University of Glasgow | School of Computing Science

Level 3 Project Case Study Dissertation

Building Healthy Communities Database

Christopher Sean Harris, 2128878h
David Andrew Brown, 2137184b
David Robertson, 2135967r
Jaklin Yordanova, 2122832y
Kiril Ivov Mihaylov, 2062132m
Maria Papadopoulou, 2136603p

March 28, 2017

Abstract

Building Healthy Communities in Dumfries and Galloway is a community development programme which aims to advance the health and well-being of all individuals within its area, but particularly those in difficult circumstances. Its main goal is to improve the lives of people who may otherwise become isolated, by encouraging participation in initiatives which help keep them physically and mentally healthy, while also promoting the creation and sustaining of social bonds. The programme consists of three different types of participants:

- Administrators
- Volunteers
- Members

Members are the individuals participating in the initiatives themselves. Volunteers run the initiatives and are responsible for tracking the member's progress and attendance. They must inform administrators not only about the progress of the members, but also about the process as a whole. Currently all this is being performed by hand. Administrators must manually input the data into a database and then search the database using only the inbuilt functions of the database software. This is a difficult and time consuming process.

Our team was responsible for building a system to automate the input and retrieval of relevant data. We designed and implemented a front-end interface that could recreate the current database and allow it to be modifiable and extensible. The system has different features for different types of users.
An administrator is able to:

- Add and delete users, areas, initiatives and medical conditions.
- Search, filter and organise various data records in the database.
- Enrol users into initiatives.
- Assign medical conditions to users.

- Export tables into csv format.
- Display various statistics and metrics that can be downloaded.
- Manage the funding for users, initiatives and medical conditions.
- Access the supplied personal information of all users and make changes if necessary.
- Archive data records.
- Keep track of the progress of members and volunteers.
- Review the answers from the feedback received.

A volunteer is able to:

- Leave feedback detailing their experiences.
- View the initiatives they are enrolled in.
- Create a new session for the initiatives they are assigned to.
- Keep track of a members' attendance to those initiatives.
- Access important details about the users in the initiatives they are assigned to.

A member is able to:

- Leave feedback detailing their experiences.
- View the initiatives they are enrolled in.
- Make a request to an administrator if they wish to change their personal information.

1 Introduction

Third year undergraduates on the Computing Science course of the University of Glasgow undergo a year long team project, the aim of which is to give the students experience of working in a team on an extended software engineering project. It helps to develop an ability to work with customers, and teaches the use of agile software development practices and methodologies. Importantly, it also teaches the value of retrospection; the act of looking back to see what can be improved in future. With this case study, our intention was to analyse and explore the methods and practices we employed, in order to hopefully better understand their intricacies, how they should be used in a project such as ours, and to potentially develop new, better ways of applying these methods in future.

This paper presents a case study of the building of a database system for the public health program "Building Healthy Communities", and the development practices we used in the process. The BHC system is designed to replace the cumbersome, manual database currently in use by the NHS team that administers the BHC initiatives. It is intended to be a much more streamlined service, allowing for much faster input and retrieval of data from the system, while also allowing for varying levels of access, depending on who is using the system. The aim of the BHC programme is to encourage individuals who may be experiencing difficulties to lead healthier, less isolated lives through a collection of healthy living initiatives. A system which allows for the secure, rapid organisation of data helps ensure this aim can be easily met.

Throughout the following pages, a more in-depth explanation will be given on how the software described above was designed and created using not only our previous knowledge, but also the knowledge that the team managed to gain from the university's courses. More specifically, the "Professional Software Development and SIT" course and the "Interactive Systems 3" course played a critical role in every step that was taken in creating the software. From requirements gathering to the coding of the software, close study of the material allowed for the development process to be much smoother than would have otherwise been the case, and led to a more polished product.

2 Case Study Background

2.1 Customer Organisation and Background

Building Healthy Communities is a programme based in Dumfries and Galloway, utilising the concept of 'Healthy Living Centres'. Founded in 2001, it is a partnership of public, community, and voluntary organisations that operate region-wide to improve the health, well-being and quality of life of all people, particularly those in difficult circumstances. The programme consists of a regional partnership of strategic and local representation to help shape and direct, alongside four local area partnerships operating across the region. These area partnerships co-ordinate and take collective action by creating initiatives to tackle community health issues and the root causes of ill health.

The primary method by which Building Healthy Communities tackles its main objective is the creation and running of health initiatives. These are regular events, run by community members, for community members, with the aim of bettering the health of individuals and the community as a whole. They foster a sense of well-being, and focus on people who might not otherwise be engaged in social events and feel isolated. Individuals are given the opportunity to become community health volunteers through training and one-to-one support. This allows local citizens to feel active and helpful in their community.

While initiative events are run by volunteers from the local communities, the programme itself is run and maintained by a group of NHS employed administrators. These administrators are NHS employees, and have access to all the data produced by the programme. This includes the personal and medical details of all service users and volunteers, along with their feedback forms, and the funding status of every user, initiative and medical condition. During the project development, contact was had with two of the programme's administrators. They attended each customer meeting, and were instrumental in the shaping of our product, providing requirements and feedback. There was also some incidental contact with another NHS employee from the same area running a different project, whose more technical advice was of particular use when designing the database and deciding which information needed to be included.

2.2 Initial Objectives for the Project

The initial objectives for the project were to develop a software which will be used by the service users, volunteers and administrators of the programme. Different types of user have different levels of access to the database. The service users would use the software to complete a small questionnaire and give a regular feedback for the initiatives they attend. Volunteers, on the other hand, would be able to view that feedback and check the attendance of the members for the initiatives they are assigned to. Administrators have the access and rights to do everything, they can see all the information about both volunteers and service users, they can add or delete an initiative and can review feedback and attendance as well. Since we are following agile development practices, the initial requirements could change over the following customer meetings, as explained later in the dissertation.

3 Agile Programming, Technologies, and Team Organisation

3.1 Team organisation

During the initial team meetings, every team member identified their strong and weak points in order to divide the work. Due to the fact that the creation of an application and database from scratch was required, the workload was huge. As a result of that, everybody initially worked on the "backend". Inevitably, certain team members had a higher level of "backend/coding" expertise and thus were tasked with the more complex areas of development. The team discussed various means of communication and decided to avoid the usage of common social media sites. The group-chat application 'Slack' was used throughout the course, which provides a more organised and professional chat room with different channels.

3.2 Agile Methods

In terms of project planning, agile methods are one of the most trusted methods to use. Specifically, they are suitable for small teams and they involve frequent customer meetings because the requirements cannot be fully collected at the beginning of the software development cycle. Particularly, they are based on focusing on the code instead of the design which is suitable for our software project, since the proposed application is to be created from scratch. The project was based on one of the agile methods called "Extreme programming". This was the most suitable decision, due to the fact that the software is being developed by novice programmers. One of the most useful features that Extreme Programming offers is called "pair programming", which was constantly being applied. Since the project was developed by students, there was a major issue regarding the lack of experience. Pair programming negates this issue to some extent by allowing team members with different levels of expertise to combine their knowledge and come to the best decision. Additionally, pair programming increases work productivity since working in pairs motivates even the less productive members of the team to contribute. Furthermore, user stories and prototyping played a critical role in understanding the application requirements. Finally, another important feature of Extreme Programming is the automated testing which will be described in more depth in section 6.

