CS993 – Time Booking System

Team Lewis – Jonathan Bowen, Boakya Dankwa, Alistair Hill, Jane Howell, Calum Mortimer

# Introduction and Background

Target – 300 words

# Requirements

Target – 1000 words

For the successful development of the proposed software according to the client’s needs, it was critical that from the offset the requirements of the software were clearly understood and documented. This offers certainty to both parties. For us, the developers, this means that the product we are designing would be clearly understood. For the client, they would understand what product was going to be delivered at the end of the project.

A good, clear set of requirements would remove ambiguity and minimise the risk of conflict between both parties. It should be obvious to both the software engineering team and the client what represents a contractual change which requires further scope and what simply represents the software engineering team abiding by the previously agreed scope.

The project team’s approach to formalising the requirements with the client consisted of the following process:

1. The client produced a user requirements specification (Appendix A.1) which detailed the requirements of the software system

2. A formal meeting was held with the client and the software development team to discuss the initial requirements and follow up with any questions in-person.

3. A numerated requirements list was developed by the software team from all of the available information (Appendix A.2) which served as the initial set of requirements.

4. A numerated set of assumptions (Appendix A.3) was developed by the software team and sent to the client for clarification.

Following step 4, no further communication was received from the client to the software team. Had the team managed to clarify the assumptions with the client and formalise the requirements list, this would have potentially allowed us to produce a formal quotation for the works required.

The following sections detail the workflow we pursued for requirements capture, the information which we obtained, and the steps we would take in the future to bring the project to eventual completion.

# Design

Target – 1000 words

V1.

The team made use of a design technique called (physical) *class-responsibility-collaboration* (CRC) cards as a method of visualising and realising the application’s architecture. This brainstorming method is useful for establishing the classes and data that would be necessary to develop the application, before starting to code. Each data structure we designate as a *class* is written on an index card, then it’s *responsibilities* (i.e. the things it can do) are listed. From this we can complete by inference the final section, by listing the other classes that we could consider each class’ *collaborator.*

Other advantages of this high-level design include that it is easily communicable to the client, regardless of their technological knowledge. Being able to communicate effectively with the client at this crucial, early stage can prevent setbacks later in the development process. Furthermore, the CRC cards serve a dual-purpose, providing an unambiguous, strong springboard from which the development team can begin to code the system.

The use of CRC cards uncovered design challenges that we had not foreseen until that point.

Using this information, a class diagram was created (Appendix A.4) which listed each class we intended to create, the data it would handle, and the methods contained in that class.

# Construction

Target – 800 words

# Testing

Target – 1000 words

# Methodologies and Tools for the Software Development Lifecycle

Target – 800 words

# Concluding Remarks

Target – few words

# Appendices

## Client Requirements Document

# 

## Project Requirements List

1. As a 'user', I need to 'log in', so that 'I am given appropriate information from the system.'

2. As a 'user', I need 'to be restricted to my own projects/tasks' so that I cannot interfere with irrelevant projects/tasks

3. As an 'administrator', I need to make new tasks/projects, so that users can book time against them.

4. As an administrator, I need elevated system privileges, so that I can administer the system.

5. As an administrator, I need to assign users to projects, so that the users have access/can use the system

6. As an administrator, I need projects to be composed of tasks, so that tasks can be easily organised and users can be assigned more flexibly.

7. As an administrator, I need the ability to assign users to specific tasks within a project, so that I can have control over what users may book time towards.

8. As a 'user', I need to submit project\_code, task\_code, time details & comments when booking time against a task. So that my work can be accurately tracked.

9. As a user, I need to be locked to one task at a time, so that there can be no time conflicts between tasks.

10. As a 'user', I need to be able to edit/recover/reset my account details, so that I am not prevented from logging in.

11. As an 'administrator', I need to be able to delete user accounts, so that old users can be removed from the system.

## Project Assumptions List

1. An admin can create projects and tasks - a regular user (time booker) cannot.

2. A task code is a sub-code of a project code.

3. A time booker can be assigned to entire projects or specific task codes within a project.

4. Admins are also time bookers.

5. A time booker cannot book time which conflicts with another of their time bookings (booking the same time twice).

6. An administrator has administrative access to all projects and time bookings - not a subset of projects which they administrate.

7. If the application is disconnected within the mobile version, how should login authentication work without a connection to the server?

8. An administrator can add and remove user accounts.

9. What is the hosting preference for the application? Cloud-computing or a standalone server? We will assume a Heroku cloud-based solution as deployment will be quick and require few man-hours to complete. However, this could be more expensive than other cloud based solutions at scale.

## Class Diagram

../Booking%20System%20Class%20Diagram.pdf