University of Glasgow | School of Computing Science

Level 3 Project Case Study Dissertation

Building Healthy Communities Database

David Robertson
Maria Papadopoulou
David Andrew Brown
Kiril Ivov Mihaylov
Christopher Sean Harris
Jaklin Yordanova

March 2, 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 members progress. They must inform administrators not only about the progress of the members but also about the whole process. Currently all this is being performed by hand. Administrators must manually input the data into a database, and also search the database using only 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 system that accepts the current database, and allows it to be modified and extended. The system has the following features:

- Registering new members and deleting those who leave the program
- Keeping track of members' attendance
- Keeping track of the progress of both members and volunteers
- Deleting and adding initiatives
- Searching the database for particular parameters and organising the information in an easy to read manner.

Education Use Consent

1 Introduction

Software engineering

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, and a system which allows for the secure, rapid organisation of data helps to ensure this aim can be easily met.

Through the following pages we will explain in more depth how we designed, created and use the software that we have described above using not only our previous knowledge, but also the knowledge that we have 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 we took in creating the software. From the 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.

Our dissertation is being divided..

2 Case Study Background

2.1 Customer Organisation and Background

Building Healthy Communities is an NHS 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 operates regionwide to improve the health, wellbeing 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, and four local Area Partnerships operating across the region. These Area Partnerships co-ordinate and take collective action by creating intiatives 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 leath 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 members of the community to feel even more 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 initiative. For the duration of the project, we were in contact with two of these administrators. They attended each customer meeting, and were instrumental in the shaping of our product, providing requirements and feedback. We also had some incidental contact with another NHS employee from the same area running a different project, whose advice was of particular use when designing the database and deciding which informtaion 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 from the members, volunteers and administrators of the programme. All of those different groups of users have different level of access to the Database. The members 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 initiative they are assigned to. Administrators have the access and rights to do everything : they can see all the information about both volunteers and members, the can add or delete an initiative and review feedback and attendance as well. Since we are working in a Agile Team Organisation, the initial requirements could be slightly changed in the following customer meetings as explained later in the dissertation.

3 Agile programming and Team organisation

3.1 Team organisation

In our initial meetings, every team member pointed out their strong and weak parts in order to divide the work. Due to the fact that we were required to create an application and database from scratch, the workload was huge. As a result of that, everyone had to work in the "backend". Inevitable was for specific team members who were in a higher "backend/coding" level to execute some more advanced parts of the coding. The team discussed various means of communication - we decided to avoid using common social media sites and to use the "slack" application, 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. Particularly, 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. Also, they are based on focusing on the code instead of the design which is suitable for our situation, since we were asked to develop a page from scratch requiring a database. Our project is based on one of the agile methods called "Extreme programming". This was the most suitable decision, due to the fact that we are novice programmers. One of the most useful features that Extreme Programming offers is called "pair programming" , which is highly occureing in our practices. As students, we lack of experience but when one student combine their knowledge with another, they can create spectacular results. Furthermore, user stories and prototyping helped us a lot to understand what really our application needed. Another important feature of Extreme Programming is the automated testing because as stated before, we have little experience.

3.3 Technologies Used

As soon as we agreed on the methods we were going to use, we had to decide the technologies that we needed to use regarding project management. The university moodle page, suggested a series of technologies to use for handling our project such as "Ant and Ivy", "Jenkins", "Apache" etc. After a small group meeting and a discussion with the supervisors about what we are allowed to use and what we are not, we decided to use Gitlab for our project management and repository. Gitlab provides every technology that is needed to handle our project repository, the permissions that each member has as well as an amazing GUI that organises the "wiki" page and the "issues" page and so on. Using Gitlab, was very helpful, since we had all our due dates clearly displayed, issues and requests organised as milestones with the appropriate labels. The most useful feature that Gitlab provided to us was the Continuous Integration (CI) feature. CI is a tool that enables all the repository users to merge their work to a local repository. This was a huge organisational feature, since everyone could interact in the project code. Every team member, could assign a ticket to themselves or even to other team members, as a result we had a highly organised page where we could track the workload that every member has done. This was very helpful, in terms of assigning work to members that have not contribute as much as the others in a specific week and assign less to members that already have done more work than needed. Furthermore, we were making our issues very descriptive - using checkboxes, weight, time estimating and different labels. This was very useful in terms of tracking progress. Another feature that we were using was creating a separate branch from the appropriate issues. In that way we were testing new features and bug fixes without pushing to the Master branch. Only after the problem was solved, a Merge Request was created followed by closing that particular issue and deleting the temporary branch.

After this discussion, the meeting was not over. Therefore, we needed to decide which technologies would be used for implementing the project itself. We had to choose which web application framework to use. The first option we discussed was Django, due to the fact that every team member had previous experience with Django from our second year web application development course. Another option a team member suggested was Rails as more powerful tool with better documentation. To put more depth in it, Rails is a web application framework with "model-view-controller" software design pattern, not "model-view-template" like Django, which we were familiar with. None of the six members in our team coded on Rails before, so we have postponed our meeting and created an issue in the Gitlab - every member in the team had to study Rails in their time. In the next meeting, we voted and agreed to use Rails. Rails provides a massive help in the testing process. Moreover, the biggest factor that persuaded the team to work with Rails was the automation of many tasks and the "Gems" documentation that provides. "RubyGems" is a package manager. Using a tool called "Gem" in your terminal you can install and work with different libraries. Gems, are previously tested libraries, which provided security and saved a lot of valuable time. Reusing previously tested tools is always highly suggested in the world of web application frameworks. Since Rails, as previously stated, is based in "model-view-controller" we needed to find a model, an application to manage our newly created database. PostgreSQL was our choice because the database would be small with just a couple of thousand text records. Furthermore, PostgreSQL is a self-contained, free to use object-relational database management system.

