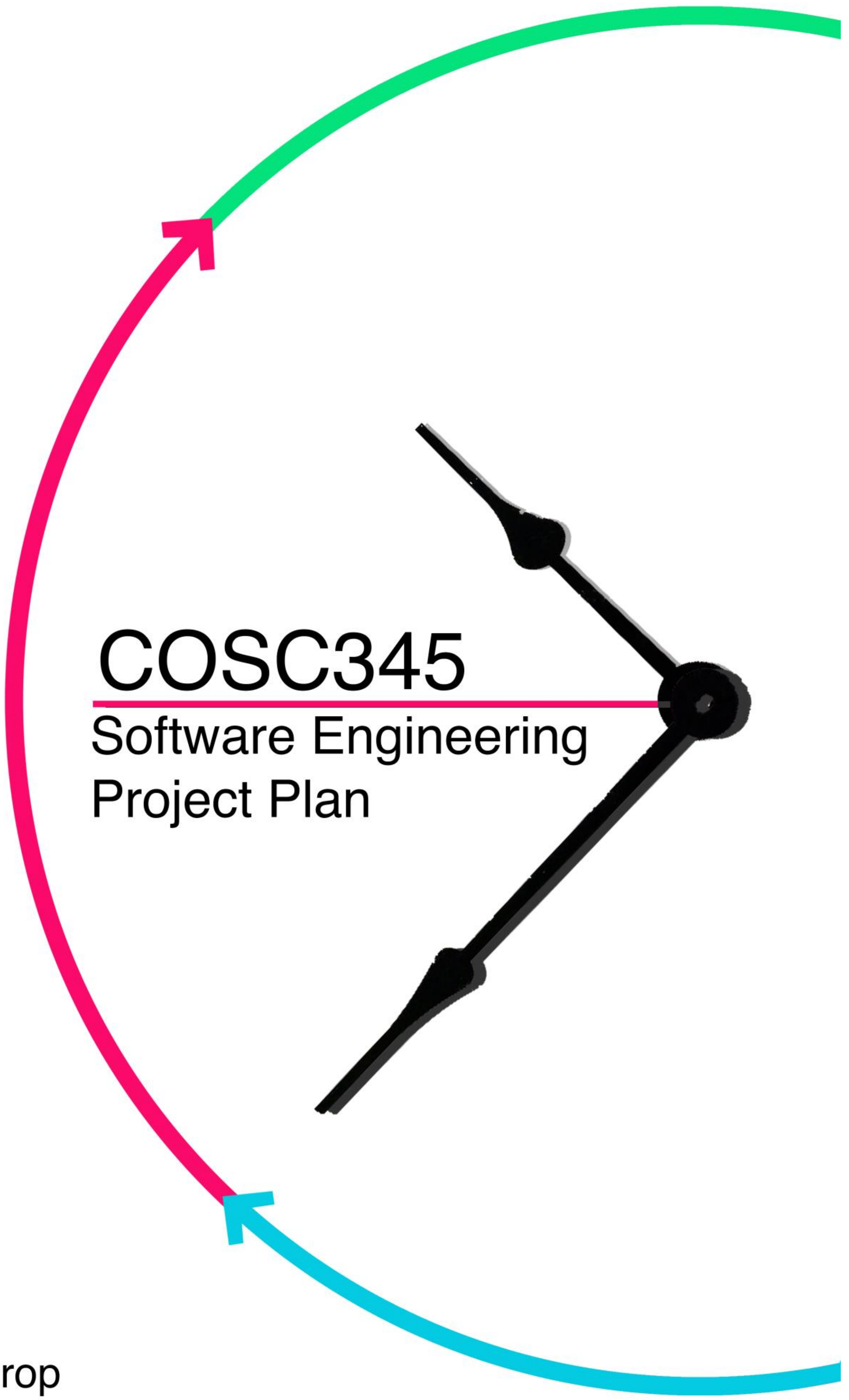


COSC345

Software Engineering
Project Plan

Alexis Barltrop
Meiqi Sun
Percy Hu
Tyler Baker

Preface

The following report is the original work of the following students as part of the COSC345: Software Engineering paper at the University of Otago.

