SEM5640 Group Project
Report

Go!Aber

Submitted in partial fulfillment of
the requirements for the award of the degree of

MEng
in
Software Engineering

Submitted by

Student No.	Connor Goddard
Student No.	Helen Harman
Student No.	Craig Heptinstall
Student No. 110036072	Samuel Jackson
Student No.	Daniel McGuckin

Department of Computer Science
Aberystwyth University

December 2015

Abstract

Contents

List of Figures

Chapter 1

Project Overview

Following from the brief provided at the start of this project[3], we are able to clarify a detailed overview of the application requested by the client. A new system to be named 'GoAber' would allow members or 'participants' from a university to interact with other participants enabling them to record activity data such as step counts, distances travelled and heart rates.

The System means to allow a number of methods for entering activity data from sources such as Fitbit[1] and Jawbone[2] devices, alongside manual and external entry via an application API. The external API would then be able to cope with users who wish to use other devices such as smart watches or health tracking mobile apps.

Users should be able to access the system either via a vanilla log in or through their university emails, though the different sign-on systems of universities may have to be considered. With the data entered by users, they would then be able to interact with other participants on a deeper level, through challenges. With challenges, the overall purpose of the application is to enable staff and students of a wide set of universities to keep active, and give them incentive while comparing their efforts to others. The application will give each participant the opportunity to work as a team when performing challenges, by allowing them to become a part of a 'group'. For instance, a certain department or group of individuals at a university may want to be represented, and therefore compete with other groups.

Following on from groups and interacting with users, it is important that different levels of authorisation is followed for anyone logged into the system. There is a total of three level of authorisation required of the system:

- Participants- as mentioned previously, will be part of a group and university(community).
- Coordinator- A user with privileges to create challenges, and add users to a group that they created.
- Administrator- A higher level user, with all the abilities of the previous,

alongside the abilities to edit and remove participants activity data, groups, and communities.

The final high level features that the system should be able to perform is to use the activity data entered by participants to display some form of league table in order to rank both groups, institutions and individuals. Alongside this, a scheduled email system will be in place for users to receive updates on their performance in challenges, and produce emails when a user is inactive for a certain length of time.

The client has asked the system to be implemented in both JavaEE and .NET, allowing a more flexible choice of installation for universities that may want either a Linux or windows machine to run their server. Each running instance of the application should be self dependent, and only send or request data from the others when interaction such as challenges occurs. A thorough set of testing should be performed before the release of the application to ensure that the user interface functions correctly and that the bilingualism the user has asked for works, while the logic in the back of the application should be tested through unit tests and stress testing to ensure the application can withstand a number of concurrent users.

Finally, this report outlines the process of work completed during the project, detailing each stage of the project. Any modifications or clarifications to to the original client brief will be described in the following section.

1.1 Detailed requirements

The requirements spoken of here are based off the initial requirements specification document [3] and some of the key requirements are mentioned alongside their requirement codes. Before the design of the project began, a few clarifications was made for some of the requirements in a QA session with the client, all of which are described in the group project meeting minutes. These will be available in the 'docs/minutes' folder in the hand-in. To best detail the requirements of the application, parts of the application can be grouped by functionality.

1.1.1 Management of users, authentication and authorisation

The first of the requirements that should be available to the user is the registration and signing into the application. As mentioned in the requirements specification under D-FR1, the system should allow users to be added to the

system and to a group within a community. As discussed with the client in the first meeting, signing up for the site should be both available through the university credentials (SSO), or a vanilla log in, with a safe means of storage.

Alongside the registration of users, there should also be the functionality to edit users details and remove them. All levels of users should be able to perform these actions, though it was clarified in the QA session that only admin-authorised can edit and remove others users. Other administrator level activities only applicable to those users include renaming groups, removing groups, renaming their community, and deleting any other users' activity data. The auditing of other users' data is described in the auditing section of this chapter.

To clarify how users will connect through groups and communities, the example below highlights an example scenario:
A groups such as 'Computer science' would be contained within a community such as 'Aberystwyth University', and a participant would be added to Computer science by a coordinator that created that group. Then if for instance another group challenged Computer science the user's activity data would combine with others in their own group to compare their totals with the second groups. This is a similar story for communities, where totals of all participants of a given community could be compared with another community.

1.1.2 Activity data

The next subset of requirements required of the system is the integral need for activity data to be saved. As mentioned in the overview, the user will should be able to link Fitbit and Jawbone devices (of which data will be grabbed on a daily basis, or by syncing manually at any time), alongside manual entry of data. This has been outlined in D-FR2, which also mentions that there should be some means of saving all types of activity mutually.

This was another part of the requirements specification that was clarified with the client, where it was decided that categories of activity data could be expected as always a numerical format and should be dynamic to allow for more categories to be added. A final clarification here was that although dynamic addition of categories of data should be possible, three types (walking, cycling and running) will suffice at this point.