3.4 Retrospective

All these process, should be summarised in a page. Here is when we decided to use retrospectives. Retrospectives are a small summary of a past situation. In our course, retrospective is used to summarise our actions which were performed in a specific amount of time for progressing our project. Our retrospectives were created in "Trello". This was very helpful because we stated what we did good, what we were lacked of in knowledge and what we could improve. To be more specific, we decided to use 4L's: Liked, Lacked, Learned, Longed for. We also decided to create and fill a new retrospective every time we had a customer's meeting, which was the end of every Milestone as well. Every team member was participating in the retrospective and filling at least one thing on the board. After each one of us have done it, we were gathering for team meeting followed by a discussion and outlining major issues that were identified through Trello. This was the most efficient way of observing problems and solving them as a team. Furthermore, we not only had time to improve, but also to learn new technologies and discover in more depth the needed technologies for accomplishing our tasks due for the next meeting.

3.5 Team roles

*** under construction, no clearly defined roles yet ***

4 Documentation

To begin with, we have divided our project to **number** different milestones which lasted a period between every customer meeting. In the following lines I will discuss the process that was taken in every milestone.

4.1 Objectives

On the first customer meeting, the day that we were assigned our project, the customers underlined their desired program which I have stated above. By the end of the meeting, we asked the customers what the wanted to see in the next meeting, their answer was: "I guess, some sketches !". A new milestone was created with our next goal to be some basic documentation and the creation of some wireframes.

4.2 Requirements Gathering

During the meeting, we have managed to outline some basic requirements. We have had developed a wiki page containing an outline of the meeting. We have used this page to identify the functional and non-functional requirements. Having an on-going contact with the customers, who stated some useful points we ended up with the final version of the functional and non-functional requirements.

The final version of the functional requirements is:

- 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 groups/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.)

Identifying the requirements was an important step which enabled us to put in action some examples. The team got divided into groups and documented various things such us user stories, user scenarios and wireframes.

4.3 User Stories

User stories play a critical role in the agile methods that we have chosen to follow. A user story formally describes the state of the application. The basic functions that a potential user expects it to do. We have managed to identify functional and non-functional user stories. User stories helped us to put in action the program and to fully understand how it will work.

A couple of highly representative functional user stories 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

Thus, we had to record some non-functional user stories so that we can give practical examples about how the system will work. Some non-functional user stories 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 users and system needs, there is nothing textual more representative than a scenario where an everyday user of the system will describe what they actually needs Thus, a couple of team members wrote some representative scenarios of example users. Example scenarios can be find bellow:

- 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 volunteers one. This is vital for keeping the program works 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. In the end of every session she collects attendance and feedback forms from the members. The process of reading and analyzing 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 collecting all this useful information, the team was ready to start developing some wireframes. Wireframes were the final goal of this milestone. Without completing them would be like we have not worked at all. Having a small research during the meeting, we decided to use balsamiq for the creation of the wireframes. Balsamiq is a very useful application, easy to use and focus on the creation of the wireframes. Before we started sketching in the balsamiq, the team decided to go pen and paper to sketch our ideas. Pen and paper sketches are always the best plan, they give you the freedom to design exactly what you need.

The formal wireframes were ready for the presentation. A screenshot of the index page can be found in Figure 1.

4.6 End of first milestone

The second customer meeting was held on the 16th November. In the meeting we have outlined some examples of user scenarios and a brief tour through our interactive wireframes. An amount of time was available for the customers in order to express their opinion about our current progress and what do they expect from the team to be done until the next meeting.

An overall satisfaction was kept from both the team and the customers. After the meeting, the team filled a retrospective. An outline of the retrospective was that the