3.3 Technologies Used

With the agile methods that are going to be used already outlined; the technologies needed regarding the project management have to be outlined too. The university Moodle page suggested a litany of technologies to use for handling the project such as "Ant and Ivy", "Jenkins", "Apache", etc. After a small group meeting and a discussion with the supervisors about what technologies were permitted, the decision was taken to use Gitlab [7] for our project and repository management.

Gitlab is a complete repository management system, in that it not only provided us with a git based repository, but also gave us an integrated wiki in which to store documentation, a comprehensive issue tracking system like Trac, and a continuous integration suite in the same vein as Jenkins. The combination of all of these features in one discrete package made Gitlab the obvious choice for us, negating the need for multiple cumbersome systems that we would need to keep track of separately. The next few paragpraphs illustrate the reasons why the use of Gitlab turned out to be a wise choice.

The first major feature Gitlab provides is the git based repository that gives it its name. This repository allowed the whole team to simultaneously work on the project in different branches, either collaborating on a single section or working individually in separate areas, all without accidentally overwriting each other's changes or breaking each other's code. The repository also features a full history, so we could revert changes if a large mistake was pushed.

The issue tracking system in Gitlab is a comprehensive feature that allowed us to create detailed tickets containing checkboxes of tasks, time estimates, comments and more, allowing us to divide the project into a series of issues. These issues can also be assigned labels; the labels allowed us to see at a glance which area of the project the issue was for (a new feature, documentation, testing etc). The created issues were generally assigned to team members, so that those members had a clear focus, and everyone else could see who was working on what. Finally, Gitlab's repository and issue tracking system are linked, which meant that we could create a new branch for each issue. This greatly simplified the process of implementing and testing features, by making each small feature its own issue and spawning a branch from it. The feature could then be worked on in isolation, and when complete and tested a merge request could be put in to integrate the branch back into master with the feature hopefully complete. The lead developers would review the code before accepting the request, and either leave a comment about any problems and wait for their resolution, or accept the request, which also automatically closes the relevant issue and deletes the merged branch.

Gitlab also features a Continuous Integration suite [3], an extremely useful toolkit that we used to continually test the system as soon as a new feature was implemented. We created and continually updated a set of tests in a test image that would be run on every important commit, in the hopes of ensuring that nothing would slip by and break the project. In the case of a feature branch created from an issue, the CI suite would build a test image of the project, and then run that test image, returning a log indicating whether the build and tests were successful. In the case of a commit or merge request to master, the suite would do exactly the same as above, except that it would also deploy the project, so that we always had a live version of the project active on our server. To test and deploy the images we used Docker alongside the Gitlab CI suite. Docker is a modern open source container platform, that automates the deployment of software by using software containers that contain everything required for an application to run, ensuring it will run the same regardless of environment. Docker is described in more detail later in this section.

The final useful feature provided by Gitlab is the wiki. This is an easily editable collection of pages used for the project documents. We created an organised system for documentation, with a sectioned wiki homepage linking to a variety of documentation types, from our design documents, to customer requirements and meetings, to our retrospectives. This simple structure allowed us to keep a very clean and easy to understand collection of documents that could be referred to later, helping us to keep focused and organised.

A decision was now to be made on which technologies were to be used to implement the application itself. Namely, a web application framework was to be chosen. All members of the team had some degree of experience in Django [1]; however, other alternatives were explored to ensure the best option was selected for this particular project. Ruby on Rails was a prevailing suggestion, and whilst being conceptually similar to Django, has been around for much longer. Rails has a mature and reliable code base, abundant documentation and excellent built in testing support, as seen in section 6.4 [10], [13]. Also, with Rails being an industry standard for modern web application development, the team felt it was a valuable experience to develop a large scale application using this framework. With all this in mind, the team chose to go ahead with Ruby on Rails. One downside of Rails is the lack of official programming style reference, something that Django does have. Despite this, many community driven style guides do exist for both Ruby [5] and Rails [6]. There was high emphasis placed on trying to ensure our code conformed to these style guides. Following a style guide not only helps keep code efficient, but it allows other developers to quickly identify the intent of code because, in theory, everyone should be working to the same style. It also ensures that for projects with multiple developers working simultaneously, a consistent layout is maintained.

3.4 Retrospective

In order to regularly summarise the team's progress, the usage of retrospectives was a useful tool. Retrospectives are a small summary of a past situation. In software engineering, a retrospective is used to summarise the actions that were performed during a specific amount of time, or a 'sprint'. For our team, sprints typically lasted approximately 2 weeks; however, due to many factors not changing significantly in a two week period, we decided to only complete a retrospective at the end of our monthly customer meetings. The creation of retrospectives was done on "Trello". Retrospectives are a very helpful tool in terms of stating what was good during the milestone, what was lacking, and what could be improved. To be more specific, the 4L's: Liked, Lacked, Learned, Longed for was used. A new retrospective was created every time a customer meeting was performed, which indicated the end of a milestone and an iteration. Every team member participates in the retrospective and fills at least one thing on the board. An issue in the Gitlab is raised for tracking the successful completion of the retrospective by each participant. Furthermore, the retrospective process took approximately one hour, allowing every team member to individually outline their thoughts and suggestions about the previous iteration. After that, an hour long meeting was performed, followed by a discussion where major issues were outlined which were identified through Trello. The team leader conducted the meeting and gave a summary of all the team member's opinions. This summary was then typed up and added to the wiki. The whole process was highly efficient in terms of observing and solving problems as a team. Retrospectives were a major positive factor in terms of seeking and underlining new issues. Having a more organised specification of the proposed issues enabled better time organisation which led to big improvements.

3.5 Team Roles

Due to the fact that the team was composed of novice programmers, the team roles were not clearly defined by the end of the first month. During the following month, after the first iteration, a meeting was performed, where the roles were clarified as follows:

- Chris Harris - Documentation
- David Brown - Lead Developer, Docker Manager
- David Robertson - Lead Developer
- Jaklin Yordanova - Documentation and Front End Developer
- Kiril Mihaylov - Documentation and Front End Developer
- Maria Papadopoulou - Documentation and Front End Developer

The team voted to elect a team leader. The team leader will conduct any retrospective meetings and any presentation of the application to the customers. Additionally, a role based model is being followed. Thus, the team leader is actually just a team manager, he is not making the final decisions alone but with the team's help. The team leader's job is to organise and delegate tasks to the rest of the team. The team voted fairly to elect David Brown as the team leader since he is one of the lead developers and was by far the most proficient at communicating with the customers. Over time we found that certain team members would face greater difficulties in resolving code-based issues than others, this is why we decided to introduce the concept of 'lead developers'. Since the workload of back end development is substantial, it allowed minor issues to be assigned to other team members, allowing the lead developers to focus on more advanced issues.

4 Documentation

After some initial consideration, we decided that fortnightly sprints would be ideal. Each sprint corresponded to the conclusion of a milestone. Issues would then be created, most of which having a deadline of the upcoming milestone. The following portion of the case study will chronologically outline the software process from the requirements elicitation, to the final software presentation.

4.1 Objectives

During the first customer meeting, where projects were assigned, the customers outlined their desired application. During the culmination of the meeting, the customers were asked what they wanted to see for the next meeting, their answer was: ”I guess, some sketches!”. A new milestone was created with the next goal to be some basic documentation and the creation of wireframes.