Alexis Barltrop	ID: 7176303
Meiqi Sun	ID: 5926587
Percy Hu	ID: 2856476
Tyler Baker	ID: 1927791

Executive Summary

This report contains a proposal for an iOS app for iPhone that can manage repeated actions or habits. A core function of the app will be the ability to create a multi-segmented customisable timer for each action. This function could have many uses, we will primarily use it in the context of reminding the user to take regular breaks from sitting down while working at a desk or computer. This would be used nearly every day and would be useful in preventing long periods of sedentary behaviour, which can be a cause of wrist, neck and back pain as well as a risk factor in type 2 diabetes and heart disease.

The proposal includes step to build a piece of software that is dependable, efficient, maintainable and easy to use. The first semester of 2017 will be spent mainly in development, testing and eventual deployment of the app on the Apple App Store. Second semester will be dedicated to further maintenance of the app to make sure we have made an app with the desired attributes. The app will also receive its first update in second semester where we aim to introduce further functionality to the app.

Development of the app will be carried out using the XCode IDE which has a suite of tools and features designed for iOS development. Additionally, several software tools will be used for version control, continuous integration, communication and management of the project.

Contents

Introduction

Brushing our teeth is a habit we all do because we know that it will improve our long-term quality of life. Other habits such as exercise, drinking water frequently and avoiding long periods of sitting down can also improve our health. These habits, however, appear to be much harder to manage than brushing our teeth. A simple reminder to carry out these habits would benefit lots of people and be something you could use every day.

Every day, for students and office workers, much of their productive time is spent sitting down. Anyone who works at a desk will be familiar with the back pain, eye strain and other soreness that develop after a long time without changing position. The effect of prolonged periods of sedentary behaviour has also been reported as a risk factor in heart disease, type 2 diabetes and cancer.[1]

Desk workers can benefit from getting in to the habit of taking regular breaks to change from a sedentary position. Taking a few minutes every half an hour to get up and stretch or change position, can reduce the chance of strains and other injuries. A popular time management method, the Pomodoro technique follows a similar pattern of 20-25 minute working periods followed by small breaks of 3-4 minutes.[2] Taking these breaks as an opportunity to stretch and move out of a sedentary position is a good example of a habitual behaviour that can improve productivity and health and safety in the work/study space.

The personal smartphone is an excellent device for managing personal habits. The device is almost always on the person and can use vibration or ring alerts to remind the user to do something. Accelerometers in smart phones can detect if the user is sitting down, standing up or moving around. The GPS receiver in the phone can be used to track the user's positon and offer location based reminders. The ease and flexibility of software development means that it should be possible to make an application that can help to develop other habits such as exercise, drinking water, or taking supplements and medication.

In this report, we propose to develop and maintain a smart phone application designed to assist in the management of life improving habits. The application will include the ability to create customizable timers to assist the user to, for example, take regular breaks from sitting while working at a computer. The main criteria for this project is that the application will be used by the group members every day or at least every week. All the group members have daily routines involving spending several hours at a computer or desk and will benefit from regular use of our proposed application.

[1] A. Biswas et.al, 'Sedentary Time and Its Association With Risk for Disease Incidence, Mortality, and Hospitalization in Adults: A Systematic Review and Meta-analysis.', *Annals of Internal Medicine*, 2015, 162, p123-132.

[2] A. Henry, 'Productivity 101: A Primer To The Pomodoro Technique', 2014, available at https://www.lifehacker.com.au/2014/07/productivity-101-a-primer-to-the-pomodoro-technique/, Last accessed 13th April 2017.

Project Description

Purpose

The aim of this project is to design and build a software product that aids in the formation of habits that will improve the quality of life of the user. This includes the habit of taking regular breaks during long periods of sedentary behaviour.

User Description

The software product is aimed at university students, office workers and others who spend most of their productive time sitting down. This type of user is most likely to benefit from reminders to take breaks during long periods of sedentary behaviour. They are likely to have other habits, such as taking supplements or exercising, that they want to manage or improve on. The target audience also has access to a smartphone, which we will use as the platform to run the software product.

