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

January 17, 2017

Abstract

Building Healthy Communities is a program which is aimed to create interventions in areas with the collaboration of the locals and generally various community groups. Its main goal is to improve the life o people by participating in various programs which will enable them to interact with other people and get a healthier approach in life. The program consist of three different types of people:

- Administrators
- Volunteers
- Members

Volunteers are responsible for tracking the members progress and attendance. They also have to inform administrators not only about process of the members but also about their process. Currently all these are being performed by hand. Administrators find all the process time consuming and difficult to generally track everything.

Our team was responsible building a software which will automate all that process. We designing and implemented a program that accepts a current database that could be modified and extended. The software has the following features:

- Register new members and delete ones that left the program
- Keeping track their attendance
- Keeping track the process not only for the members but also for the volunteers
- Delete and add programs

Education Use Consent

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

1 Introduction

Software engineering

The following presentation presents a case study of building a database for the public health program "Building Healthy Communities". By the expression "case study" we mean the investigation of a person, group of people or a situation. The following report investigates the situation of creating a database. The team that is involved in this case study is "Team N" which consists of six, third year colleagues studying Computing Science in the University of Glasgow.

Through the following pages, we will explain in more depth, how we managed to successfully design, create 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 have taken in creating the software. From the requirements gathering to the coding of the software, the lecture notes provided a step by step guide that lead to a perfect software.

As expected, the path that lead us to the creation of the program, wasn't easy. The majority of the team members had a little previous knowledge in the technologies that we have used. As a result, we had many difficulties, many will be described in the following sections.

Our dissertation is being divided..

2 Case Study Background

Include details of

- The customer organisation and background.
- The rationale and initial objectives for the project.
- The final software was delivered for the customer.

3 Agile programming and Team organisation

3.0.1 Team organisation

In our initial meetings, every team member pointed out his/her strong and weak parts so we can divide the work. Due to the fact that we were required to create and application and database from scratch, the workload was big so 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 started discussing for various means to communicate, we decided to avoid using common social media sites and to use the "slack" application, which provides a more organised and professional chat room.

3.0.2 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. 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". Extreme programming was the most suitable decision. Due to the fact that we are novice programmers, a very useful feature that extreme programming has

called "pair programming" was highly included in our practices. As students we lack of experience but when one student combine his/her 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 was the automated testing because as I stated before we have a very small experience.

3.0.3 Technologies Used

As soon as we decided 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 aren't, 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 and provides an amasing GUI that organises the wiki page, 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 his/her self 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 haven't done a lot in a specific week and assign less to members that done more work than needed.

After this discussion, the meeting wasn't over. We know needed to decide which technologies would be used for implementing the project it self. We needed to choose a web application framework. The first suggestion was Django this, every team member had previous experience to Django from the second year course. A team member suggested that Rails is a more powerful tool with better documentation. To put more depth in it, Rails is a web application framework. Its software design pattern is the "model-view-controller" and not "model-view-template" like Django which we were familiar. None of the six members coded on Rails before so we have postponed our meeting and created an issue in the Gitlab so every member in the team could study Rails in his/hers self time so we can vote. In the next meeting, every team member 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, this 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. SQLite was agreed to be used because the database would be a small with just a couple of thousand text records. SQLite is a self-contained, free to use relational database management system.

3.0.4 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. Every team member, was filling at least one point in every box. This way, we would not only have time to improve, but also to learn new technologies or discover in more depth the needed technologies for accomplishing our tasks due for the next meeting.

3.0.5 Team roles

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

3.0.6 Testing

maybe move it in the back-end section ???*

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. 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.

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.

4.0.1 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.

4.0.2 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 continuous contact with the customers whom 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.

4.0.3 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.

4.0.4 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 he/she 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.

4.0.5 Wireframes

After collecting all these useful information, the team was ready to start developing some wireframes. Wireframes was the final goal of this milestone. Without completing them would be like we haven't 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.