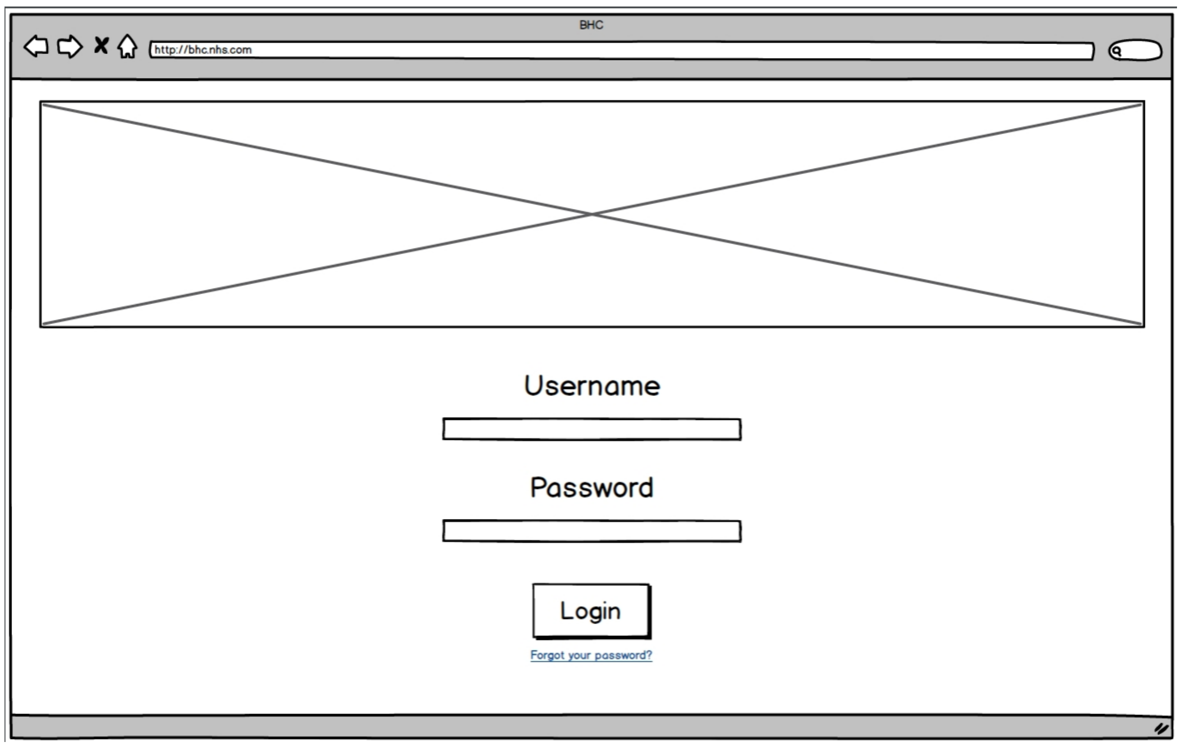


Figure 1: Given structure of WireFrame

team during this period of time improved its knowledge on using an application such us "Gitlab" as a project management tool. Moreover, we learned that in order to complete the project successfully, we need not only coding skills but strongly organisational skills with the huge amount of the documentation. The team was also aiming to improve the communication with the customers, since we did not understand quite well all the aspects of the application in the first place. Therefore,the second customer meeting was very helpful on doing it so.

5 dont know

5.1 Objectives

During the next iteration, the team held a group discussion to decide the objectives for the next customer meeting. Some pieces of documentation were still missing, such as the ER (Entity-relationship) and Component diagrams. The team assigned to particular team members the diagrams as new issues. The team had not only create the diagrams but also start designing and coding the application. Some team members also have been assigned to start the actual application.

5.2 ER Diagram

An entity-relationship diagram plays a critical role in the development of an application. In fact, the team was unable to start developing the application without such a diagram. An ER diagram presents the relationships between the entities in a diagram, which is why without the diagram the team was unable to have a clear view of the structure of the database. Having an ER diagram presents clearly the relationships between the entities, the attributes that consist them and which entities are depended on others.

Our ER diagram is expressed in figure 2.

To give a brief explanation of the diagram: The main aspect of the diagram is the initiatives, the courses that the program provides which are specified with a specific name, id and location. The initiatives are provided in specific areas that have and ID. The buildings provide some sessions such as art class. The sessions have an id, a date and time. In every session attendance is recorded from the members, who are able to give a feedback. The members must specify some information in order to get

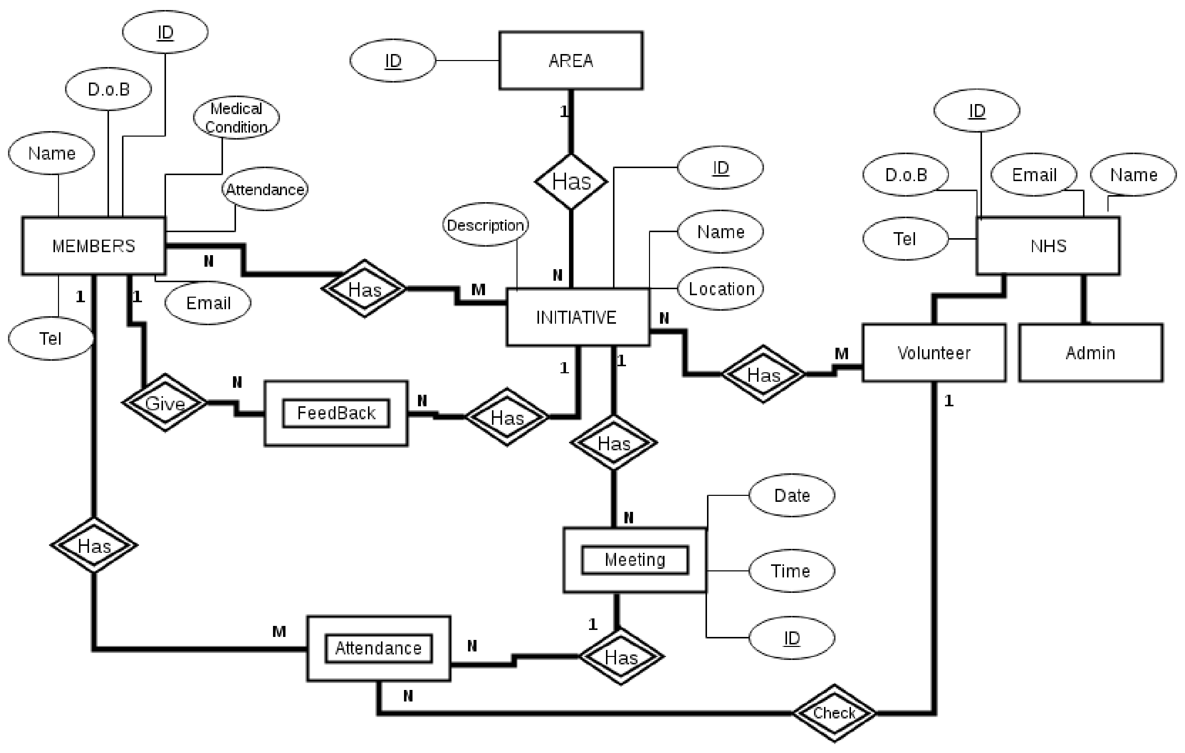


Figure 2: Entity Relationship Diagram

tracked such us date of birth, name, email etc. The NHS team is consisted of the administrators who can control almost everything in the site and the volunteers who have a more limited access and they are responsible for tracking the attendance of the members in every class. The NHS team also have some basic retrieval information in the system, similar to the members.

