurance engineer and we decided to then follow the Extreme Programming method.

It was the Quality Assurance engineers responsibility to perform constant refactoring in order to improve the design and readability of the code, create tests and to produce source code documentation. We wanted the comments to explain the purpose of the code without being verbose so it could be updated easily in the future. To achieve a good balance of detailed but short comments we experimented

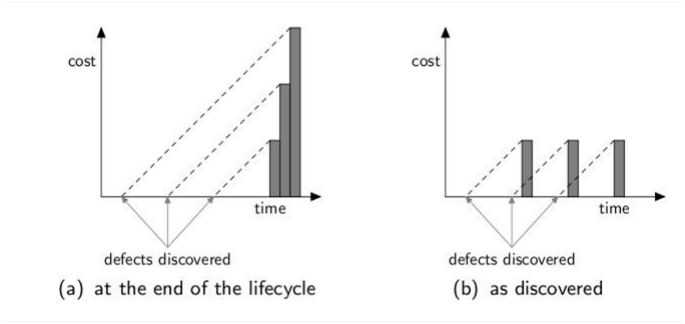

Figure 3: Cost of fixing defects

with different styles and tested them with members of the team to see if they could understand it and found it helpful.

The coding style became unified with meaningful class and variable names as well as consistent formatting throughout. All outcommented and duplicated code was removed and it became easier to understand.

3.2 Testing

The tests were created at the end of the project. However the libraries that were recommended in official documentation did not work or created conflicts with other libraries being used which produced errors or prevented some features to work.

The main problem was that the Play platform libraries were frequently updated or replaced with libraries that were not backwards compatible (many of them have changed API). We also had this issue while developing the back end and this was solved by trial and error by testing different libraries.

If we added a conflicting library to fulfill the testing requirements we risked being unable to deliver a working system. Both The Agile Manifesto and Agile Principles emphasise having a working solution, therefore we prioritised that over having working tests.[**?**][**?**] Nevertheless the test templates were created for each part of the application and can be used in future development if unconflicting libraries are found. Currently these tests are switched off in order to enable the build of a working application. Nevertheless the test templates were created for each part of the application and can be used in future development if unconflicting libraries are found. Currently these tests are switched off in order to enable the build of a working application.

However we could have done it in a different way. The solution is change management workflow, namely parallel development. We could have created a branch dedicated to implement testing. Then design working tests there and gradually add

small sections of the existing application. Eventually we would have the whole application together with working tests and make this branch the official version.

4 Client Meetings

4.1 Background

After initial contact with the project customers, it became clear, from their suggestions and the nature of the project itself, that meeting with potential clients who could have a use for the finished application would be of great use in terms of understanding the requirements and which features the application should have. The customers offered several contacts that they had within the industry with whom we could meet and discuss ideas, however it transpired that the earliest meeting we could get with these clients would be some time after Christmas, much later than when we wanted to have a good idea of initial requirements for the project. Thankfully one of the team members had his own contacts who could meet at a much earlier basis so this allowed for a better understanding of the project at an earlier time and in turn allowed the next phase of the project to continue.

4.2 The meetings

Two different firms who were seen as potential users of the application were met with - Scotia Nitriding/Ceramco and Malcolm Joinery. Scotia Nitriding and their partner company Ceramco are in the business of making various materials more durable for use in other process/providing ceramic moulds for various chemical processes. Malcolm Joinery is as the name suggests a joinery firm but also has a part in plumbing and electrical work.

The first meeting with Scotia Nitriding/Ceramco took place at their factory and it was immediately obvious even from the materials scattered around the place that the businesses deal in an extremely large variety of different materials. After discussing a little about what the company does, we posed the question of whether they currently bought or sold waste products on a regular basis and it was discovered that occasionally the company buys a specific type of waste product which they then transform into a new more usable material, which they then sell on at a profit. They showed some interest in a platform which allowed this trade to be completed more easily as currently it is difficult to find sellers of this industry specific material. Furthermore, the client spoken to indicated that they currently sell/recycle certain by-products such as scrap metal, however an auction like system which allowed scrap dealers to bid against each other would be of interest. Finally when asked for any suggestions of features they would like to see implemented, the clients view was that the site should be account driven. They also suggested a transport/disposer account type.

