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 19, 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 of people by participating in different programs which will enable them to interact with each other and get a healthier approach in life. The program consist of three different types of participants:

- Administrators
- Volunteers
- Members

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

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

- Registering new members and deleting ones that left the program
- Keeping track of member's attendance
- Keeping track of the process not only for the members but also for the volunteers
- Deleting and adding programs
- Searching for a particular member based on given specifications

# 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 led us to the creation of the program, was far from easy. The majority of the team members had a little previous experience in the technologies that we have used. As a result, we had a lot of difficulties, many will be described in the following sections.

Our dissertation is being divided..

# 2 Case Study Background

## 2.1 Customer Organisation and Background

Building Healthy Communities is a NHS programme based in Dumfries and Galloway, which is part of the concept of Healthy Living Centres. The aim of the organisation is to influence health and wellbeing by social, economic and environmental factors, as well as to improve the quality of life of all people.The programme consists of a variety of Local Health Initiatives that address the needs of the local people.It requires partners outside health services to help address issues around health. Individuals are given the opportunity to become Community Health Volunteers through training and one-to-one support. Building Healthy Communities develop an effective partnership between people as enabling individuals to be active citizens within their own community.

## 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.

- The final software was delivered for the customer.

# 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.

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. SQLite was our choice because the database would be small with just a couple of thousand text records. Furthermore, SQLite is a self-contained, free to use 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. Every team member, was filling at least one point in every box. This way, 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.