5.3 Component Architecture

Since Ruby on Rails uses Model-View-Controller architecture pattern, it allows our team to develop 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 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.

5.4 Initial Prototype

Having the basic database structure, the team created some initial prototypes of the page, in order to discuss potential design changes. The first version of the page can be found in Figure 3.

The initial page is very basic, not very colourful and errors are contained. It was created to point out the idea of the design and to outline a vision of how the site would work. The team decided to add the colours and the feel of the blog that the program already had.

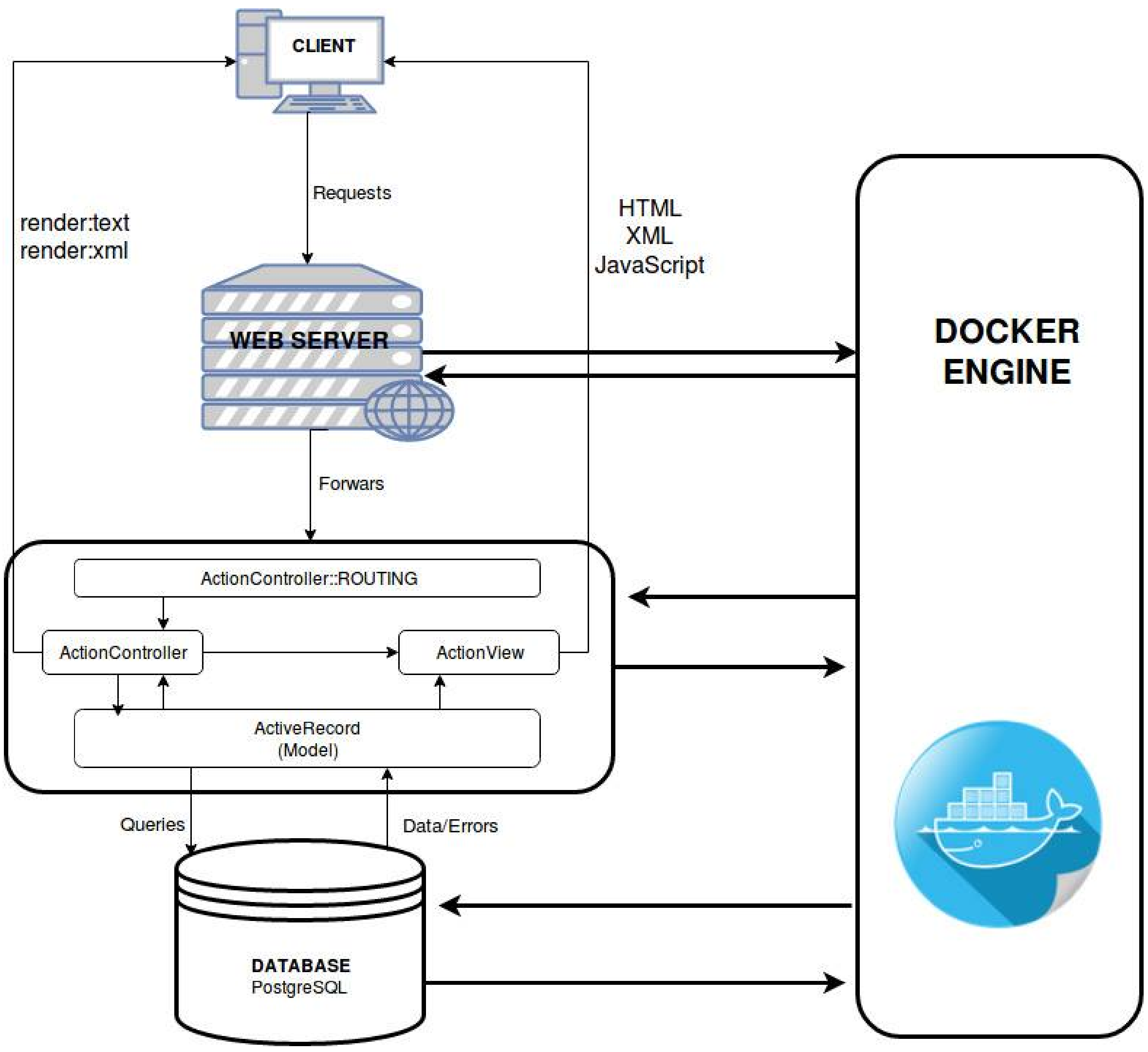
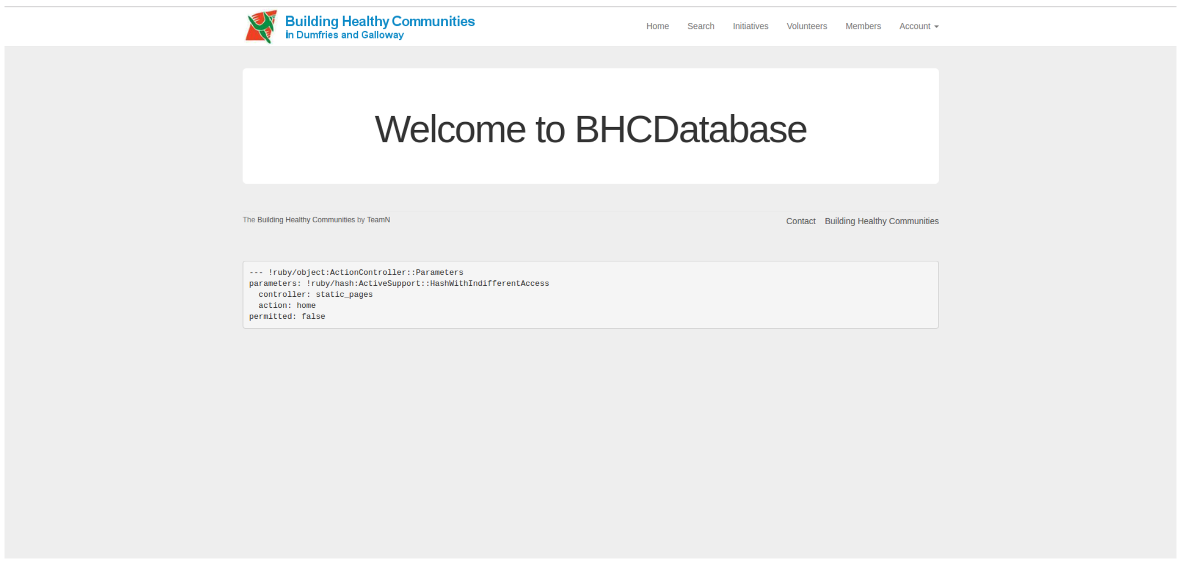