The second meeting took place once again at the clients place of work. Once again, there was a some talk about what sort of work the firm usually undertake and which kind of materials are dealt with on a day to day basis, once again we were surprised at the variety of materials dealt with.

The second client was asked the same questions as the previous. First we asked whether or not the client currently traded in any waste or by-products and their answer resonated with the previous client in that they recycled/sold some material, however finding a buyer is not always possible. This client also mentioned that they are frequently left with off-cuts of wood which are hard dispose of in any sort of profitable way.

In terms of features they would like to see implemented, this client mentioned the importance of contact information through a profile feature for each account. His intention was that users would be able to meet through the site but deal with much of the logistics of the trade themselves, either face to face or through email/text. Finally, this client mentioned the idea of allowing users to advertise their coherence with certain regulations for certain forms of waste on the site itself, this tied in with our earlier realisation that certain types of waste material require certain certificates to trade in legally.

4.3 What we took from the meetings

Both client meetings were extremely helpful as they gave us a much better understanding of what features our application should have and allowed us to see how it could be used in a real life, professional context. The insight into how firms currently trade waste materials allowed us to better understand the nature of the industry itself and this directly translated to the requirements that we as a team elicited.

From a design point of view, the most important factor concluded from the meetings was that the web application should be a platform in which buyers and sellers can find one another, as opposed to our initial idea that it would be an eBay type auction site where all stages of the transaction were handled. Instead, it seemed that a Gumtree type classified ads website was much more applicable as often the transaction will be complex and not necessarily Buyer A gives Seller B a fixed amount for some goods.

This idea of a platform is reflected in the final design of the site, whilst users can post and make offers on listings, the transaction itself is dealt with elsewhere. This allows the seller to choose from a number of different offers based not only on the amount offered, but also on the proximity of the buyer, whether or not they have the required documentation to deal in the type of waste in question, and any previous experience they have with the potential buyer (positive or negative).

The experience of talking to real life clients about the project was initially challenging due to their - completely understandable - lack of technical knowledge relating to website development. However, we soon developed a way of discussing features

and aspects of the website in a non-technical way which allowed us to see the clients opinions in a constructive way. For example, when discussing with Scotia Nitriding any features they would like to see implemented, they suggested that the platform be account driven. As a standalone comment this suggestion offers little bearing on how the site should be set up. However after more discussion about this feature - whilst all the time steering clear of an jargon or technical terms - it became clear that the feature they were suggesting was that accounts themselves should each hold a great deal of detail - location, contact information, certificate upload feature. This realisation is what lead us to the Meeting Platform basis of the application.

5 Front end development

5.1 Wireframes

After the initial meeting with our customers we decided to create an initial prototype for the front end of our application by producing wireframes. By the third time the team met we had drawn out rough sketches of the front end on paper. We then decided to redraw them on the 'moqups' web app as depicted in figure 4. This was so we could produce digital copies as they would provide us with the portability required to develop the front end as everyone could have a copy of the designs.

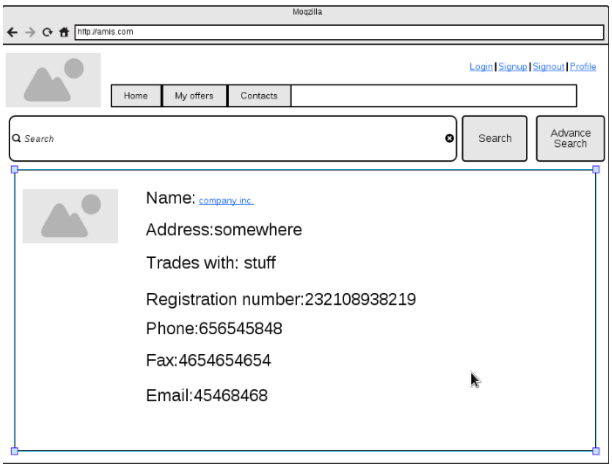

Figure 4: Initial wireframe of the site design and user profile

5.2 Prototype

Using the initial wireframes as shown in figure 5, we developed a prototype covering a small section of the site that had limited functionality but was sufficient enough to present to our customers at an upcoming meeting.