1.1.3 Data auditing

Alongside the entry of data, D-FR8 was a feature of the application that the client was intent on being implemented. The auditing of data should be

possible for administrators and for participants (where the participants are only changing or removing their own data), and notes should be left by the remover to allow for a reason to be left to a user wanting to know why data was modified or deleted.

An extension of the auditing data feature that was clarified again with the client is the auditing of all actions by the administrator. Unlike the data auditing, this would mean an auditing record should be created for any administrator level user that performs an action only available to them. Another highlighted clarification of this requirement is that all auditing should be final, and that no rollback functionality is required. For instance, if an administrator deleted data from a user, to undo this action would not be possible.

1.1.4 Challenges

Linking back to the way users interact in the system, challenges is one of the largest and most important aspect of the proposed system. C-FR1 to C-FR4 outline the challenges requirements, though again clarifications have been made since starting this project. Rather than assuming a challenge will be sent to another community or group for them to accept, instead the flow of challenges should be as follows:

- Coordinator of a group creates a challenge
- A coordinator from another group joins the challenge
- Both groups users can compare the competitions progress and statistics

In addition to the information about overall progress, users in a challenge should be able to see a league table within their own group. It was clarified in the QA session that summaries for each user should be available to all others in their groups, meanwhile outside their own group, only a very basic stat should be visible. This goes also for privacy of names and personal information, which should only be visible by people in their own group, although users will need control of their user name, which could be made anonymous if requested. As for higher level users, administrators will see all users details within challenges, and coordinators will see the same information but at a restriction of only overall data.

Upon completion of a challenge which will be decided by length of time of the event, it will be the responsibility of the coordinator that created the challenge to publish results and inform any participants taking part. Although this should be automated, initially sending out results should be completed

by the coordinator to give time for validation of the data. Alongside the emailed out results, data from completed challenges should be visible on any participants of the challenges' dashboard upon next log in to the application.

A smaller additionally desired but not essential feature discussed during the QA meeting was extra results for categories such as 'best walkers' etc.

1.1.5 Updates and emails

This brings this part of the requirements specification to how emailing will be handled. Emails are an additional though highly desired part of the brief responsible for both sending mail out for completed challenges and for reminding inactive users to register data. Where reminders and challenge news is sent out, the amount of time between each automated email should be modifiable by an administrator.

As per discussed with the client, the default times for each email type should be as follows:

- Participant progress emails- Weekly
- Missing readings emails- Daily
- Results of users- Manual

By combining the automatic tasks of sending emails and receiving data from external services, it should be possible to make use of a general scheduler to help perform a range of actions when the user decides.

1.1.6 User interface and internationalisation

Because the application will be a web-application, it is key that the application has an easy to use appearance that is preferably mobile friendly. The client has agreed that the default appearance using the Bootstrap[4] is allowed here. In addition to a friendly user interface, internationalisation is a required feature of the system, and should provide the user the ability to switch between at least the English and Welsh languages with opportunity for more.

1.1.7 External endpoint

Although part of D-FR5 which covers the requirement of receiving data from a variety of means, a special mention to the requirement that the application should have an API endpoint. The API (to be SOAP) will be required to

allow interaction between different servers running the application in order for different communities to communicate and send challenges.

Alongside this, the API endpoint should also be able to receive generic activity data to be inserted into the system (though the client has clarified that this part of the API will only require POST, as to only receive data).

1.2 Desired and required libraries

In addition to the requirements of the application, the client has requested that third party software should be made use of to encourage a faster implementation of the product. Uses of third party software was discussed during initial meetings, with some group members already having a good knowledge of possible email libraries and OAuth libraries to help with the connectivity to Fitbit and Jawbone services. Both of which will be discussed further in this report, with reasoning for why such libraries were chosen.

Chapter 2

Development Methodology

Chapter 3

Design

This section outlines the initial designs for the system to be produced. As this project is being developed using a Scrum based development methodology, there is no concrete up front design for the system. Much of the design presented here was created during the initial stages of the project and therefore has been subject to change throughout the implementation stages of the project. However, we were not going to start development on the project without any design whatsoever. Especially since nearly all team members were unfamiliar the technologies we were going to be working with.

Chapter 4

Implementation

Chapter 5

Testing

Chapter 6

Project Status

Chapter 7

Critical Evaluation

References

[1] Fitbit. Developer api. `http://dev.fitbit.com/uk/`, year =.

[2] Jawbone. Jawbone up api. `http://jawbone.com/up/developer`, year =.

[3] Nigel Hardy Neil Taylor. Go!aber requirements specification. *SEM5640 Group Project*, 2015.

[4] Twitter. Get bootstrap. `http://getbootstrap.com/`, year =.