Figure 3: Component Architecturem

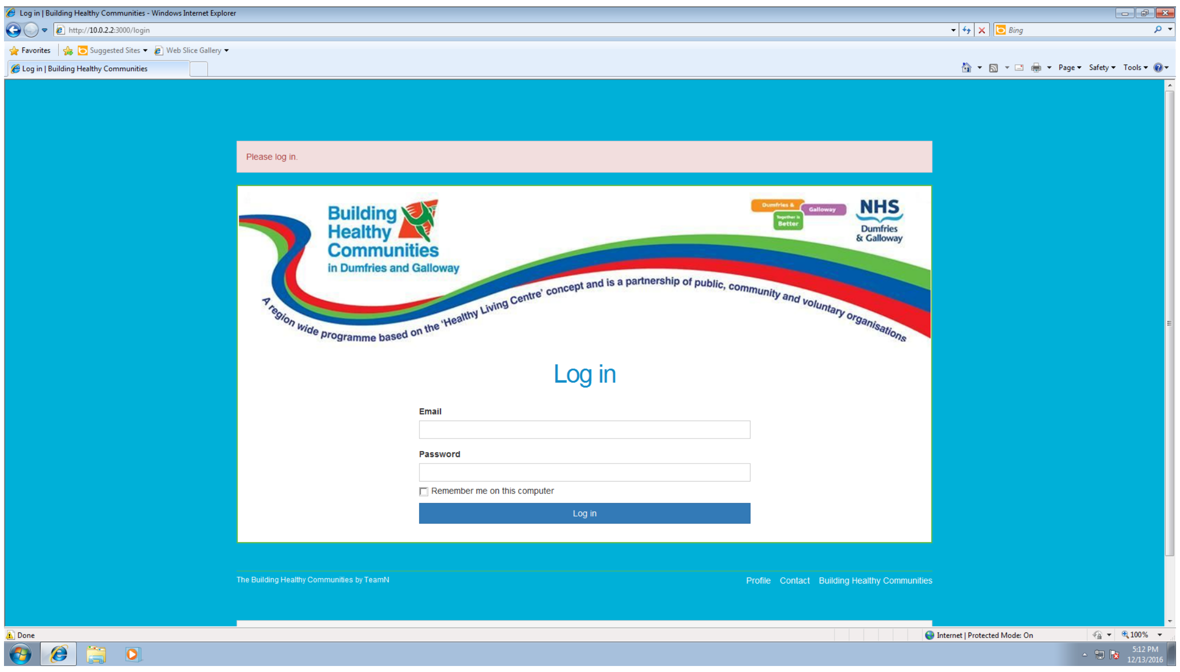


Welcome to BHCDatabase

The Building Healthy Communities by TeamN

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

Figure 4: Initial Prototype


Please log in.

Log in | Building Healthy Communities - Windows Internet Explorer
http://10.0.2.2:3000/login
Building Healthy Communities in Dumfries and Galloway
A region wide programme based on the 'Healthy Living Centre' concept and is a partnership of public, community and voluntary organisations
NHS Dumfries & Galloway
Log in
Email
Password
Remember me on this computer
Log in
The Building Healthy Communities by TeamN
Profile
Contact
Building Healthy Communities

Figure 5: Second version of the prototype

5.5 Final Prototype

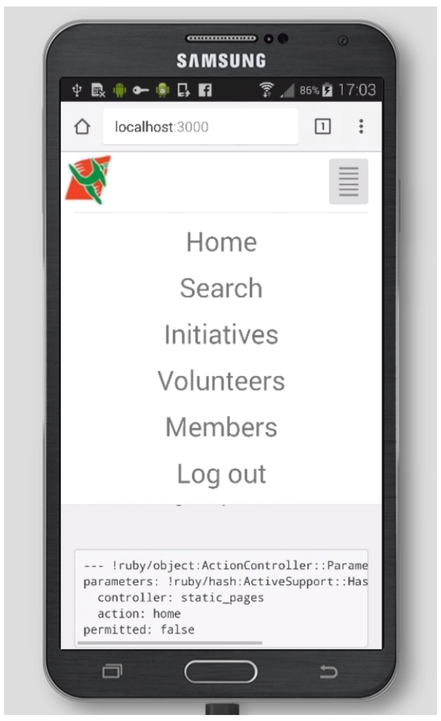
The second design was more simplified in terms of account organisation. The customers clarified that the members would not be able to sign up, which has as a consequence to deleting this feature. Furthermore, the home page and the login page have been merged, so that a simpler view of the site is provided. The new prototype can be found in Figure 4.