At that meeting we showcased our designs to the customers and received generally

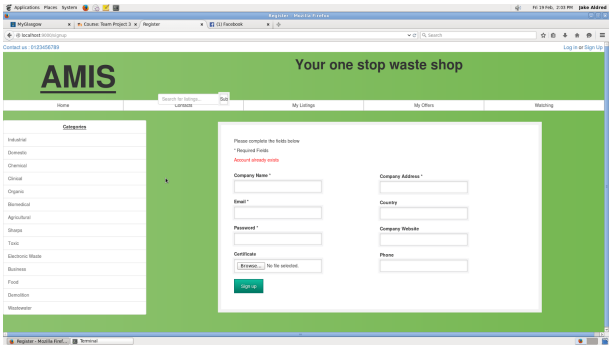

Figure 5: Prototype design of the site

positive feedback about the look and feel of the site, however the customers mentioned that they thought the top header part of the site took up too much room and the same about the categories section.

5.3 Final design

The final design of the finished application, as it is shown in figure 6, was reached through a continuous design process comprised of many incremental changes.

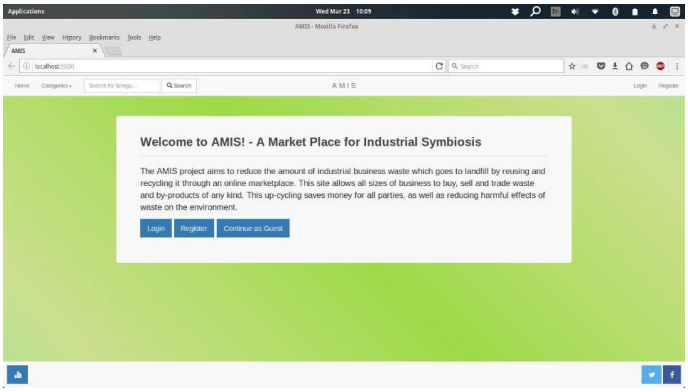

Figure 6: Final design

After hearing the customers feedback regarding the initial prototype, the top navbar was implemented. This has the advantage of saving space previously wasted by the top header, and also allowing categories to be hidden in a dropdown menu, freeing up more room for listings and other aspects of the site and also allowing all categories to be browsed through without the need to scroll. In keeping with the eco-friendly message of our site we decided to keep the green background but make it a lighter shade as comments we received from customers suggested that the previous colour was rather overpowering.

The navbar implementation has the added advantage of being scalable on smaller screens using the bootstrap navbar class which includes the hamburger icon, which

can be expanded to show the various navbar links. Finally, a sticky footer was implemented to add social media links, and also allow a link to the newly created statistics page to be accessed wherever the user is on the site.

5.4 Challenges and gained knowledge from front end development

Throughout the development of the front end we encountered many challenges as a team which required collaboration from everyone involved in the project. One of the main challenges was ensuring that the team members working on the front end communicated frequently to ensure the designs were consistent throughout. At the beginning of the project the colours and text size used varied depending on who was developing at the time. When this started to affect the overall design of the site the members of the team responsible for the front end decided on set colours, fonts and sizes to use throughout the project. This helped us to ensure that the design was consistent and alleviated any confusion regarding the look of the site.

If a member of the team wished to change the design of an aspect of the web application they would inform all the other members of the team and we would decide whether to accept their changes or keep the current design. An example of this is when we decided to change the design of most forms from having multiple columns of text fields to just one. The member of the team who put forward this change notified everyone else, we all agreed and so the forms that would benefit from this change were altered throughout the site. However, this wasnt always the case as at the start of the project members would change the design of the site without always consulting other members. This resulted in members having to revert the designs back to the previous one if the team didnt agree on the design which took up valuable time.