4.2 Requirements Gathering

During the first customer meeting, some basic requirements were outlined. Notes were taken during the meeting and a wiki page was created containing a summary of the meeting. Our aim during this meeting was to get a basic understanding of the functional and non-functional requirements of the website. Using our summary and further email clarifications we managed to establish a final version of the functional and non-functional requirements for the application.

The final version of the functional requirements are:

- An administrator should be able to view all information stored in the database.
- An administrator should be able to add/remove/modify volunteers, members and groups that are stored in the database.
- An administrator should be able to search and sort on specific fields belonging to tuples.
- An administrator should be able to view data metrics on how particular initiatives are progressing.
- Different groups of users should have differing levels of permission to the data stored.
- Volunteers should be able to view the data associated with their specific group(s), that they are allowed to see, and no other groups.
- Volunteers should be able to record member attendance.
- Provide a facility to register attendance.
- Provide a facility to gather feedback from end users (members).

The final version of the non-functional requirements is:

- The system should be secure.
- The system has to conform to the Data Protection Act.
- The system should be stable whilst in use and have a high up time.
- The system should be able to operate for long periods of time without intervention from system admins.
- The system should be simple to use.
- The system should be modular and hence maintainable.
- The system should be lightweight.
- The site should be compatible with many platforms. (Portable)

After we had identified the requirements, the team got divided into groups and documented various artefacts such as user stories, user scenarios and wireframes.

4.3 User Stories

User stories [4] play a critical role in the agile methods which we are following. A user story formally describes the state of the application and the basic functions that a potential user can expect it to have. Functional and non-functional user stories were created. User stories help to put the application into action and to fully understand how it will work.

A couple of highly representative functional user stories (with their corresponding story tasks) are:

- As an administrator, I want to add a member to the database, so that they can join a group.
  1. Add an activity to add a person
  2. Add an activity to enter their information
  3. Add an activity for choosing groups
  4. Create a query for retrieving groups
- As a volunteer, I want to be able to register attendance at my group(s), so that I can contribute data.
  1. Create a query to retrieve groups I run
  2. Add an activity to log attendance
  3. Create a query for members of the group
  4. Add an activity to enter attendance per member
- As a member, I want to be able to log in, so that I can view my information.

1. Add an activity to log in
2. Add an activity to enter log in information

Now, some non-functional user stories had to be outlined in order for practical examples to be created, specifying how the system will work. Some non-functional user stories (which their corresponding story tasks) are:

- As an administrator, I want to replace a member's name with numbers in the database, so that I can find them easier.
    1. Create a query for retrieving a person
    2. Add an activity to update a person's information
    3. Create a warning message for wrong type of input
    4. Do not allow this modifications to be saved
- As a volunteer, I want to be able to have access to the database, so that I can add a new program that the administrators haven't added yet.
    1. Restrict the "Add a new page", "Add a new page", "Add a new member", etc buttons to can be only accessed from administrators
- As a member, I want to be able to add or delete a group that I am part to, so that I can register my self to new groups or delete me from the old ones.
    1. Restrict the member's redirected page to do very basic tasks
    2. Do not provide this functionality to members.

4.4 User Scenarios

Although user stories provide a clear outline about what the user and the system needs, there is nothing more representative than a scenario where an everyday user of the system will describe what they actually need. Thus, a couple of team members wrote some representative scenarios of example users. Example scenarios can be found below:

- Carolyn is an administrator in the Building Healthy Communities program. She is in charge of tracking the volunteers progress and checking if they are doing their work correctly. It is important for the program to not only track members attendance and feedback but also to track the same for volunteers. This is vital for ensuring the programme is being run correctly and also would be an advantage in building the relationship between members and volunteers.

- Rebecca is a volunteer in the Building Healthy Communities program at the Arts group. She is currently working with a group of eight people. At the end of every session she collects attendance and feedback forms from the members. The process of reading and analysing them is very time consuming. Various handwritten forms are messy and some information are not that useful. She would prefer to have an automatic program that does all this so she can completely skip the time she spends on tracking attendance and focus on improving the Art class based on the feedback provided.

4.5 Wireframes

After creating and compiling the previously mentioned documentation, the next process was to develop wireframes. Wireframes were the final goal of the first milestone and the only requirement specified by the customers. After researching wireframing tools as a team, Balsamiq proved to be the most suitable application for our needs. Balsamiq is a very useful application, easy to use and focuses on the creation of wireframes. Before we started sketching in Balsamiq, we decided to take an initial low-fidelity route and sketch the wireframes on paper. Low-fidelity wireframes are often a good starting point as they are quick to produce and thus many designs can be produced rapidly, allowing the best from a selection to be chosen to move ahead with.

A sample screenshot of the index page which was presented to the customers can be found in Figure 1.

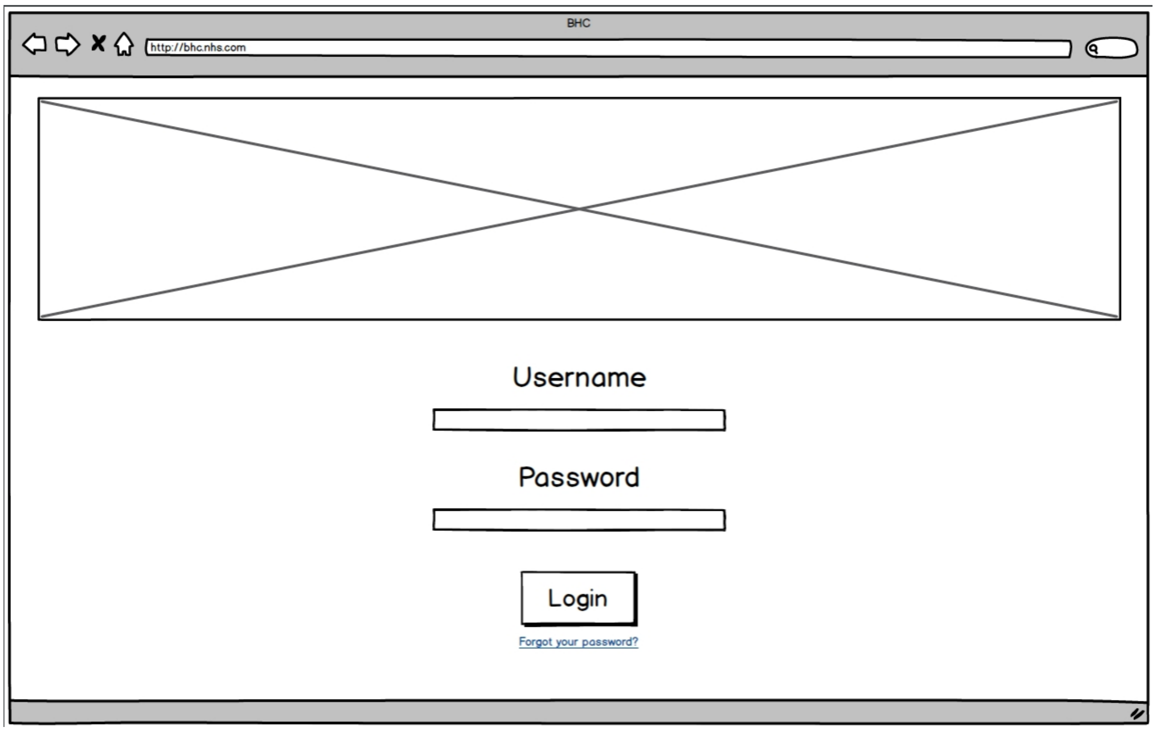


Figure 1: Wireframe of Index Page

4.6 End of First Milestone

The second customer meeting was held on the 16th November. In the meeting some example user scenarios were outlined and a brief demo was given through our interactive wireframes. The customers expressed their opinion about our current progress and what they expect to be completed by the next customer meeting.