5.6 Component Diagram

6 Working on the backend development

6.1 User Authentication

Maybe the most important factor of the development of the site was the the protection of the user information. A secure user authentication was a very crucial development part. By the phrase "User authentication" we mean 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 he/she has 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. The team performed a small research online 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". 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 a very good book called "Ruby on Rails Tutorial (Rails 5)" written by Michael Hartl. The book provides some online tutorials which the authentication is based until chapter 6. The team finalised the authentication with some simple html commands based on previous experience.



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

Figure 6: Mobile Compatibility Prototype

6.2 Browser and Mobile Compatibility

Our aim for browser compatibility is quite broad. From meetings with the customers, we have gathered the information that the support for the most popular browsers is essential, and that the website should function on iOS and Android (Running in both tablet and mobile form).

A huge issue 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, we have taken the 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.

An important amount of the user's site, will only have as a mean of access to the site their mobile phones. Thus, a good mobile compatibility is demanded. Moreover, some screenshots have been taken from the initial prototypes of the page showing the mobile compatibility.

6.3 Testing

It is well known, that the development of any type of application is incomplete without the proper testings. Testing is a vital procedure in the process because it detects many defects and errors. Since the first day that we started coding, every section that we completed, we had also provided the proper testings. Providing testing in advance reduces the time and the cost needed to identify and correct defects. The graph provided in figure 6 displays the sharp increase of the costs of providing the defect tests very late in the process. The graph is from the level two "Software Engineering" course lecture notes.

Gitlab's CI provides tests in every build using the *.gitlab-ci.yml* file. Our file was using all the three stages provided: build, test and deploy. Then we have deployed the *Runners*. A runner picks up a build in your project. Then, every time you commit or merge something in 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 more than 90%.

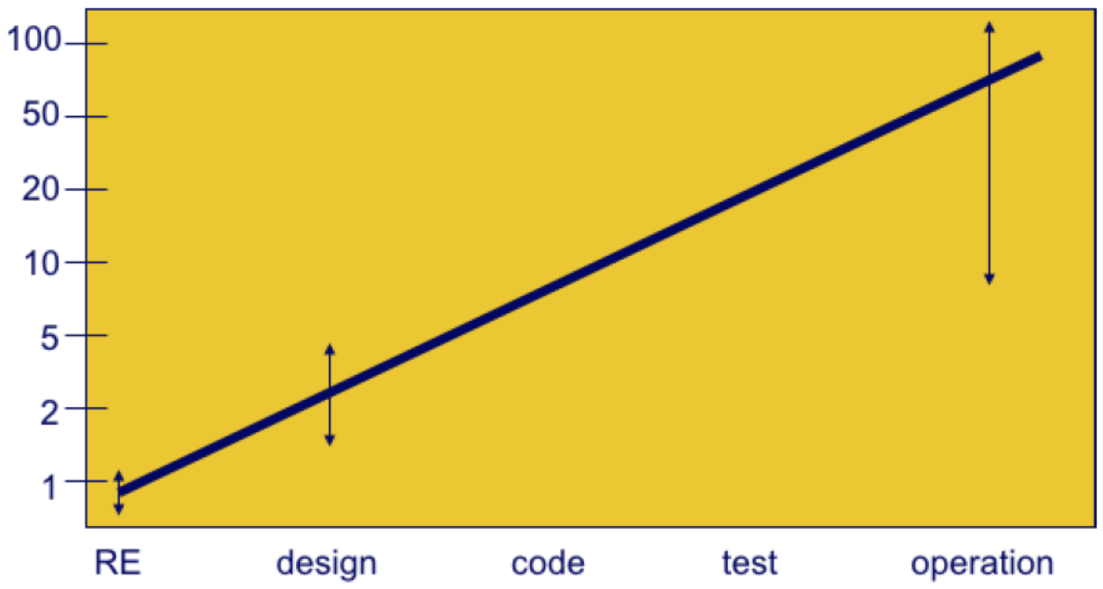


Figure 7: Increasing rate of costs in different software development stages

7 User Documentation

One other important thing that we needed to create was the user documentation. Due to the fact that this is a university course, the software life cycle won't be strictly followed. For example, no future maintenance will take place. The team had to develop by the end of the development of the site a well defined user documentation. The documentation needed to be divided in two parts. The first part will be a brief documentation about how to navigate in the program and perform simple tasks such us adding a new user or deleting an old one. This was easily developed using a simple pdf file. A team member created a navigation document containing screenshots and arrows showing how to perform simple tasks in the website. Figure **number** contains a screenshot of the user manual.

The second part of the documentation is about explaining what technologies are used and generally how to manage the back-end development. Providing a well-understood documentation is highly important since the programs that are provided are funded. This results to the fact that they are economically depended to many different organisations that vary every year and these organisations vary the information that they need to be undertaken from the program users. The development of the documentation was based on a single rule, develop it such us every aspect, every word will be read by a small child. This way, the team managed to develop a well understood documentation.