Another challenge was deciding when to sacrifice a design aspect the customer would like to ensure that the overall design of the web application wasnt compromised. During the second meeting we asked if we should design our own logo or if one would be provided. The customers said we should make our own. As a team we decided to leave out a logo on our web application as having AMIS in the center of the navbar seemed more aesthetically pleasing to us than having an image as the customers upon seeing the new format, agreed.

Finally, perhaps the greatest challenge faced when carrying out front end development was integrating pages and changes with the back end of the application. There were some complications which involved communication between the back end and front end sub-teams and working on both areas was initially difficult to coordinate. However, after some discussion during retrospectives this miscoordination was corrected through better use of Trac, and both front and back end sub-teams commenting code to a better extent. For example, if a new page was created by a front end member which included buttons to link to other pages, or required some back end functionality, the member would post a ticket on trac explaining what back end functionality was needed, and explain the feature to the other members.

Therefore, having faced these challenges in front end development, we have

learned how to better coordinate our work between all members of the team. More frequent communication between front end and back end sub-teams significantly improved our workflow. Using Trac more to convey our intentions and making sure everyone knew which part was currently being worked on was also a great addition to our teamworking capability.

6 Back end development

6.1 Architecture and data models

After the first meeting with our customers and extracting our functional requirements from the user stories that we developed, we created our initial architectural model as depicted in figure 7.

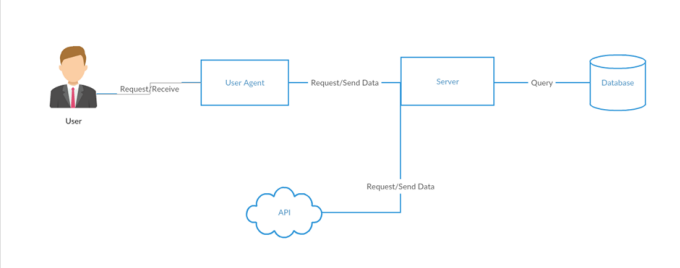

Figure 7: Architectural model

The idea behind this initial architectural model was that the application would have 3 main components: a server to handle user requests both from web browsers and from other applications that use our API, a database to store our users data and the listings and offers posted on our website, and an API to allow future development of a mobile application that would work as an extension to the main website.

After developing the architecture we began work on the database model that we were going to use. We discussed the structure of the database with our clients and came up with the model depicted in Figure 8.

The initial model comprised of 6 entities: the admin a specialised account that will allow the application admins to manage the web app, a buyer account which can be certified or not if certified the buyer would be able to place offers for the waste that they are certified and allowed to buy by law, a seller account,a listing which will represent the items that our users wish to sell, an offer representing the price and amount of waste a potential buyer wishes to buy and the waste entity that represents the type of waste that is being traded.

During the process of development some of these entities were dropped such as the waste entity which became a type attribute of the Listing entity. The reason for this was that we wished to cover a much greater number of types of waste that our

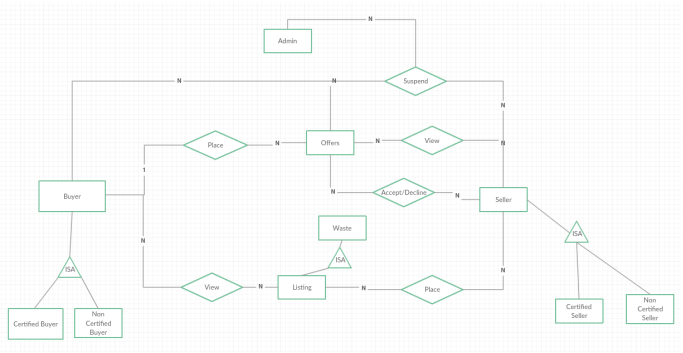

Figure 8: Database

clients wished to trade with. We replaced the waste types with several wide-ranging categories which allow for a much broader number of items to be traded.