Platform

The software product will be designed to run on a smart phone device, specifically the Apple iPhone range. The iPhone is consistently the highest selling smart phone device and represents the largest target market for smartphone applications. The software will be written for iOS 10 which is supported by the most recently released iPhone models (See Table 1). User interaction occurs mainly through the touch screen input used on the iPhone. The app will be local and should not need to communicate with a server or cloud system to function completely other than for updating the software.

Description of Functionality

The software product will be developed incrementally, first developing and testing the core functionality of the segmented timer. Once this is complete we will add functionality for other habits and expand the manager. A mock-up of what the user interface could look like is shown in Figure 1. The extent to which we implement it depends on the time constraints and difficulty of the implementation. Ideas to further extend the application and make full use of the iPhone's capabilities are included here, but we will only attempt these once the previous functionalities are working as intended.

Core Functionality

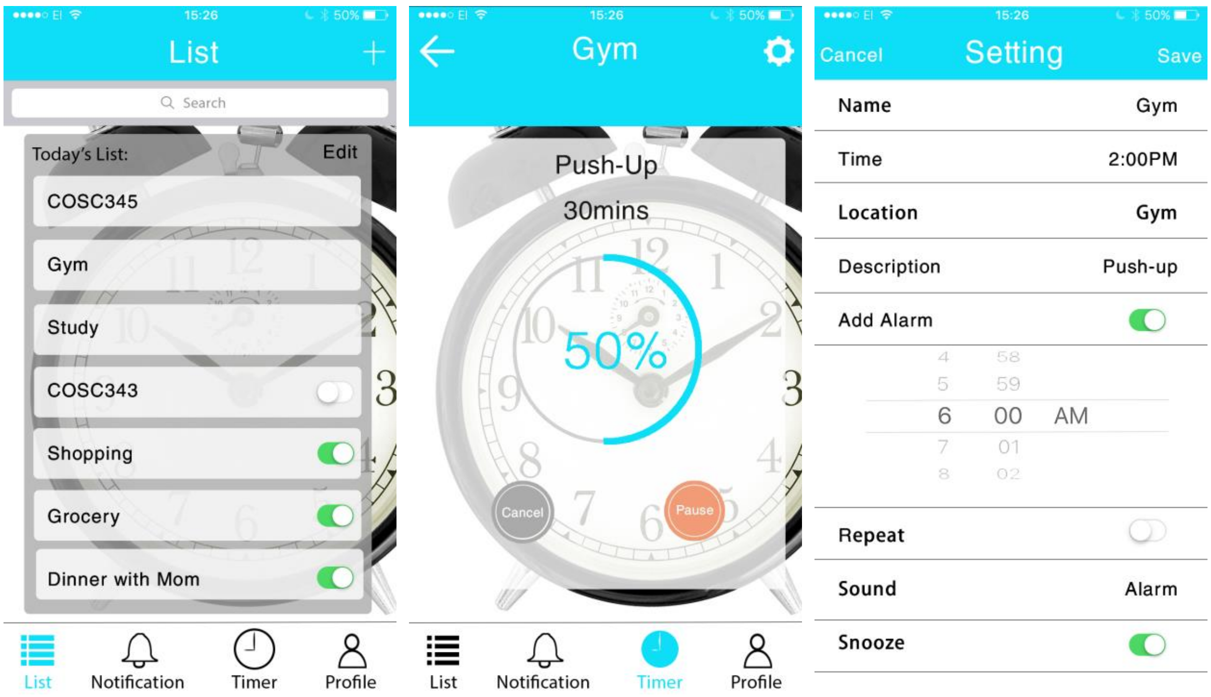
- Creation, use and editing of a segmented customisable timer i.e. a timer that can remind you to take a 5 minute break every 30 minutes and the user can adjust each timer length.

Extended Functionality

- Adding actions that don't necessarily require a timer.
- Keep track of what day of the week it is and what actions are relevant to that day.