After the meeting, the team completed a retrospective. A summarisation of the retrospective was that the team during this period of time improved its knowledge on using Gitlab as a project management tool. Moreover, a conclusion led to the fact that in order to complete a project successfully, not only are coding skills needed but also strong organisational skills, combined with strong documentation. The biggest aim was to improve the communication with the customers, since some aspects of the application were left ambiguous and misunderstood.

5 Design of the Website

5.1 Objectives

During the next iteration, after a group discussion, the objectives for the next customer meeting were stated. An ER (Entity-relationship) diagram and Component diagram were to be designed for the documentation of the project. Particular team members were assigned with new issues to do this. Not only were the diagrams created, but the process of designing and coding the application had begun.

5.2 ER Diagram

An Entity-Relationship diagram [12] is a useful tool in terms of clarifying the development of every application. Having an ER diagram represents clearly the relationships between the entities and the attributes that belong to them. The team started the development of the application only after an ER diagram was present to specify the structure of the database.

The main aspect of the diagram is that the users of the programme have different privileges depending on whether they are administrators, volunteers or members. Each group of those users is permitted to do different actions as already mentioned in the Abstract section above. For example, only the administrators are allowed to add/remove users and volunteers or to enroll/unenroll them from different initiatives. Moreover, only the administrators can apply different fundings either to specific initiatives or members based on their medical conditions. Furthermore, the volunteers have restrictions on seeing all of the members' personal information compared to the administrators. Only the volunteers are able to track members attendance and to create new session of the initiatives they are enrolled to. The members would fill

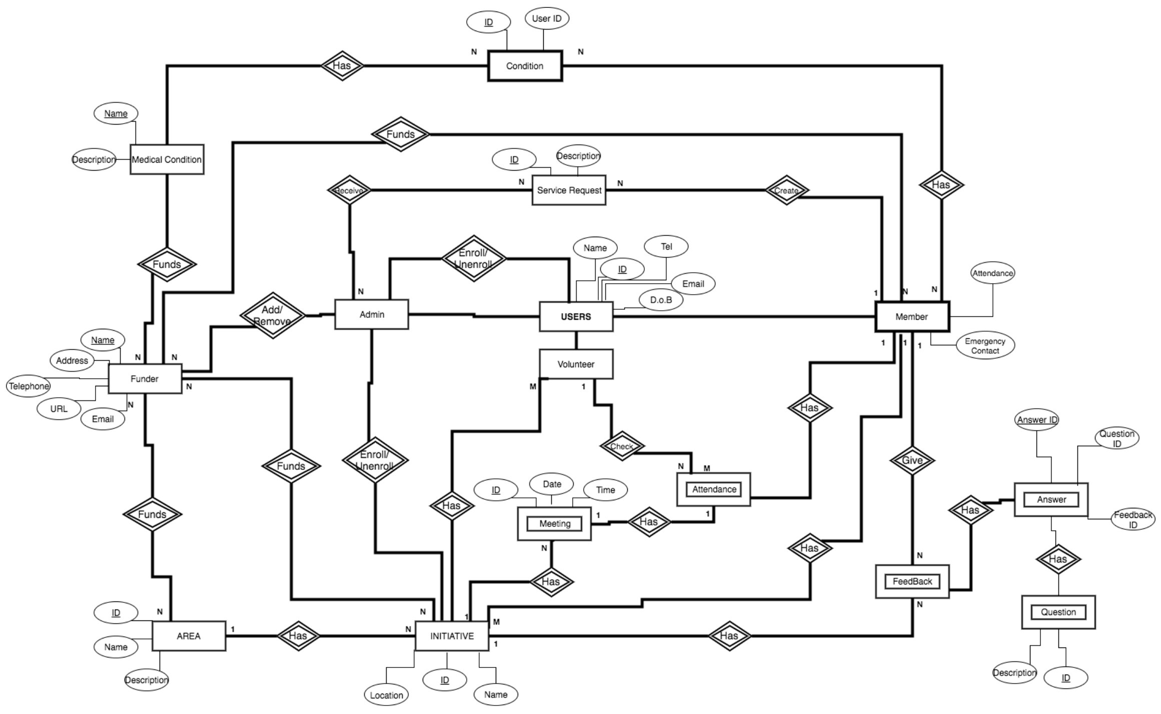


Figure 2: Entity Relationship Diagram

the questionnaire form in order to give their feedback every 3/6 months. As the customers required, both volunteers and members could not change their own personal details. The only way to do that is to send a service request to the administrators, specifying the changes they wish to apply. The full ER diagram is shown in Figure 2.

5.3 Component Architecture

Since Ruby on Rails uses a model-view-controller architecture pattern, it allows our team to develop an agile application. The model layer represents the logic of the application and the management of interactions with elements in the database. The view is the front-end of the application and the HTML files with embedded Ruby code. Controllers interact with models and views because the requests coming from the browser are processed by them. This means that the controllers fetch data from the models and pass it to the views for representation. In addition we decided to use Docker [2] to aid the implementation of our Continuous Integration. This contains everything we needed to run the application from code to libraries. It builds a Docker image of our application using Dockerfile and then runs the test suite inside a container. Using Docker has many benefits, the most important one regards the deploying of the project. All of those are illustrated in Figure 3 which demonstrates visually the component architecture of our application.