After meeting with our clients in December we decided to drop the certified and noncertified user types and to merge the buyers and sellers into a single entity. The reason for these changes was that our clients wished to be able to both buy and sell waste at the same time which was previously impossible under the old model since they would have to register 2 separate accounts. The reason to drop the certified and noncertified accounts was that implementing a system which could make an accurate distinction between the 2 types would be too heavy given the amount of legislation that must be taken into account and the lack of a global unified legislation and or database of companies and the types of waste they are allowed to work with.

If we were to implement such a system we would have to either hire people that would have to read every account application and make a decision on whether the user is allowed to trade with a given type of waste, which wouldve taken an enormous amount of time, which made it unfeasible for this project. The solution we came up with was to allow users to upload the documents that prove that they are who they say they are and to allow others to view them and decide whether the given entity is who they say they are. The ER diagram of the final working data model can be seen in figure 9.

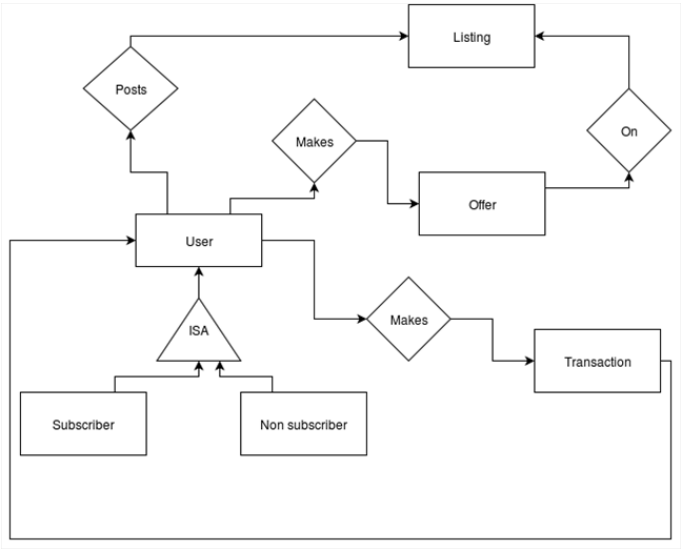

Figure 9: ER diagram

As Play is a framework that follows the model-view-controller (MVC) architectural pattern, to display the appropriate information on the website we had to create appropriate models that would underline the information to be stored in the database and then create the appropriate controllers that would pass information onto either the database for specific information to be retrieved or the views for a page to be displayed with that specific information. As per our initial data model, we knew that the most important models to start with were the User and the Listing models and later the added Offer model.

For the models, we could divide our code into the following chunks:

- appropriate variables that would store the user/listing/offer information (name, address, website, phone, country, email and password),
- getters and setters for these variables,
- default constructor that would allow us to create the user/listing/offer,
- validation method that would be used during registration/listing creation/offer creation to check whether all the required fields are valid and not empty.
- authentication method exclusive to the user model that would allow us to authenticate the login details during the login process (checking whether user exists and checking entered password against the stored hashed password),
- appropriate finder instances of the Finder object to perform searching of appropriate data.

This structure of our models allowed us to perform the necessary tasks we needed - saving appropriate instances of these models, performing a search for them (by id,

posted time, name etc) and to verify that when they are created that the appropriate fields are valid. This would then be connected to the front end via the controllers. From this experience, we had learned how to further develop and visualise the stored data with the help of an MVC framework. As we had experience with Django, a lot of the previous experience was carried over, which meant we had the opportunity to hone our skills and get practice from a different perspective since we were working in a different language.

6.2 Security

After meeting with our clients in December and February we identified a new requirement for our application. The application must provide an industrial level of security in order to protect itself and its users from hacker attacks. In order to meet this requirement we conducted some research to identify the most crucial vulnerability areas of Java Play applications in general and in our own application in particular.