Ideas For Further Extension

- Use the motion sensing capabilities of the iPhone to determine if the user is sitting or standing and adjust the timing of breaks accordingly.
- Location based reminders where the app reminds the user to take an action when they reach a certain GPS coordinate.
- Allow the user to add reminders and control the app using the Siri voice recognition software on the iPhone.



Today's List:

- COSC345
- Gym
- Study
- COSC343
- Shopping
- Grocery
- Dinner with Mom

Push-Up

30mins

50%

Name	Gym
Time	2:00PM
Location	Gym
Description	Push-up
Add Alarm	6 00 AM
Repeat	
Sound	Alarm
Snooze	

Figure 1. A mock-up of a possible user interface design. (left) Daily list of tasks/ habits to carry out. (middle) Timer interface. (right) Interface for editing timers and tasks.

User Requirements

A breakdown of the core and extended functionalities we hope to implement are included here. These requirements will be monitored and updated as the project progresses. The progress review will include a comparison between how our app is functioning and these requirements.

1. Creating a Segmented Timer (Core Functionality)
   1.1. The application will allow the user to create and customize a timer with multiple segments.
   1.2. Each segment should contain:
      a. A name.
      b. A length of time between 1- 120 minutes.
      c. An action to take at the end of the segment.

   Rationale: A segmented timer is necessary for a Pomodoro style routine. Standard times for the Pomodoro technique exist, but we decided that the user is the best person to decide what lengths of time to use. This will allow for flexibility to use the timer for activities other than work or study.

   1.3. The user should be able to create a list of timer segments and set them to repeat.

   Rationale: If the timer repeats the same pattern over an extended time then it is much more convenient for the user to set the pattern to repeat than to re-enter the pattern several times.

   1.4. The application will allow the user to start, pause and stop the timer

   Rationale: If there is a distraction or likely change in the strict workflow set by the timer then the user should be able to pause all functions of the timer where they are and continue when get back to their desk. At the end of the day the user should be able to stop or turn off the timer completely.

1.5. The application should alert the user when each segment has finished with either a ring tone or a vibrate action.

Rationale: The timer should notify the user at the end of each segment to remind the user that an action needs to be taken. The application may be used in a quiet space and so the user should be able to use the vibrate function of phone to remind them. It may be possible for the 'start break' and 'start work' alerts to be different.

2. Creating a Weekly Habit Reminder (Extended Functionality)
   2.1. The application should be able to create and schedule a reminder to action a habit.
   2.2. The reminder can be scheduled for a certain day of the week and /or certain time.

   Rationale: The days and the time that the application reminds the user to do a certain action should all be able to be set when first creating the habit. The product is specifically targeted at habits, done regularly every day or week. We are not creating a calendar for one off reminders at specified dates.

   2.3. The application should show a list of all the habits to be actioned on the current day.
   Rationale: A list of all the tasks needed to be done on the current day should be the first thing the user sees when opening the app. This allows the user to see what actions they need to do for the day quickly and efficiently.
   2.4. The application should allow each habit to be "crossed off" for the day.
   Rationale: When the action is complete the user should be able to turn off any alarms or reminders for that task for that day. The action of crossing an item off the list is also a simple thing that can encourage the user and give a sense of accomplishment.
3. General Requirements
   3.1. The app should be able to run in the background without impeding other activities on the phone.
   Rationale: The installation of the application on a smart phone should not affect the other functions of the phone, otherwise it will likely be removed and not used as intended.
   2.1. The app should either pause or mute during calls.
   Rationale: Alerts during a phone call may interrupt the call and be an annoyance to the user. Any alert occurring during a call should be postponed to the end of the call.

Key Attributes

The application should exhibit the following four attributes:

Usability

- The software should be intuitive and easy to use without a large amount of effort from the target user. The user interface should be appropriate for the platform and the target user. If deemed necessary a tutorial or adequate documentation should be available to aid the user.

Dependability

- The nature of the project is one that we hope will improve the quality of the users life and health by managing the formation of good habits. To do this it must reliably remind the user to perform the habitual actions and only the habitual actions that are selected by the user. To miss a reminder or turn off reminders unexpectedly would count as a failure of the software to meet its purpose. To this end the user must be able to depend on the software to function without fault due to problems inherit within the software.