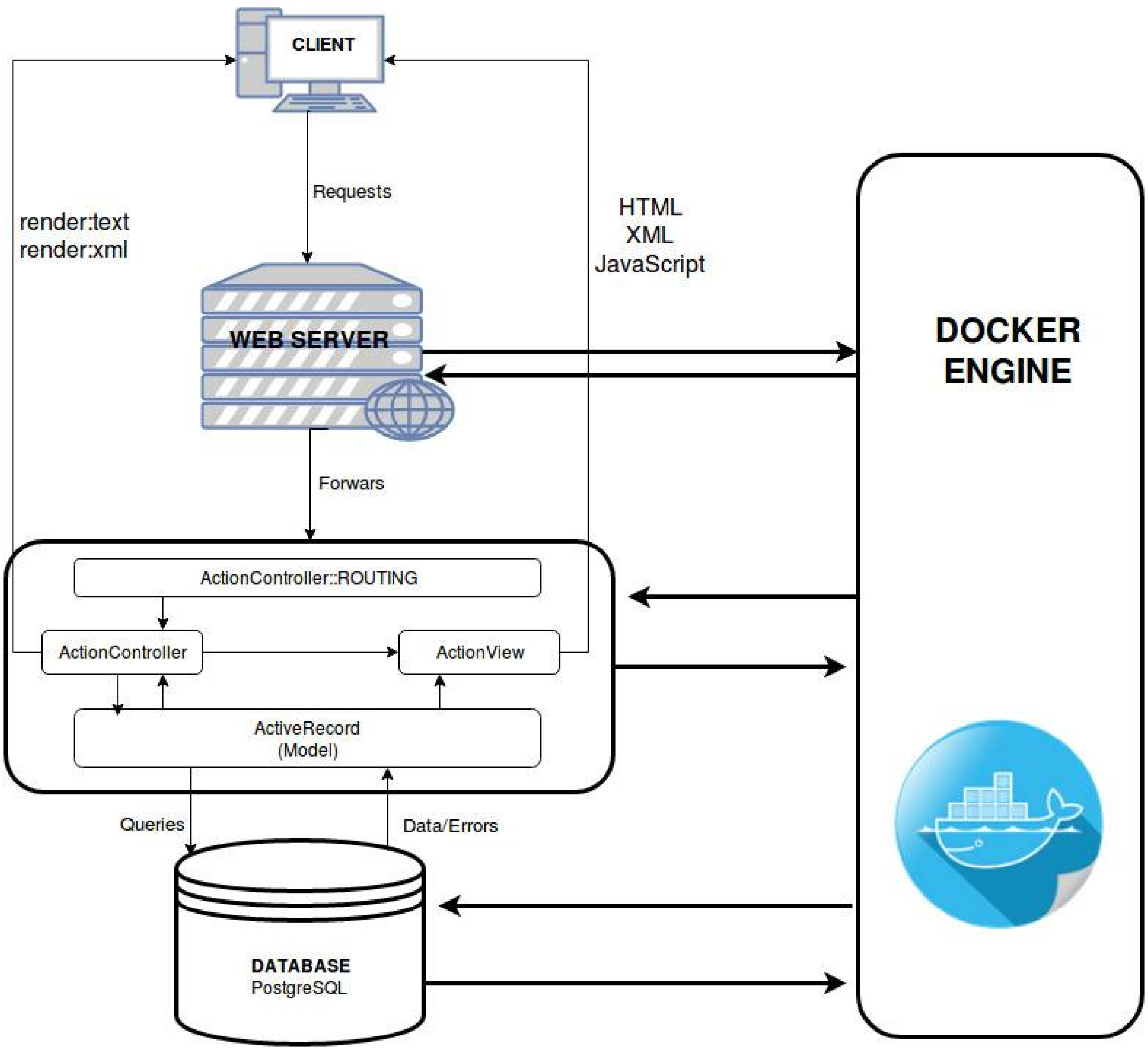


Figure 3: Component Architecture

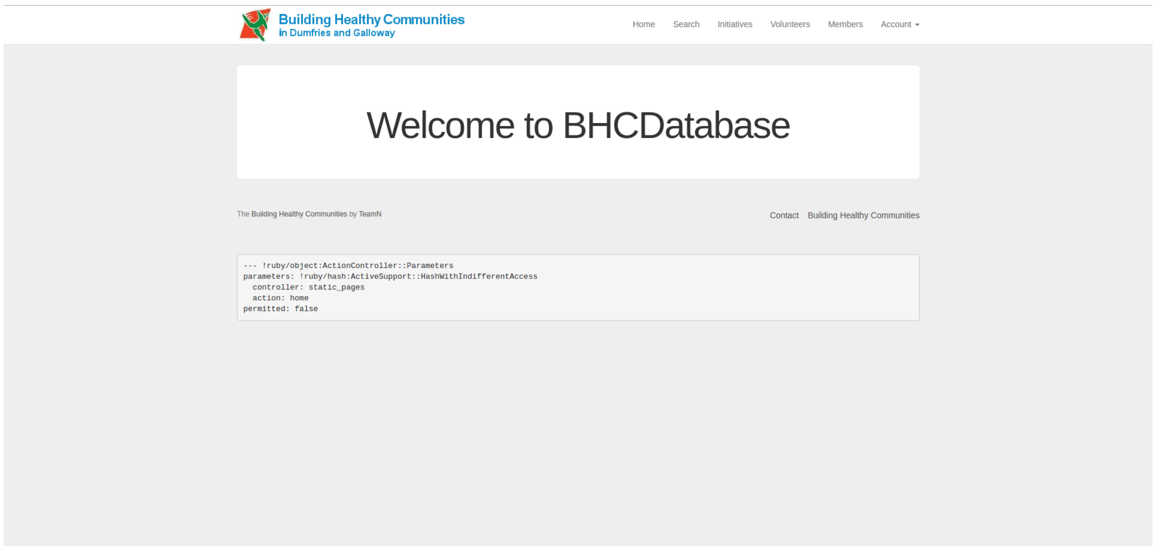
5.4 Initial Prototype

Having the basic database structure, initial prototypes of the page were created, in order to discuss potential design changes. The first version of the page can be found in Figure 4.

The initial page was very basic, not very colourful and errors containing. It was created to point out the idea of the design and to outline a vision of how the site would work. After that we added the colours following a similar theme as the blog, which the customers already had.

5.5 Final Prototype

The second design was more simplified in terms of account organisation. The customers clarified one of their requirements regarding that the members would not be



Welcome to BHCDatabase

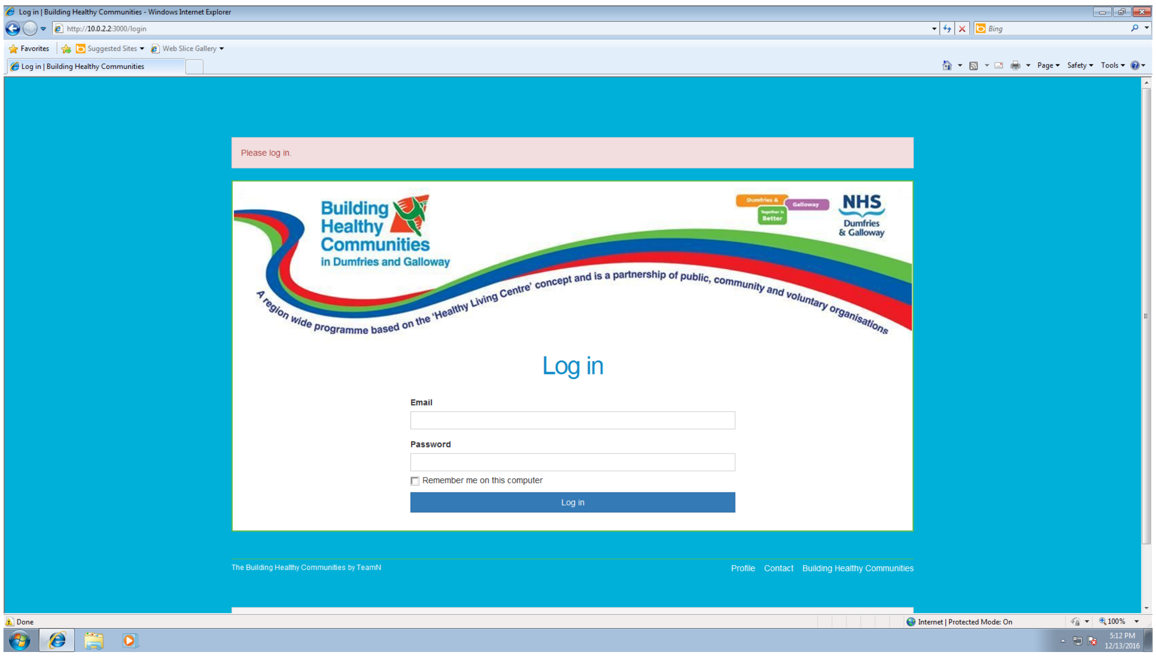
The Building Healthy Communities by TeamN