Initially we consulted the Play Frameworks documentation [**?**] which provided us with the following vulnerabilities: SQL injections, session injection and session hijacking. After identifying these vulnerabilities we made changes to our search function in order to prevent SQL injections. We sanitise the queries passed to the application via the search bar, removing any SQL escape characters such as semi columns and single quotes. In all other scenarios where the users request data from the database the data is returned using the Ebean API for the Play framework which provides an extra layer of security by abstracting the exact interactions between the applications back end and database server.

In order to eliminate any vulnerabilities that may arise from our session handling, such as session injection and hijacking, we had to restructure our code and migrate from using simple session storage, which just stores data in plain text form on the users machine, to a more complicated model which uses a separate database entry in order to make sure that the user is logged in from a single location at any given time. The challenge with this new approach was that we had to work from scratch since the Play framework does not provide any built-in mechanisms to prevent identity theft. Previously we used simple session storage to save the users email in order to know when they are logged in and when not. The problem with this approach was that if a hacker knew the email address of a given user all they had to do is change their cookies to have the same email as the victim and then they would enter the users profile bypassing the password authentication process.

To resolve this issue we implemented a class called Session.java which stores a randomly generated session id which is associated with a given user. The session id is generated when the user logs in and is deleted after they log out. The randomness associated with the way the session id is generated insures that hackers wont be able to hijack the users identity by simply changing their cookies since theyll have to guess a randomly generated 128 bit number which is based on the time that the user logged in at.

After conducting the research and inspecting the code for any potential leaks due to its structure or functionality we discovered that the application might suffer from functionality abuse and transport layer vulnerabilities. The challenge with eliminating leaks caused by the applications functionality was that these leaks are very hard to identify at first and most of them we found during testing and debugging. The leaks included the ability of a hacker to call upon back end functions just by passing the correct URL. For instance in order to make an offer the hacker would only have to send a simple POST request to the following URL http://amis.domain/offers/create/id where id is the id of the listing they wish to post an offer to which they could get by just viewing the listing itself. In order to prevent such abuses we added checks to every method that is exposed externally, which check if the person who is sending the request is logged in and whether they have the necessary permissions to execute the method which they are trying to invoke.

After solving the functionality issues we started work on transport layer vulnerabilities. After doing some extra research on transport layer vulnerabilities we found out that we cant make any changes to our application that would prevent any attacks against us from the transport layer since we dont have access to the inner workings of the Java Play server nor to the users devices. What we did instead was to provide encryption for our application in order to make sure that even if a hacker manages to conduct a successful attack against us and steals our data they would not be able to read it nor use it. In order to provide the best possible encryption we consulted with other teams as well as researching online and what found out is that the best algorithm is Rijndael.[**?**] The challenge we faced with adding encryption was that there was no library or module in the Play Framework which provided this algorithm so we had to make our own implementation. To do this we created a class called Rijndael.java which provides functions for encrypting and decrypting text. Our implementation was based on the Android implementation of the Rijndael algorithm found at [**?**].

6.3 Payment System

For a large part of the project, monetisation of the website was a big challenge that we had to consider. We had considered a number of possibilities and came to the conclusion that if we did implement it in any sort of way, it would most likely be via a payment system where users could pay for the listings they are interested in. After the payment is made, a percentage would go to the website itself. PayPal would be the main payment system to use as it is currently one of the most used alternative payment systems in the world with over 179 million active user accounts.[**?**] We had found out that PayPal integration would be relatively easy to implement. However, after discussions amongst ourselves and with our customer, it was decided against it.

The reason was that it would increase the risk of misbehavior on the website, mostly involving scamming. This would increase the responsibility for the website managers and the authenticity of users could not be 100% proven without performing appropriate checks, which is out of scope of the project. We did implement a way for a user to upload a certificate to help prove their authenticity, however, there are no

checks performed whether it is actually a real document and therefore it could pose a risk for users.