Efficiency

- The software will be designed to operate in the background with only periodic interaction with the user. The mobile platform, while continuing to increase in capability, still has fixed limitations to its memory and processing power. The software should make efficient use of both memory and processor resources to ensure that the device is not unreasonably impeded by the application.

Maintainability

- The premise of this project is to maintain the software for the duration of the second semester. In this time, the functionality or layout of the software may change. Therefore, the software should be written in a way such that it can change and ideally that it is easy to do so. The software must be written and documented such that, in the case where a programmer is unavailable, another group member can understand and make constructive changes to the software.

Resource Requirements

Software

The IDE chosen for this project is the XCode IDE. The XCode IDE is developed by Apple specifically for making apps on Apple operating systems include iOS. XCode has a user interface builder where we can create the user interface in a visual manner, without knowledge of the Swift code that they are based on. The XCode iPhone simulator can simulate the running of our app on any previous iPhone model and will be a core tool in testing and development of the app. XCode also uses a compiler that is optimized to work with iPhone CPUs, helping achieve our target of efficiency.

The XCode IDE manages the building of executable files from our source code, so we do not require make or other software for the build. If the build is done correctly and the application works as expected on the simulator then we assume that the application is valid for use on an actual iPhone.

Version control for the project will be implement using Git with a repository hosted on github.com. All source code and related documents should be stored here. It is also a requirement of the project to submit the deliverables on GitHub.

We will use the Travis continuous integration platform to link to test our source code as it is deposited on the GitHub repository. Travis can interface with its own version of XCode and test the build of our app on several different iPhone models at once.

Communication between team members we be done predominately through email and messaging through the Discord application. Sharing and backup of public documents will be done using GitHub, any private documents will be managed through Google Drive. We will also be using the Trello web application to share and collate information about bugs that are found during testing or feature changes that need to be made. Meetings will be held on occasion especially when important decisions relating to the project are to be made. We expect most communication to occur through digital means to allow for group members to work when and wherever is most appropriate.

All visual design and graphics for the app will be done using Adobe Photoshop as per the preference of the designer. We will also be using a web application TeamGannt to manage the scheduling of the project.

Hardware

The iPhone

In this project we will use the XCode iPhone simulator for a majority of testing, however it will still be necessary to test our application on a physical iPhone running iOS 10. Testing on a physical iPhone gives us the chance to test the performance of the application when several other applications are running at the same time. The iOS 10 operating system is compatible with iPhone models iPhone 7, iPhone 7 Plus, iPhone 6s, iPhone 6s Plus, iPhone 6, iPhone 6 Plus, iPhone SE, iPhone 5s, iPhone 5c, iPhone 5. The application will only be developed for iPhones with 64-bit architecture which excludes models iPhone 5c and earlier. A short description of the relevant hardware is given in table 1.

Table 1. iOS 10 Compatible iPhones relevant to the project and their specifications.[3]

iPhone Model	Storage Capacity / GB	Processor (Max CPU Clock Rate)	RAM
iPhone 7 Plus	32, 128, & 256.	64-bit A10 Fusion chip (2.34 GHz)	3GB
iPhone 7	32, 128, & 256.	64-bit A10 Fusion chip (2.34 GHz)	2GB
iPhone6s Plus	32 & 128.	64-bit A9 chip (1.85 GHz)	2GB
iPhone 6s	32 & 128.	64-bit A9 chip (1.85 GHz)	2GB
iPhone SE	32 & 128.	64-bit A9 chip (1.85 GHz)	2GB
iPhone 5s	16, 32. & 64.	64-bit A7 Chip (1.3 GHz)	1 GB

Given the simple nature of the app we do not predict that there will be a large drain on memory or processor resources. We will still aim to make an efficient app, but the processing power of the phones means that small optimisations may go unnoticed by the user.

Development Hardware

The XCode IDE is exclusive to MacOS and we will require computers running MacOS to develop the application. In addition to the personal computers of group members, we also have access to computer suites at the university which contain many iMacs desktop computers available 24 hours, 7 days a week. XCode can run on a normal desktop computer and we do not predict the need for any other specialist hardware.

