CSCI 3428 - Software Development Plan

Group 4

Monday 18th November 2019

1 Schedule

See figure on final page.

2 Team Organisation

With a larger group, it is possible to delegate more members to work on the front-end of the project, focusing specifically on creating a user-interface that is well-suited to the needs of each of the individual clients. The underlying core functionality will likely be the same for each of the users, and so fewer back-end developers are required. A larger group also grants more flexibility for group-members to move back and forth between the different development areas, as there is unlikely to be a shortage of people able to work on any one aspect of the project at any given time. This means that, if a member decides they would like to move from front-end to back-end to implement a novel idea, they are free to do so (within reason).

A larger group presents management challenges, however, and can slow the ideation and implementation processes. As a result, one possible consideration involves dividing the group into two smaller sub-teams, that will each focus independently on development for a specific client. This will likely happen later in the development process, once a baseline for the UI and functionality has been established, and the focus turns to tuning the software according to the client's needs.

2.1 Roles

Leader(s):
Veronica Hatala

Front-end Developers:
Gautham Chalapathy, Lulu Chen, Akhtar Rafid, Yilin Zhang

Back-end Developers:
Veronica Hatala, Martin McLaren, Brian Philip, Sheldon Taylor, Kreetish Venkataswami

Server-side Developers:
Sarah Clarke, Martin McLaren, Brian Philip, Sheldon Taylor, Kreetish Venkataswami

Specialists:
Gautham Chalapathy, Sarah Clarke

3 Proposed Standards, Procedures, Techniques, and Tools

3.1 Waterfall w/ Phased Development

From a group-management perspective, development of the project will occur under the Waterfall framework. While an iterative approach provides a standard for adding functionality as the clients' understanding of the product develops, phased development streamlines the management of the development process into a linear sequence of steps, in turn reducing the complexity of the project. Additionally, rigorous planning mitigates the impact of difficulties faced in the program's development. Lastly, the Waterfall model emphasises the importance of creating detailed documentation outlining the system's functionality, design considerations, and long-term management. Given that maintenance of the project upon its completion will not fall to the team's members means having such documentation will be especially valuable.

3.2 React

The front-end of the project will be built up using the React library for JavaScript. React allows us to encapsulate the layout and structure of the program within a JavaScript function (or functions). Doing so allows the front-end (and in particular, input from the users) to be posted directly to the server, and in general does away with any intermediary steps in the communication between the program on the user's end, and the server and database. This is particularly helpful in instant messaging applications, which require frequent communication with the hosting server/database.

3.3 Django

User-management, security, and communication between the front-end and the server will be done using the Django framework. In general, developing under a web framework ensures a consistency in both the design and functionality of the various aspects of our project (through the use of standard libraries, for instance). They additionally provide tools that makes creating databases and handling user management more efficient, both of which are key challenges the project faces. Django in particular is well-suited for these needs, as it uses a Python back-end that is relatively simple to learn and automatically solves many of the various security considerations. While there will be some initial difficulty merging the front-end that we create using React with the back-end and server under the framework, this initial overhead is acceptable considering the gains that will be made further along in the development process.

3.4 MySQL

For the purposes of this project, any relation database management system available today would present a suitable choice. However, given the wide availability of resources, its accessibility on the host-server, and the server-development team's past experience with the software, MySQL will be used to create the program's database, and manage user-data (which includes their login credentials, contacts, and conversation histories).

4 Configuration Management Plan

4.1 GitHub

Given the scale of the project and the size of the group, a mechanism for sharing code is required to ensure that development progresses efficiently. GitHub is an obvious choice for such a platform, and will act as the main hub for the project. Other attendant benefits include the ability to track any changes made to the code, revert to an earlier version in case of serious error, and ensure that all progress is saved in the event of data loss. Lastly, the ubiquity of the platform and the number of resources available that document its functionality means the overhead cost of learning to use GitHub is relatively small compared to the benefits gained.

4.2 Long-Term Management

The long-term management of the project will fall to the system administrator, whose duties will include user-management (including adding new users, and helping existing users with issues pertaining to their accounts), adding or modifying the functionality and appearance of the program, and resolving any issues. Along with editing the source files, there may be additional need to migrate the project to a new host-machine in the future. The details of this is clarified in section 6 below.

5 Risk Management Plan

While it is difficult to assess potential risks while the details of the group's organisation and ultimate goal are still being assessed, it is possible to identify several general obstacles that the project might face:

5.1 Withdrawal

Probability: Low (<20%)
Given the nature of the course (third year, no examinations, single large-scale project), and that the term is nearing its completion are a third of the way, it is unlikely that a group member will withdraw from the course going forward.

Impact: Low (2/5)
Though a withdrawal would typically have a large negative impact on the progress of the group's development, a large group means that a single member's responsibilities can be distributed across the team without putting too much burden on the other members.

Mitigation:
Discuss with the team prior to delegating responsibilities and beginning development to assess any potential challenges they might face both in the course and outside of it that might push them to consider withdrawing, and discuss possible solutions (assigning less work, getting extra support, etc.).

5.2 Data Loss

Probability: Medium (50%)

The loss of data, whether through broken, lost, or faulty equipment, is a common occurrence, and one that is difficult to predict.

Impact: High (4/5)
Without taking the proper precautions, data loss presents one of the most substantial risks to the project, threatening weeks of progress, and resulting in large set backs, or an inability to meet the requirements.

Mitigation:
Protecting against data loss is becoming increasingly easier with the wide availability of cloud services and the reduced cost of storage media. Additionally, distributing new code to each member of the group using GitHub lessens any potential impact.

5.3 Late-Stage Changes in Client's Needs

Probability: High (>60%)
A client's needs change on a regular basis, and this is particularly true for a product that is custom-designed for them. While such a change is not itself a problem, a large-scale change late in the development cycle can pose a significant setback.

Impact: Medium (3/5)
Depending on the scale, and the time during the development cycle in which they are brought up, this could represent a serious setback to the project, potentially requiring ground-up changes in functionality or design. Alternatively, small-scale changes can be implemented at virtually any point in the development cycle at little extra cost.

Mitigation:
Involve clients at every point of the development cycle (within reason), determine a set of core needs that won't change, and work on addressing these first. Communicate with clients what a reasonable set of expectations for the product looks like, and what the team is able to achieve given the time constraints.

5.4 Failure to Meet Deadline

Probability: Very Low (<10%)
With the size of the group, the importance of the project, and the schedule created, it is unlikely that the base project will be incomplete by the deadline.

Impact: Very High (5/5)
A non-functioning product, or one that doesn't meet the client's specifications represents a significant failure to complete the project.

Mitigation:
Create a detailed plan that specifies the key features of the project. Ensure each group member has a clear idea of what their responsibilities are, and where in the development process they should be at various stages. Ensure resources are allocated appropriately, and address any problems as they arise.

6 Installation and Maintenance Plan

The project is a web application that is be hosted on the undergraduate computer science student's server at Saint Mary's University (`ugdev.cs.smu.ca/~group4`). The details of how this project's databases and server-side components is implemented is clarified within the Installation and Maintenance guide.