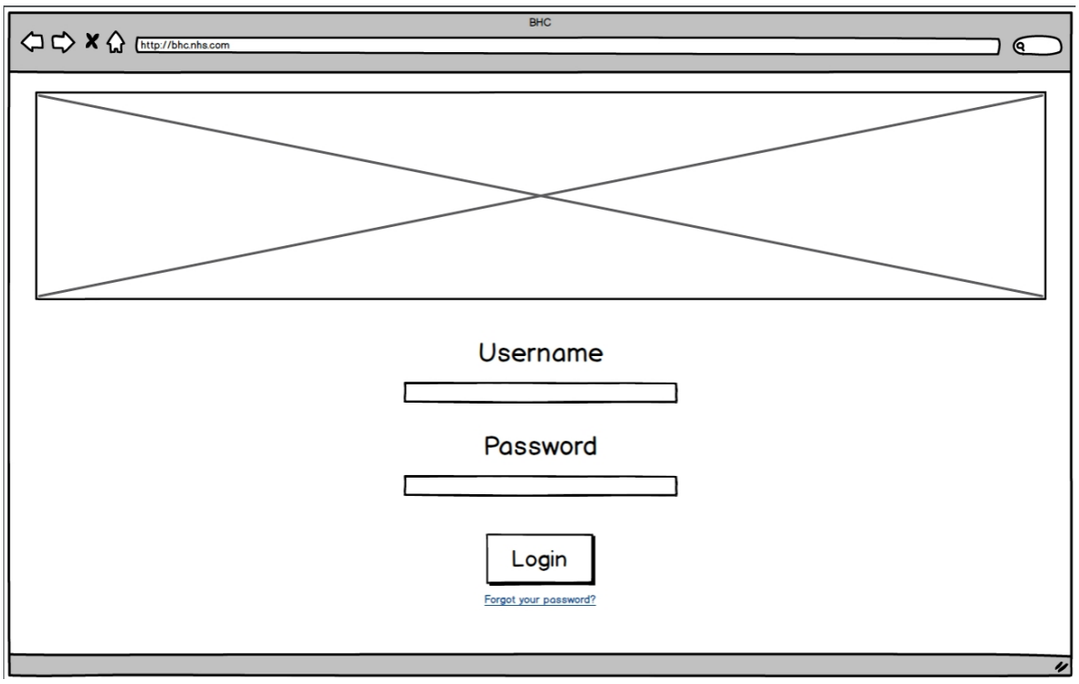


Figure 1: Given structure of WireFrame

### 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 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.

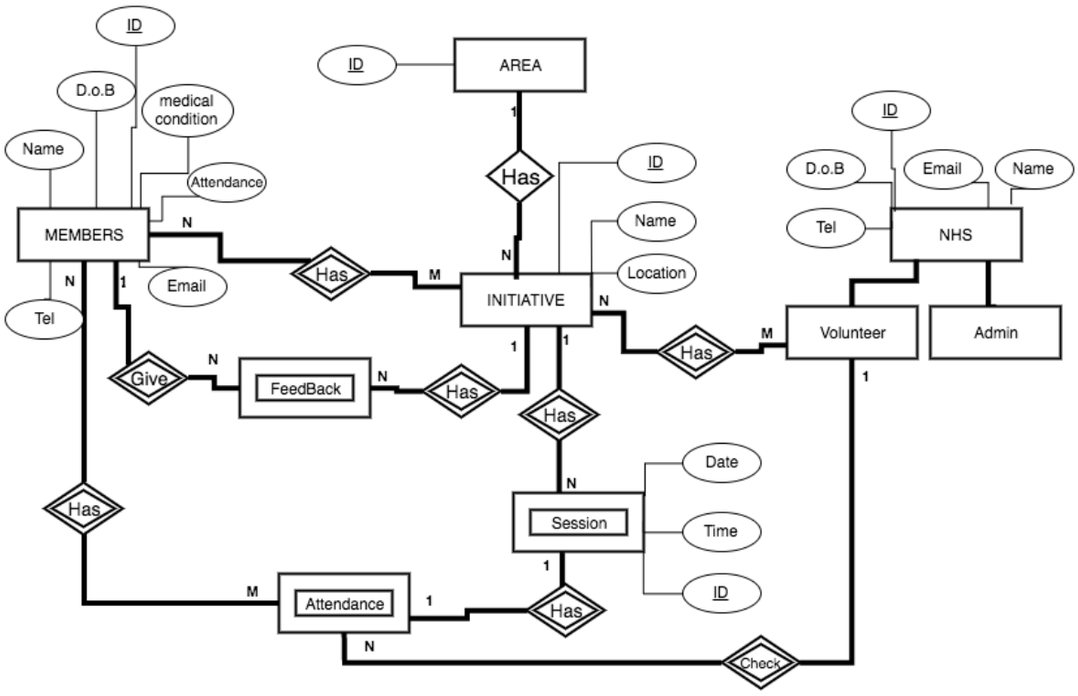


Figure 2: Entity Relationship Diagram

# 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 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.

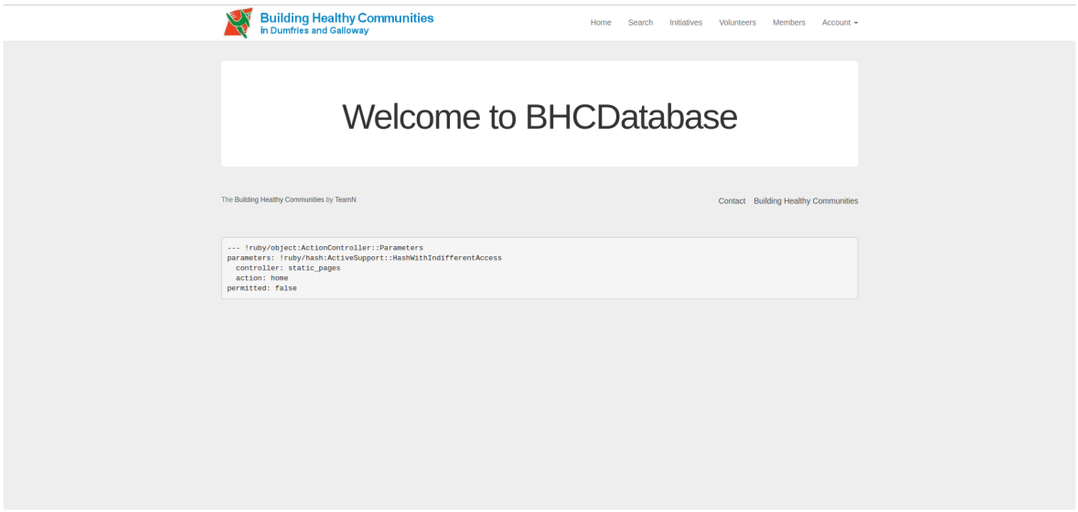


Figure 3: Initial Prototype


Figure 4: Second version of the prototype

## 5.3 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.

## 5.4 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.

### 5.5 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.

### 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

***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. Providing testing in advance reduces the time and the cost needed to identify and correct defects. The graph