[3] Compare iPhone models, available at https://www.apple.com/nz/iphone/compare/, last accessed 13th April 2017.

Organisation

Our group consists of four members; each assigned a role to make best use of their different skills and backgrounds. During the project, we aim for all group members to develop a proficiency in C/C++ and Swift programming languages. Our group is small and it is likely that during the project all the group members will contribute in some way to each part of the project.

Programmers

1. Percy Hu

Percy is a third-year computer science student and the most experienced programmer in the group, having previously worked on projects in C/C++. Percy takes the main responsibility for the implementation of the software and writing documentation during the implementation.

2. Tyler Baker

Tyler also has C/C++ experience and will assist Percy in the implementation of the app. Tyler previously studied anatomy and physiology before computer science and will offer a different perspective to Percy. Tyler is responsible for leading the testing of the app. In the event Percy is unavailable, Tyler will be able to take the lead and ensure the implementation continues.

Editor

3. Alexis Barltrop

Alexis has minimal C/C++ experience but has completed an MSc in Chemistry and is a capable writer. It will be his responsibility to finalise reports and deliverables making them ready for submission. During the project, Alexis will also inspect documentation and review the progress of the project to make sure the project is on schedule.

Designer

4. Maggie Sun

Maggie is a Psychology student, now doing computer science. Maggie has experience in graphic design and is responsible for the design of the visual components for the app. She will work closely with the programmers to design and make the user interface for the app.

Project Break Down

The project is broken down into 6 areas Planning & Analysis, Development & Design, Testing, Deployment, Maintenance and Training. A breakdown of the tasks and milestones involved in each step can be seen in Table 2 which is visualised in the Gannt Chart in Figure 2 and Figure 3.

1. Planning & Analysis

The first task is to consider whether the application is feasible given the restraints imposed on us. There are several questions to ask with related tasks and milestones:

- What software & hardware do we need for developing iOS applications?
  - Milestone: Development software and hardware installed and working for all group members.
- What features and requirements do we want in our program?
  - Milestone: All group members sign off on user requirements.
- What similar products are available on the App Store?
  - Task: Review related Apps on App Store.
- How do we publish software to the App Store?
  - Milestone : Access to App Store publisher account.
- How do we join the two languages of C++ and Swift?
  - Task: Investigate joining C++ & Swift, teach all members of the group how to do it
- Can the project be down with a small team in a very limited time frame?

Progress reviews throughout the project will allow us to answer the last question. Comparing the schedule and plan in this report with what is actually being achieved should allow us to adjust our schedule if it is not feasible. Most the planning and analysis at this stage will have been completed before the submission of this report.

2. Development & Design

Development of the software product is to be broken down into a number of smaller size projects as discussed previously in the functionality description. Development of the core functionality is the most important first stage and should occur before attempts made to implement the extended functionality. At this stage the designer and the programmers will work together to create the user interface and decide how the implementation will proceed. Before work on the implementation starts, all group members must agree on the layout of the user interface.

At certain points in the development process the editor, Alexis, will perform a review to monitor progress, documentation, and any changes to the initial specification. Findings from these reviews will be reported back the group and any changes to the schedule made.

3. Testing

Pre-deployment Testing

Before the software product is launched on the App Store extensive testing will be need to be done to make sure it meets our requirements. A key assumption of the testing process is that the software is valid if it meets the user requirements when running in the iPhone simulator in XCode.

At each stage of testing we are looking to do three things:

- Unit Testing
    - Test each component class and function to ensure that each part is operating as expected. This can be done as soon as each piece of code is written.
    - Using code coverage tools to see how much of code is actual being run and identify possible bug locations. XCode has coverage tools to assist with this.
    - Ensure that our app is dependable.
- Performance Testing
    - Establish a baseline for memory and processor use and the performance of the app to that.
    - Ensure that our app is efficient.
- User interface Testing
    - Compare the user requirements to the actual function of the app.
    - Ensure that our app is easy to use.