Contact Building Healthy Communities

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: static_pages
  action: home
permitted: false
```

Figure 4: Initial Prototype

able to sign up, which had the consequence of deleting this feature. Furthermore, the home page and the login page have been merged into one single page in order to provide easier access for the customers. The new prototype can be found in Figure 5.


Log in | Building Healthy Communities - Windows Internet Explorer
http://10.0.2.2:3000/login
Favorites
Suggested Sites
Web Slice Gallery
Log in | Building Healthy Communities
Page
Safety
Tools
Building Healthy Communities in Dumfries and Galloway
NHS Dumfries & Galloway
A region wide programme based on the 'Healthy Living Centre' concept and is a partnership of public, community and voluntary organisations
Email
Password
Log in
Profile
Contact
Building Healthy Communities
Done
Internet | Protected Mode: On
100%
5:12 PM
12/13/2016

Please log in.

Log in

Remember me on this computer

The Building Healthy Communities by TeamN

Figure 5: Second version of the prototype

Note that this is not the final design of the page but final prototype presented to customers in terms of structure.

5.6 End of second and third milestone

The above section was generally outlining the progress in the second and third milestone. The progress was steady and specific. By the 7th of December, a third customer meeting has held which was again mainly used for further clarifications about the software implementation and navigation to some more complete wireframes. Furthermore, after the Christmas break and before the fourth customer meeting at the 26th of January, the team managed to create an initial version of the application. The application was presented to the customers up to its current point and their opinion was taken into account. An impresion was expressed for the progress that was taken place to the website since in the previous customer meeting the back end development was not even started. Further clarifications were made about the navigation of the page, since once the customers had a visualisation of the application, it was more convenient for them to express their view for it. Additionally, they requested a new feature which was never discussed before - the addition of a funding flag which represents which funding is from which organisation and where in which groups it is applied. Furthermore, different funding's apply to a specifying group of people and different funding's apply to a specified initiatives group.

The retrospective performed by the end of the second milestone, showed some lack of communication. The team did not consider this as a major issue since there was a three week holiday gap during which everyone was focused on self studying about how to develop and deploy the application. Additionally, the team lacked on clarifying some basic features of the website with the customers. All the issues outlined on the second retrospective, they were clearly solved during the next. The team was more organised and communicative. Due to the fact that an initial stage of the application was presented on the fourth customer meeting lack of clarification of the features that was raised as an issue in the previous retrospective was know clearly solved.

6 Working on the backend development

6.1 Objectives

Before the beginning of the fourth milestone, the next objectives were outlined. Due to the fact that by the end of the current milestone a working version of the website was expected to be delivered,the main goals were to resolve opened issues that were raised from the previous retrospective and customer meeting. In the following weeks the website had all the added functionality created. The most critical issues that were raised is the newest demands that the customers were requested including the flag feature described above.

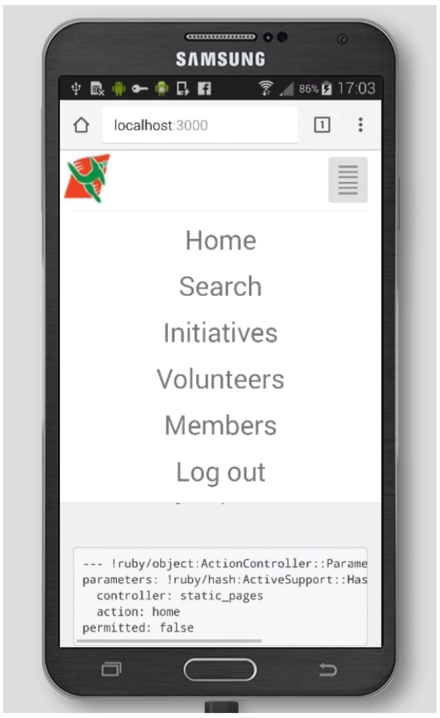
6.2 User Authentication

One of the most important factors of the development of the site was the protection of the user's information. A secure user authentication was a very crucial implementation part. By the phrase "User authentication" [11] is meant the process where the user of the page types some credential information which are compared with the information that are kept in the database, if the information are matched the user is able to access various other information based on the level access that they have on the site. Improper development of the user authentication could lead to the leak of personal information from one user to another or the ability of every user editing the database. A small research online was performed in order to explore the best practices of developing a highly secure user authentication. A highly suggested, "gem" for developing a good user authentication was the "Devise" [9]. Devise is one of the most popular user authentication tools. Some of its features are the ability of assigning multiple models in at the same time and its modulation. A user authentication tool such that provide a lot complexity and unnecessary tool that we will never user. On top of that, all these additional functionality may introduce issues that as programmers with a small experience may be unable to solve. All in all, a conclusion has been taken that a pre-built solution was not a very good idea. However, building from scratch the user authentication seemed a better solution since the complexity is controlled and every new feature is known along with any anomalies that may introduce. The user authentication was based in an authenticated book called "Ruby on Rails Tutorial (Rails 5)" written by Michael Hartl [8]. The book provides some online tutorials which the authentication is based until chapter 6. The authentication was finalised with some simple HTML commands based on our previous experience.

6.3 Browser and Mobile Compatibility

The aim for browser compatibility was quite broad. Throughout the customers' meetings it was clear that broad support of the most popular browsers is essential and that the website should function on both iOS and Android (Running in both tablet and mobile form).

A huge issue that was raised at that point, was the browser compatibility and more specifically the support of Internet Explorers version 7 and before. The earliest Internet Explorer version we aim to support is IE 8. This is because IE8 comes as standard with Windows 7. Although possible, we have chosen not to support IE 7 due to it coming with Windows Vista. Vista has 'Extended support until 11 April 2017.' After all, the final decision not to support IE7, as to do so would be to support a soon to be dangerously out of date operating system. An important point to note is that Bootstrap only officially supports Internet Explorer 8-11. Another requirement that was provided for the system is the mobile compatibility. Thus, a good mobile compatibility is demanded. Although, all the different types of users support mobile compatibility, the one type that it is guaranteed to fully work is the volunteer's side which was actually the only site that was demanded. The volunteer side compatibility was highly important since a volunteer must be able to quickly take attendance from his/her mobile phone. Moreover, some screenshots have been taken from the initial prototypes of the page showing the mobile compatibility. An example screenshot can be found in Figure 6.