Having a written documentation as a separate file is not enough. The program, throughout the years will be used by many different people in many different places. The documentation will not be available to them all the time. As a consequence, the integration of a user guide in the page is essential. The team decided to use an already created engine called ”how$_t$o”$.Thisengineenablesboththeuserandtheprogrammertoeasilymanagevariousd$

8 Formative Application Evaluation

A critical stage where maybe the most useful changes is the one of the Formative Application Evaluation. After the successful early deliver of the program to the customers, the team created a questionnaire and in combination with the simple user guide an application evaluation was performed. To put more in depth, the team performed the evaluation in two different stages. The first stage was related with the customers. Due to the fact that the team had already discussed the design issues with the customer, the evaluation and questionnaire were mainly focused in the navigation and understanding of the page. On the fourth customer day a team member performed a navigation to the `usermanual.png`

Figure 8: User Manual

page, explaining to the customers how the page works. Additionally, a user guide and a simple questionnaire was given. The goal was for the team to study whether the customers are able to navigate to the page and if they are able to solve their issues through the usage of the user guide. For the creation of the questionnaire a simple "Google Form" was used. The questionnaires were collected and studied. The questionnaire can be found in Figure ***number***.

As a consequence of having a small amount of responses the results where mainly qualitative than quantitative. The analysation of the responses showed no major navigation issues and the customers were satisfied with the features that the application had.

The second stage of the evaluation was performed by random colleagues. The presentation of the page was a little differentiated than the performed to the customers. Ten colleagues participated to the evaluation. Firstly, every colleague was given a paper which was explaining the purpose of the application. Furthermore, it outlined that everything that was collected were confidential and anonymous. A team member outlined the usage of the page and performed a small navigation based on the user guide. Every colleague performed some simple tasks and asked to answer a small questionnaire. The results of that questionnaire were mainly used to identify some more advanced issues, since a third year student is able to give a more advance opinion compared to the inexperienced customers. A major issue that was identified was a bug in the search engines. Some foreign keys like the area foreign key were used as an integer but presented as a string. Thus, when a user was performing a search was using the name of the area instead of the number of the area. Consequently, no results were returned. The team managed to solve the issue by changes some foreign key rules and relations in the schema.

9 Challenges

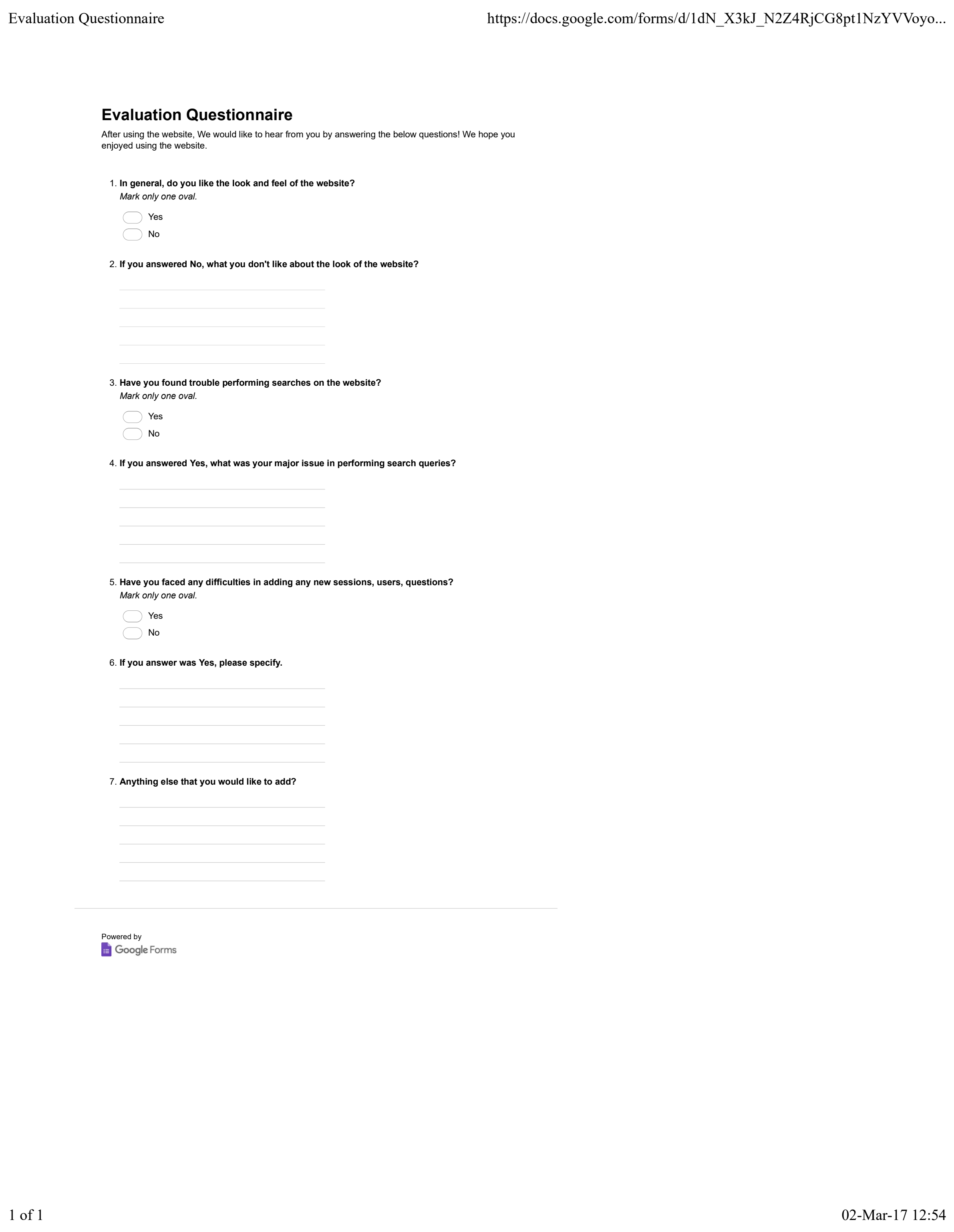
Developing a site from scratch, always creates some major challenges. The second and third retrospective were pretty representative for stating what challenges the team faced and some suggestions of how they were solved. Usually a team has a major issue developing test. Tests were mainly time consuming since we needed enormous amounts of tests. Surprisingly, the tests had a very high coverage of an approximate 90