Verification of the core functionality can be carried out at the same time that the extended functionality is being implemented. Ideally the testing of the core functionality and implementation of the extended functionality will be complete at the same time. Then while the programming team is debugging the core functionality, the testing team can test the extended functionality. In this way we hope to efficiently use the short amount of time we have.

XCode has several tools to assist with software testing and code coverage that we hope to make use of. Training and research into how to use these tools effectively is scheduled for the testers while the code is being written.

Ongoing Testing

Once we have launched the app on the App Store, we will continue to test the app and any updates or new functionality added to the app. At this stage, all the group members will begin using the app in their everyday routines to thoroughly test it and give feedback on the design. We may also have feedback from other users which we will act on at this stage.

4. Deployment

The deployment stage commences once we have a working software product that meets our user requirements. The main activity in deployment is to begin the application process for submitting our app to the Apple App Store. This involves reviewing the App Store Guidelines in a timely manner and ensuring out app follows them. Our second deliverable is a working version of our app, which will be the same version we submit to the App Store.

5. Maintenance

The main milestone of the maintenance phase is the release of a major update to our app. The update will include extended functionality, compatibility with the iPhone 8 and any changes based on user feedback.

In July, we expect a large update to be announced by Apple. This includes announcing iOS 11, iPhone 8 and Swift 4. Once the developer beta versions of iOS and Swift 4 are available we will have to dedicate a significant amount of time to compatibility testing and adaptation to make sure our application is still valid with the updated operating system and coding language. We aim to make these changes and release an update for the app to make it ready for the iPhone 8 when it is released in September.

Collection and analysis of user feedback will begin in June when university resumes and all the group member have a chance to use and test the app in everyday life. A full review of this will be included in the third deliverable; the usability report.

Any further functionality that we would like to add, for example the ideas mentioned in the project description, may be implemented here or alternatively there may be suggestions from users or other opportunities to extend the app. All group members must agree on the new features to be added in the update.

Once the update is released, the remaining time will be spent preparing the final deliverable, the finished software. We expect this to involve more testing, debugging and tweaking of the app to make sure it meets the user requirements and has the four key attributes we desire.

6. Training

The project is the first long term software engineering project for many of the group members. Learning to use all the software efficiently will be a major part of the development process. It is difficult to say how long it will take for each group member to learn a certain skill but we can still set some training goals.

1. All members of the group should be familiar with C/C++ and Swift languages by the end of first semester. This applies mostly to Maggie and Alexis who have less programming experience.
2. All members of the group should be familiar with XCode and how to use the testing and coverage tools available by the end of first semester.

Rationale: Training all group members in the basics of these languages and the tools in XCode, should allow the implementation, maintenance and debugging of the software to continue even if one of the programmers is unavailable.

Action: Maggie and Alexis have started weekly meetings to learn the basics of C/C++ together. We will use the chat client as a forum for sharing questions and answers we have during the learning process. During the review process, we will have a chance to meet and for the non-programmers to ask questions directly about parts of the code and also for sharing of any new knowledge about using the tools in XCode.

3. All group members should be familiar with the contents of the Swift 4 update before we start working on the first update to our app.

Rationale: We expect the Swift 4 update to potentially cause lots of problems and will benefit from multiple group members being able to work on different parts of update. This requires a retraining for all group members to use the new Swift code.

Action: A review of the contents of the Swift update will occur when the information is officially released by Apple. At this stage, we will meet and discuss the changes, how it affects the code and how we will deal with it.

It is the responsibility of each group member to communicate, in a timely manner, new knowledge that may be of use to the group and the project.

Table 2. Task breakdown for project. ID codes with M refer to milestone, those will D are deliverables, those without letters are tasks.