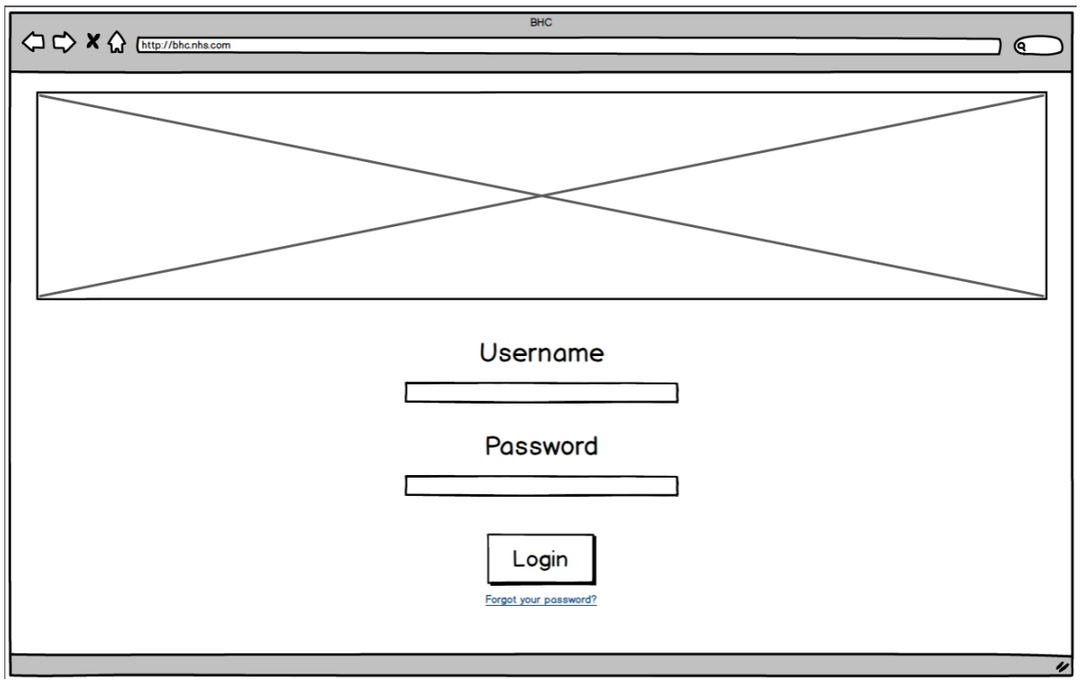

Figure 1: Given structure of WireFrame

4.1 End of first milestone

The second customer meeting was held in the 16th of 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 kept from both the team and the customers. After the meeting, the team filled a retrospective. An outline of the retrospective was that the team during this period of time improved its knowledge on using an application such us "Gitlab" as a project management tool, that one project needs not only coding but first large amount of documentation to be produced. The team is also hoping to improve the communication with the customers since we didn't understand all the aspects of the application in the first place and 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 meeting to decide the objectives for the next customer meeting. A small amount of documentation was still missing such as the ER (Entity-relationship) and Component diagrams. The team assigned to team members the diagrams. The team should not only create the diagrams but also to 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, without the diagram the team is unable to have a clear view of the structure of the database. Having an ER diagram presents clearly the relationships between the entities, the

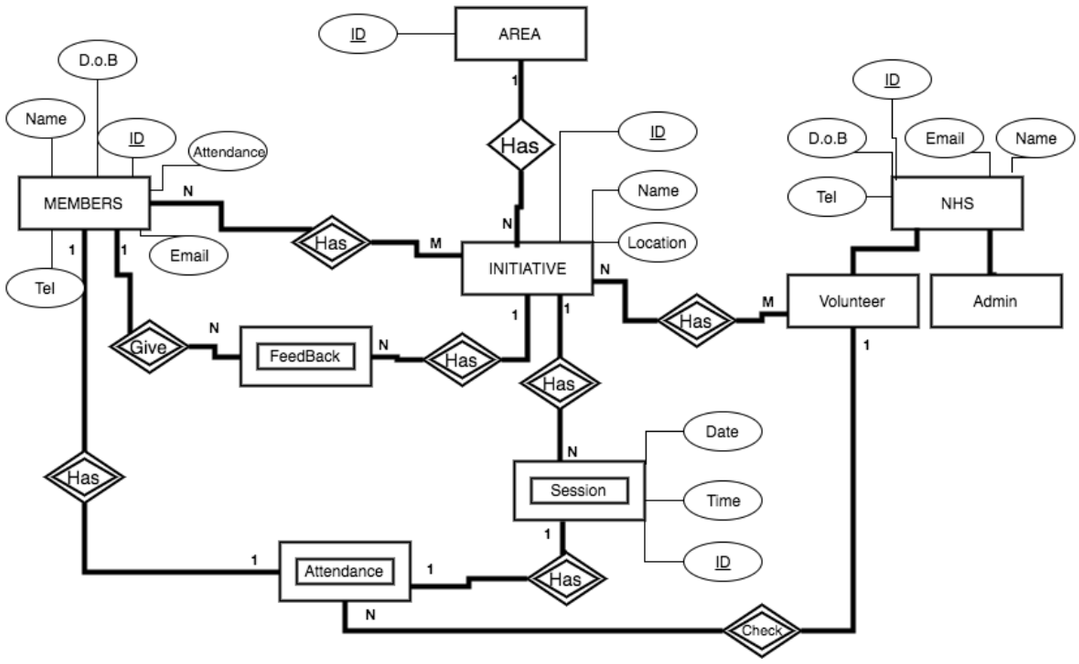

Figure 2: Entity Relationship Diagram

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 i order to get 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.

6 Alice

7 Choice of Colours

The following diagrams (especially figure **??**) illustrate the process...

8 Managing Dress Sense

In this chapter, we describe how the implemented the system.

9 Kangaroo Practices

10 Knots and Bundles

11 Conclusions

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.

Rails https://en.wikipedia.org/wiki/Ruby$_on_Rails$

Django https://en.wikipedia.org/wiki/Django$_(web_framework)$

Rails vs Django http://www.skilledup.com/articles/battle-frameworks-django-vs-rails

RubyGems https://en.wikipedia.org/wiki/RubyGems

SQLite https://www.sqlite.org/about.html

CI https://docs.gitlab.com/ce/ci/quick$_start/README.html$

User stories http://www.agilemodeling.com/artifacts/userStory.htm

ER diagram https://www.smartdraw.com/entity-relationship-diagram/