```
--- !ruby/object:ActionController::Parame
parameters: !ruby/hash:ActiveSupport::Has
  controller: static_pages
  action: home
permitted: false
```

Figure 6: Mobile Compatibility Prototype

6.4 Testing

An integral step in developing an application is constructing a test suite that is as complete and as robust as possible. It is vital that testing begin early in project development, as the costs required to implement defect tests late in the process can be extremely costly, as evidenced in Figure 7. With this is mind, testing began almost immediately, any time a feature was to be implemented, the corresponding tests were to be written before the issue could be closed. The tests were constructed using Rail's built-in testing framework. Upon creation of an app in Rails, a 'test' directory is created which holds many other directories intended to contain different testing groups. For example, the 'controllers' directory holds all tests for the controllers and the 'integration' directory holds all tests that involve many controllers interacting together. When creating a model or a controller, the skeleton code for the tests is automatically generated and placed within the corresponding directory. 'Fixtures' are a way of creating and organising test data. On top of using Rail's testing, the 'SimpleCov' gem was introduced. SimpleCov is a code coverage analysis tool which amongst many other things, displays coverage statistics upon test execution. It also generates a 'coverage' folder which contains interactive documentation displaying the code coverage of each file, which lines were hit, and how many times. The application's tests were designed in an attempt to ensure correct functionality in as many target areas as possible, this includes: controller execution, model validations, uniqueness validations, page restriction validations, etc. Tests were not designed to achieve a high test coverage. Whilst lines covered does give a good indication of which areas of the project need attention, it should not be used as a defining tool for 'good' test coverage. It is perfectly possible to have extremely high test coverage, with relatively meaningless, superficial tests.

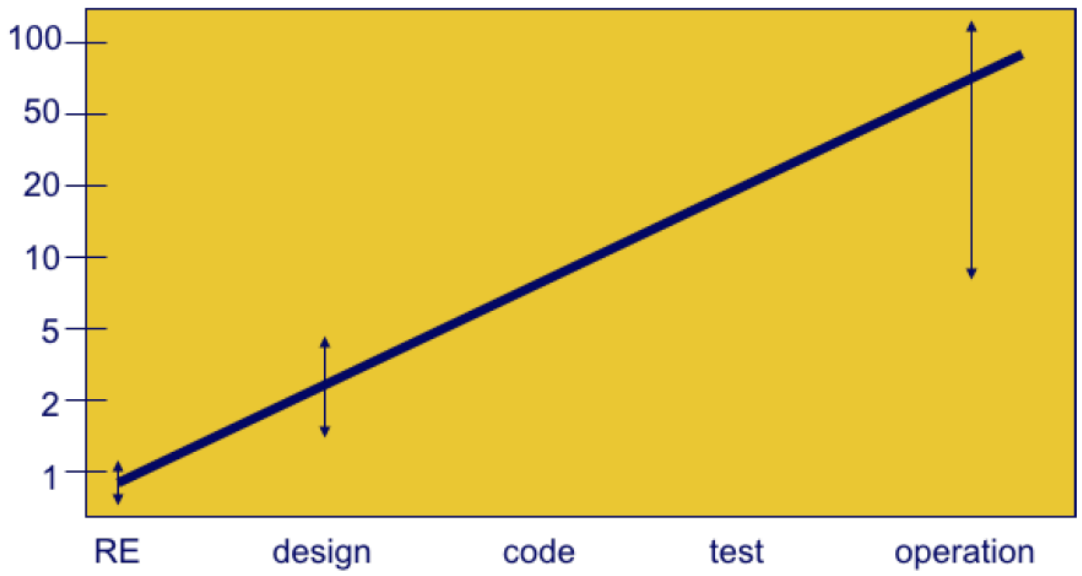


Figure 7: Increasing rate of costs in different software development stages

Gitlab's CI provides tests in every build using the *.gitlab-ci.yml* file. The file created for the specific software is using all the three stages provided: build, test and deploy. Then we have deployed the *Runners*. A runner picks up a build in the proposed project. Then, every time a commit or merge is triggered to the repository, tests are running to check whether there are issues. Furthermore, the team developed their own tests using mutants. The program has an impressive percentage of testing coverage reaching an approximate of 86.6%.

6.5 End of fourth milestone

The fourth milestone was successfully completed with all its major goals succeeded. The website was successfully presented to the customers on the customer meeting performed on the 23rd of February. Although some minor features were not yet implemented, a fully working version was delivered and in the following month all the features were progressively added. During the navigation process, information were provided to the customers about how to access the deployed website in all three different types of users (administrators, volunteers and service users). The initial online deployment was in heroku. Unfortunately, the customers made more last minute demands to the team. This additions were added to the next milestone. Finally, a questionnaire was asked to be answered based on their navigation experience which was later used in the next milestone for the application evaluation.

7 User Documentation

A very essential product that needed to be developed was the user documentation. Due to the fact that this is a university course, the software life cycle would not be strictly followed. For example, no future maintenance would take place. A well defined user documentation had to be produced. To put more depth, the documentation mainly explains how to navigate in the program and perform simple tasks such as adding a new user or deleting an old one. The documentation was given in the form of a small book. It is divided in three different parts, the administrator section, the volunteer and the member section. Three sections, each referring to one of the three different authentication types. The development of the documentation was based on a single principle, develop it such as every aspect, every word will be possibly read by a person who has no idea how to navigate through the site so he/she can understand and successfully complete their task. Due to the fact that this is a fund based program, many different people will have administrate the website through the years. Thus, a well understood user documentation is obligatory. Figure 8 contains a screenshot of the user manual.

Focusing on Figure 8 which demonstrates an example page of the user documentation, we can observe that both text and screenshots are included, leading to a conclusion that the navigation is interactive, enabling the user to instantly locate the section that answers their question.

8 Formative Application Evaluation

The Formative Application Evaluation is the stage when a developer receives objective feedback, useful comments and suggestions about their product. After the successful early delivering of the application to the customers, the next step was creating a questionnaire. In combination with the simple user guide, an evaluation was performed. The evaluation was only performed from the customers. Due to the fact that the team had already discussed the design issues with them, the evaluation and questionnaire were mainly focused in the navigation and understanding of the page. On the fourth customer meeting, the team leader performed a sample navigation to the page, explaining to the customers how the pages work. Additionally, a user guide and a simple questionnaire were given. The goal of the questionnaire was to study whether the customers are able to navigate to the page and to what extent they are able to solve their issues using the user guide. The questionnaires were collected and studied. The questionnaire was made via simple ”Google Form”. Some questions that the customers were asked are:

- In general do you like the look and the feel of the website?

usermanual.jpg

Figure 8: User Manual

- Have you found trouble performing search queries in the website?
- Have you faced any difficulties in adding new sessions, users, questions and so on?

As a consequence of having a small amount of responses, the results were rather qualitative than quantitative. The responses of the questionnaire showed no major issues except of the fact that they have asked for more features that again were not discussed in previous meetings. The team focused on solving the small navigation issues that the customers identified first and in a following time managed to add the new features.

9 Challenges

Developing a site from scratch, always creates some major challenges. The second and third retrospective were extremely representative in the matter of stating major challenges and proposing possible solutions. A very common issue that is experienced during the software development process is the development of test cases.Tests are one of the most time consuming task since every small aspect of an application needs to be tested. Surprisingly, the tests created for the proposed software had a very high coverage of an approximate 86.6% . Leading to the fact that testing was a challenge only in terms of time consume.

The organisation of the database was another big challenge that the team have encountered. Although the proposed database can be characterised by its simplicity, retrieving the information was sometimes tricky since a member of a team could be a volunteer in another team and a general administrator. Possible authentication issues can be raised questioning the fact of where that specific user is allowed to alternate information and where is not.

As in every project developing process, programmers will face some last-minute demands from the customers. One of these demands were the creation of some type of flag pointing which members or groups are funded from specific organisations. The current demand raised some issues in in the database development since this is an addition very specific information regarding every member of every group. The feature was flagged as an "extra feature" which would be developed only after the rest of the software is properly developed. The decision of taking that action was stated clearly to the customers from the very first day that it was suggested. The feature caused last minute additions to the ER diagram. The feature was successfully created and added to the website after the correct testing.