The organisation of the database was another big challenge that the team have encountered. Although the database was characterised by its simplicity, retrieving the information was tricky since a member of a team could be a volunteer in another team and a general admin. This raised some authentication issues whether 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 users 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 team assigned that as an extra feature which would be developed if there was time, something that was stated clear from the very first day that it was suggested. The team managed to create the feature and successfully add it to the site.

Since we are Agile team, we had another change of the initial requirements regarding the deadline for delivering the final product - we were asked to be ready one month before the due date. In that way the customers would have a month to test the application and to give us feedback.This was quite a challenge for the team because there still was a lot of work to be done. The team managed to deliver the software on time but without same minor features which were progressively added in the software.

Another thing the team was asked to do, as previously mentioned, was the User Manual. This documentation would be useful for the current customers and for any other future users of the application. In that way they can have a simple and brief guidance how to navigate through the application and to take as much advantage of it as they could.


Evaluation Questionnaire https://docs.google.com/forms/d/1dN_X3kJ_N2Z4RjCG8pt1NzYVVoyo...

Evaluation Questionnaire

After using the website, We would like to hear from you by answering the below questions! We hope you enjoyed using the website.

1. **In general, do you like the look and feel of the website?**
Mark only one oval.
Yes
No

2. **If you answered No, what you don't like about the look of the website?**

3. **Have you found trouble performing searches on the website?**
Mark only one oval.
Yes
No

4. **If you answered Yes, what was your major issue in performing search queries?**

5. **Have you faced any difficulties in adding any new sessions, users, questions?**
Mark only one oval.
Yes
No

6. **If you answer was Yes, please specify.**

7. **Anything else that you would like to add?**

Powered by Google Forms

1 of 1 02-Mar-17 12:54

Figure 9: Evaluation Questionnaire for customers

10 Kangaroo Practices

11 Knots and Bundles

12 Conclusions

Taking everything into account, designing and developing a software application wasn't an easy process. The team faced many challenges but managed to overcome them.

Agile programming proved it self that is a very good practice to be followed from novice students. Pair programming played a critical role in terms of identifying bugs and errors in the code. Team organisation saved many valuable time since communication is very important. Although, at some point during the Christmas's break the team faced some communication issues, they were not considered important since the team was already in a very good position in the project so very few things were done during the break. Christmas break was mainly dedicated on revising and self studying about the project.

One of the best practices that we 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.

On the 24th of February the team managed to deliver a fully working program to the clients. Furthermore, on the ** date *** the team managed to complete the program with its added functionality.

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, learn 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 clients and listen with respect their demands, even when they are impossible to be developed.

Explain the wider lessons that you learned about software engineering, based on the specific issues discussed in previous sections. Reflect on the extent to which these lessons could be generalised to other types of software project. Relate the wider lessons to others reported in case studies in the software engineering literature.

12.1 Bibliography

Technology

- ”wikipedia.org: Ruby on Rails”
  https://en.wikipedia.org/wiki/Ruby_on_Rails[Accessed 14 December 2016]
- ”wikipedia.org: Django”
  https://en.wikipedia.org/wiki/Django_(web_framework) [Accessed 14 December 2016]
- ”skilledup.com: Comparison of Django and Rails
  http://www.skilledup.com/articles/battle-frameworks-django-vs-rails [Accessed 14 December 2016]
- ”wikipedia.org: RubyGems”
  https://en.wikipedia.org/wiki/RubyGems [Accessed 14 December 2016]
- ”postgresql.org: PostgreSQL”
  https://www.postgresql.org/ [Accessed 14 December 2016]

User stories

- ”agilemodeling.com:User stories”
  http://www.agilemodeling.com/artifacts/userStory.html [Accessed 26 December 2016]

Entity Relationship diagram

- ”smartdraw.com: ER diagram”
  https://www.smartdraw.com/entity-relationship-diagram/ [Accessed 12 January 2017]

User authentication

- ” searchsecurity.techtarget.com: User authentication”
  http://searchsecurity.techtarget.com/definition/authentication [Accessed 15 January 2017]
- ”github.com: Devise”
  https://github.com/plataformatec/devise [Accessed 15 January 2017]
- ”railstutorial.org: Ruby on Rails Tutorial (Rails 5) By Michael Hartl”
  https://www.railstutorial.org/book/ [Accessed 15 January 2017]

Testing

- ”docs.gitlab.com: Continuous Integration (CI)”
  https://docs.gitlab.com/ce/ci/quick_start/README.html [Accessed 19 January 2017]

User Authentication

- ”github.com: How$_t$*ouserdocumentation*”
  $https://github.com/amuntasim/how_to$