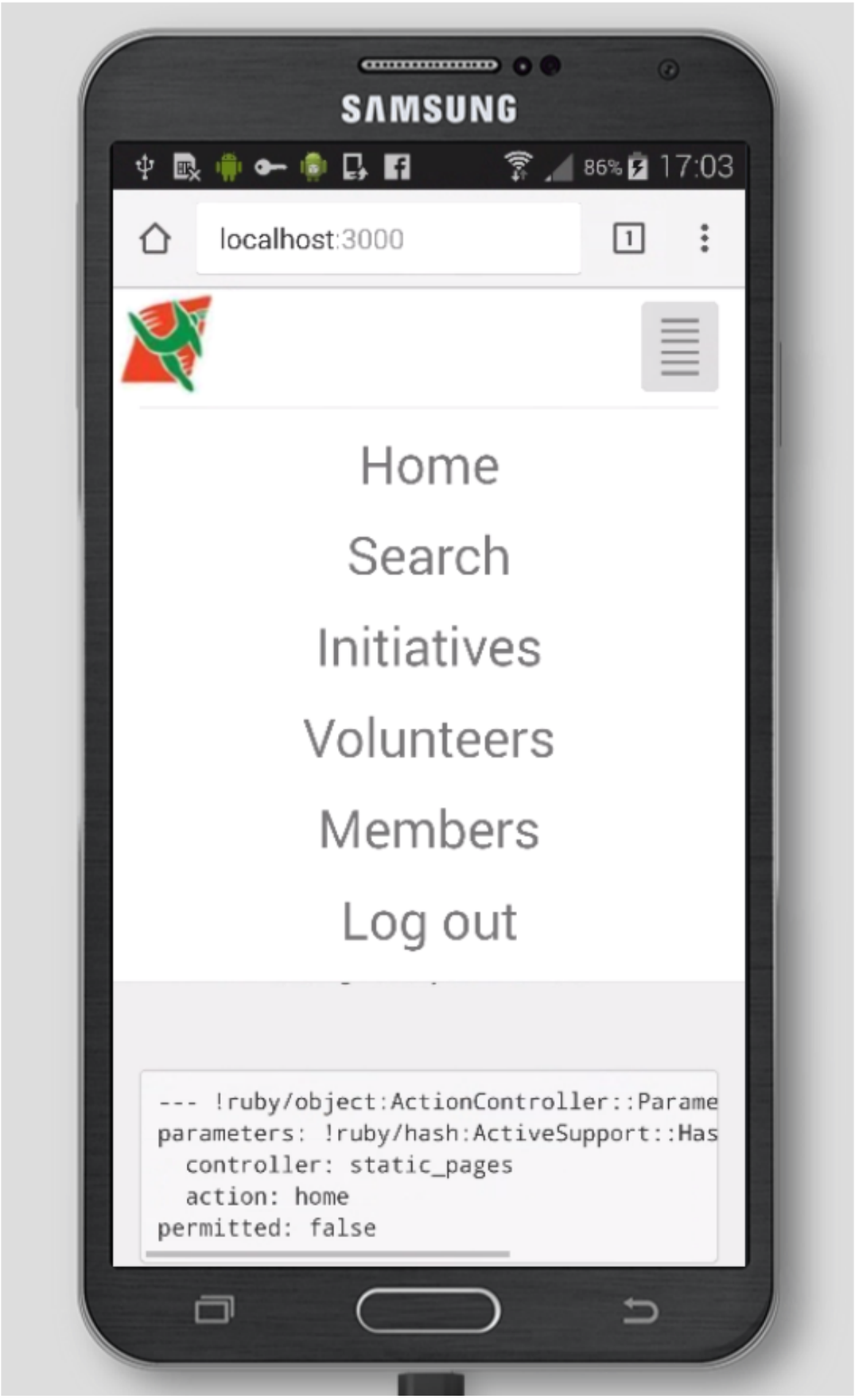


Figure 5: Mobile Compatibility Prototype

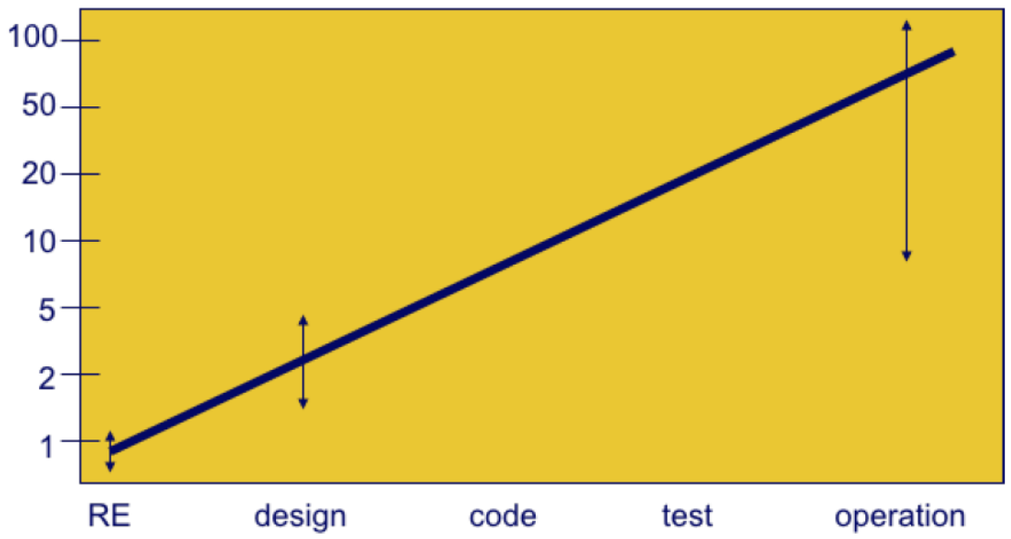


Figure 6: Increasing rate of costs in different software development stages

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

# 7 Alice

# 8 Choice of Colours

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

# 9 Managing Dress Sense

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

# 10 Kangaroo Practices

# 11 Knots and Bundles

# 12 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$_o n_R ails$

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/
User authentication http://searchsecurity.techtarget.com/definition/authentication
Devise https://github.com/plataformatec/devise
Ruby on Rails Tutorial (Rails 5) By Michael Hartl https://www.railstutorial.org/book/