Having an agile oriented development sometimes leads to unexpected demands. This was the case of changing the deadline for delivering the website to a month before the due date. The customers suggested an early delivery of the website for testing purposes and feedback submission. The following action raised some challenges since the overall coding time was only one and a half months. The team managed to deliver the software on time but without some minor features which were progressively added in the software.

Last but not least was the creation of a User Manual. Developing a user manual is useful not only for the current customers but also for any other future users of the application. Any user can have a simple and brief guidance how to navigate through the application and to take as much advantage of it as they can.

10 Final Customer Day

On the 22nd of March the final demonstration of the product was performed. The team had to present the website to the customers and lecturers.

The final state of the product was a fully working software with some added functionality. As previously stated, the customers were having new requirements in the ongoing process of meetings and communication. All of the major and important features were successfully delivered on time. Considering the fact that a major addition had to be done due to a mistake performed by the customers just eight days before the final customer day, not all the extra functionality that the customers required was added. The customers were informed of the time constraints, thus a couple of minor features were just a suggestion and not a demand resulting good relationships between the customers and the team.

During the demo the teams were graded on the following factors:

- Clear explanation of motivation for project
- Outline of key achievements and product features
- Explanation of technical features/challenges
- Live demonstration
- Proposals for future work
- Effective use of visual aids

The demonstration lasted approximately 20 minutes and it was a combination of a brief presentation about the project and a live demonstration. The demonstration was consisted of three different parts:

- Navigation through the administrator's side
- Navigation through the volunteer's side
- Navigation through the service user's side

The demonstration ended with questions from the audience and the customers. An overall summary is that the customers were satisfied of the progress and the lecturer was impressed with the high test coverage that the team managed to create.

After the demonstration had finished, the customers expressed their gratitude to us for the work we had done. Upon request the deployment version was updated including the latest updates in order to enable the customers to test the application themselves.

11 Conclusion

Taking everything into account, designing and developing a software application was everything but an easy process. Considering the fact that the team consisted of six novice students with no experience at all in using agile software programming, the presented result was more than satisfactory. Having real customers with high demands was definitely a huge challenge. By the adaptation of several techniques and methodologies every difficulty was successfully confronted.

Agile programming proved itself that is a very good practice to be followed from novice programmers. Pair programming played a critical role in terms of identifying bugs and errors in the code. By the early weeks, the team lacked of organisation but reviewed some valuable Agile programming rules. Communication was also identified to be extremely important, since many major issues that could not be solve from one team member, were solved from another one. Although, at some point we faced a few communication issues, they were not considered important since the team was already in a very good position with the project. Putting in practice so many technologies such as Ruby and Docker was undoubtedly one of the hardest tasks that the team had to undergo. Thus, during the Christmas break a wiki page was created which pointed out many useful resources for self-studying and self-practicing.

One of the best practices that were kept, was to strictly follow the university guidance and create test cases for each new aspect we created in the application. The test cases identified many bugs and errors which the team managed to successfully solve. Many other teams did not kept that practice and as a result many serious issues were not identified in advance resulting to the inability of delivering their software within the time constraints.

Furthermore, branching was another important practice that we kept. The team followed a strict plan of creating a new branch for every issue and resolving that issue in the branch. After everything are checked and tested, a merge request is created which only two of the six team members can accept after inspecting the code for conflicts. Thus, the master branch was always in a fully working condition with the testing percentage to be from the very beginning at 80-90%. Performing frequent commits also helped the testing to be constantly kept high.

Delivering a software does not only demand writing code but also creating a project meeting the customers' views and expectations. Despite the fact that the clients were characterised by politeness and cooperation, they were very high demanding. Until the very last week new features were requested. After endless hours of re-factoring the system the team managed to deliver as many features as they were able. The team learned that even with many demandings, you have to always tell the truth to the clients. When the team decides that due to time constraints a feature cannot be implemented or even when the team does not know how to implement a feature, the customers must be contacted and the issue must be discussed. Promising features that will never been delivered will result to a mistrustful relationship between the clients and the developing team. This experiences helped the team to realise that requirements negotiation is an important part in the software development process.

On the 24th of February the team managed to deliver a fully working website to the customers. Furthermore, on the 22nd of March, the final demonstration date, an improved website with added functionality was delivered to its final state.

To conclude, working with real clients, gave to each member the opportunity to develop a minor working experience, to realise how the future is going to be and develop the proper bases needed for working in a real company. Every student discovered in more depth their abilities and weakness and most importantly learned that a customer has not only a vision but also important requirements that need to be followed step by step. Everyone managed to develop professionalism were you have to constantly communicate with inexperienced customers and listen with respect their demands, even when they are impossible to be developed. Backtracking the experience, everyone managed to develop new skill that will be later used in their software engineering career.

References

[1] Faq: General. https://docs.djangoproject.com/en/1.7/faq/general/. [Online; accessed 14 December 2016].

[2] What is docker. https://www.docker.com/what-docker. [Online; accessed 26 January 2017].

[3] Continuous Integration (CI). https://docs.gitlab.com/ce/ci/quick_start/README.html, 2017. [Online; accessed 19 January 2017].

[4] Scott W. Ambler. User stories. http://agilemodeling.com/artifacts/userStory.htm, 2005-2014. [Online; accessed 26 December 2016].

[5] Bozhidar Batsov. A community-driven ruby coding style guide. https://github.com/bbatsov/ruby-style-guide, 11 Semptember 2011. [Online; accessed 24 February 2017].

[6] Bozhidar Batsov. A community-driven ruby on rails 4 style guide. https://github.com/bbatsov/rails-style-guide, 25 Semptember 2011. [Online; accessed 24 February 2017].

[7] Karen Carias. Simple words for a gitlab newbie. https://about.gitlab.com/2015/05/18/simple-words-for-a-gitlab-newbie/, 18 May 2015. [Online; accessed 14 December 2016].

[8] Michael Hartl. *Ruby on Rails Tutorial (Rails 5)*. Addison Wesley; 4 edition, 7 November 2016. [Online; accessed 15 January 2017].

[9] Carlos Antnio Jos Valim. Devise. https://github.com/plataformatec/devise, 2009. [Online; accessed 15 January 2017].

[10] Daniel Kehoe. What is ruby on rails? http://railsapps.github.io/what-is-ruby-rails.html, 11 October 2013. [Online; accessed 14 December 2016].

[11] Margaret Rouse. User authentication. http://searchsecurity.techtarget.com/definition/authentication, February 2015. [Online; accessed 15 January 2017].

[12] LLC SmartDraw. Er diagram. https://www.smartdraw.com/entity-relationship-diagram/, 1994-2017. [Online; accessed 12 January 2017].

[13] Nick Toscano. Comparison of django and rails. http://www.skilledup.com/articles/battle-frameworks-django-vs-rails, 03 October 2015. [Online; accessed 14 December 2016].