From this we had learned that even though there is a way to implement a feature, there are a lot of factors that have to be considered and for such topics very detailed discussions and appropriate measures have to be taken before these features are actually implemented into the system. In this particular case it was very important to talk to the customer about the possible risks that could arise from implementing such systems and due to no prior experience with monetisation, we, together with the customer, came to the conclusion that it is the best to focus on the other aspects, mainly functionality of the website.

6.4 Challenges and gained knowledge from back end development

We came across quite a few challenges and issues during back end development of the application, which allowed us to gain quite a significant insight on developing these parts of the application but unfortunately slowed down the progress of creating a more complete application.

When we decided to work with Play, we had done a quick speed prototyping test to see whether we would like working with it and whether it would suit our needs and after working with it for roughly a week, we decided to continue with developing with it. However, we had naively assumed certain workings of the framework.

Our first challenge was creating appropriate authentication and validation of the users. Unlike Django, Play does not provide out of the box registration functionality or authentication methods of its own. Instead, we had to create all of that by ourselves. Having created the User model, we had to ensure the following things:

- the entered user details are checked when registering,
- the user data is secured properly, especially the password,
- the user details are appropriately checked during the login process,
- the user is appropriately saved in the database after registering.

This process of manually creating the necessary requirements for user creation has greatly helped us gain an insight into how user registration works in web applications.

The other challenge we faced was having data securely stored, especially the passwords. Compared to the Django framework where we could trust that the data, especially passwords, was secured after registration, we had to ensure that ourselves in Play. Therefore, as underlined in the Security section 6.2, we used the Rijndael algorithm to store most data and BCrypt which was used for hashing the passwords and storing them as such. This has further helped us understand the necessity and

complexity of such algorithms and the procedures of storing sensitive user information.

The final important challenge we encountered, which we were unable to fix, was handling session data. As discussed in the security section, we had implemented a new way of handling user session to ensure further security of users data. However, we had encountered a problem where if a user logged out and logged in with a new user or if the person had closed their browser without logging out and reopened it, the site would throw errors on certain parts of the website (making offers and so on). This happened at the end of the development cycle and we could unfortunately not fix the issue in time.

This stemmed largely due to the inability to implement testing into our system and due to our lack of understanding of the framework. From this we definitely learned that important aspects of the website like that have to be tested thoroughly and consistently to ensure that no such problems are discovered at the end of development. However, we also learned a lot about how sessions work and how they should be managed and had we found a solution to the problem, we believe that it would be a good working solution, improving the security of the website by a large margin.

7 Conclusion

We learned a lot from this project. It was also the first time we had worked with an agile software development process. In the first few weeks of the development process we made many of the common mistakes found in the agile software process. After the reorganisation of the team and stronger leadership we were able to again learn from our mistakes and understand how to prevent them in the future.

In hindsight, we realised we should have created tests from the beginning and failing that used parallel development to create the tests instead in a different branch, having one branch with the working project and the other for the working tests. However ensuring the project worked was the primary goal and that was accomplished.

Talking to and working with real clients who would use our product and working with customers for whom the project was created for also gave us a lot of valuable experience. Customers explained what they wanted and we spoke to the clients to find out what they wanted and communicated that back and forth till we got a product that met the needs of both stakeholders needs. It also gave us the skills to talk about a technical project without using jargon and in a non-technical way so we could communicate ideas with our stakeholders easily.

In a technical respect, our web development skills grew a lot. None of the team had experience with the Java Play framework but learned to work with it. Although we had some issues with the lack of security provided by Java Play this gave us the opportunity to implement our own security measures, and from researching solutions and implementing them we learned a great deal.

We found the retrospectives at the end of each sprint to be particularly useful and this helped us identify problems in our development process, tools and team. Being able to so quickly identify problems meant we were always able to quickly find a fix to them before they grew too large.

For some of us it was our first time working in a team environment of this size for this long. We got to experience working in a team where everyone was working on different tasks. Having to integrating all the different sections of the project posed many challenges initially. Overcoming these challenges helped us develop valuable communication and team working skills as well as learn many other technical and non-technical skills.