ID	Name/Title	Estimated Time / Days	Assignment	Dependencies
1	COSC345 Project : Habit Manager App	207		
1.1	Analysis	38		
1.1.1	Finalize Requirements for Software Product	4	Whole Group	
1.1.2	Finalize Requirements for Development	4	Whole Group	1.1.1
1.1.3	Install Software & Hardware Needed for Development	4	Whole Group	1.1.2
1.1.4M	Development Software & Hardware Installed & Working	N/A	Whole Group	1.1.3
1.1.5	Investigate Similar Apps on App Store	4	Whole Group	
1.1.6	Develop User Requirements	7	Whole Group	
1.1.7M	Group Agreement on User Requirements	N/A	Whole Group	1.1.6
1.1.8	Investigate Joining C++ to Swift	14	Percy , Tyler	
1.1.9	Figure Out How to Get App Onto App Store	2	Alexis	
1.1.10M	Access to App Store Developer Account	N/A	Whole Group	1.1.9
1.1.11D	Deliverable 1: Project Plan	N/A	Alexis	
1.2	Development and Design	29		
1.2.1	Design Class Library and Program Structure	7	Percy , Tyler	
1.2.2	Design User Interface	7	Maggie	
1.2.3M	Group Agreement on User Interface	N/A	Whole Group	1.2.2
1.2.4	Write Core Functionality in C++	11	Percy , Tyler	1.1.8, 1.2.3
1.2.5	Create User Interface in Swift	11	Maggie , Percy , Tyler	
1.2.6M	Documentation Inspection #1 & Progress Review #1	N/A	Alexis	
1.2.7	Write Extended Functionality #1	4	Percy , Tyler	1.2.4
1.2.8	Design App Thumb Nail	4	Maggie	
1.3	Pre-deployment Testing	18		
1.3.1	Testing Core Functionality	4	Alexis , Maggie , Tyler	1.2.4
1.3.2	Testing Extended Functionality #1	4	Alexis , Maggie , Tyler	1.2.7
1.3.3	Debugging and Suggested Changes From Testing	4	Percy , Tyler	1.3.1, 1.3.2
1.3.4M	Documentation Inspection & Progress Review #2	N/A	Alexis	
1.4	Deployment	35		
1.4.1	Investigate App Store Guidelines	4	Alexis , Percy , Tyler	
1.4.2	Make Changes To meet Guidelines	4	Percy , Tyler	

1.4.3	App Store Submission	18	Whole Group	1.3.3, 1.4.2
1.4.4D	Deliverable 2: Completed Software	N/A	Whole Group	1.3.3
1.4.5M	App Available on App Store	N/A	Whole Group	1.4.3
1.5	Maintenance	81		
1.5.1M	Full Project Review	N/A	Whole Group	
1.5.2	Collect and Analyse User Feedback	4	Whole Group	
1.5.3	Assess Compatibility of Software to iOS/ iPhone 8Â	4	Whole Group	
1.5.4	Identify Requirements for Update #1	3	Whole Group	1.5.2, 1.5.3
1.5.5M	Group Agreement on Content of Update #1	N/A	Whole Group	1.5.4
1.5.6D	Deliverable 3: Usability Report	N/A	Alexis	
1.5.7	Implement Changes for new iOS/ iPhone 8	4	Percy , Tyler	1.5.5
1.5.8	Write Extended Functionality #2	11	Percy , Tyler	
1.5.9M	Documentation Inspection & Progress Review #3	N/A	Alexis	
1.5.10	Debugging Extended Functionality #2	10	Percy , Tyler	1.5.8
1.5.11M	Update App on App Store #1	N/A	Whole Group	1.5.7, 1.5.10
1.5.12	Monitor User Feedback	11	Whole Group	1.5.11
1.5.13	Further Debugging	17	Whole Group	
1.5.14D	Deliverable 4: Finished Software	N/A		1.5.13
1.6	Ongoing Testing	32		
1.6.1	Core/Extended #1 Functionality Testing - Continued	11	Alexis , Maggie , Tyler	
1.6.2	Extended Functionality # 2 Testing	11	Alexis , Maggie , Tyler	1.5.8
1.7	Training	46		
1.7.1M	Learning Swift - Basic UI Design	N/A	Whole Group	
1.7.2M	Learning C++ - Basic Coding	N/A	Alexis , Maggie	
1.7.3M	Learn Testing & Code Coverage Tools in XCode	N/A	Whole Group	
1.7.4M	Learning Swift - Differences Between Swift 3 and 4	N/A	Whole Group	

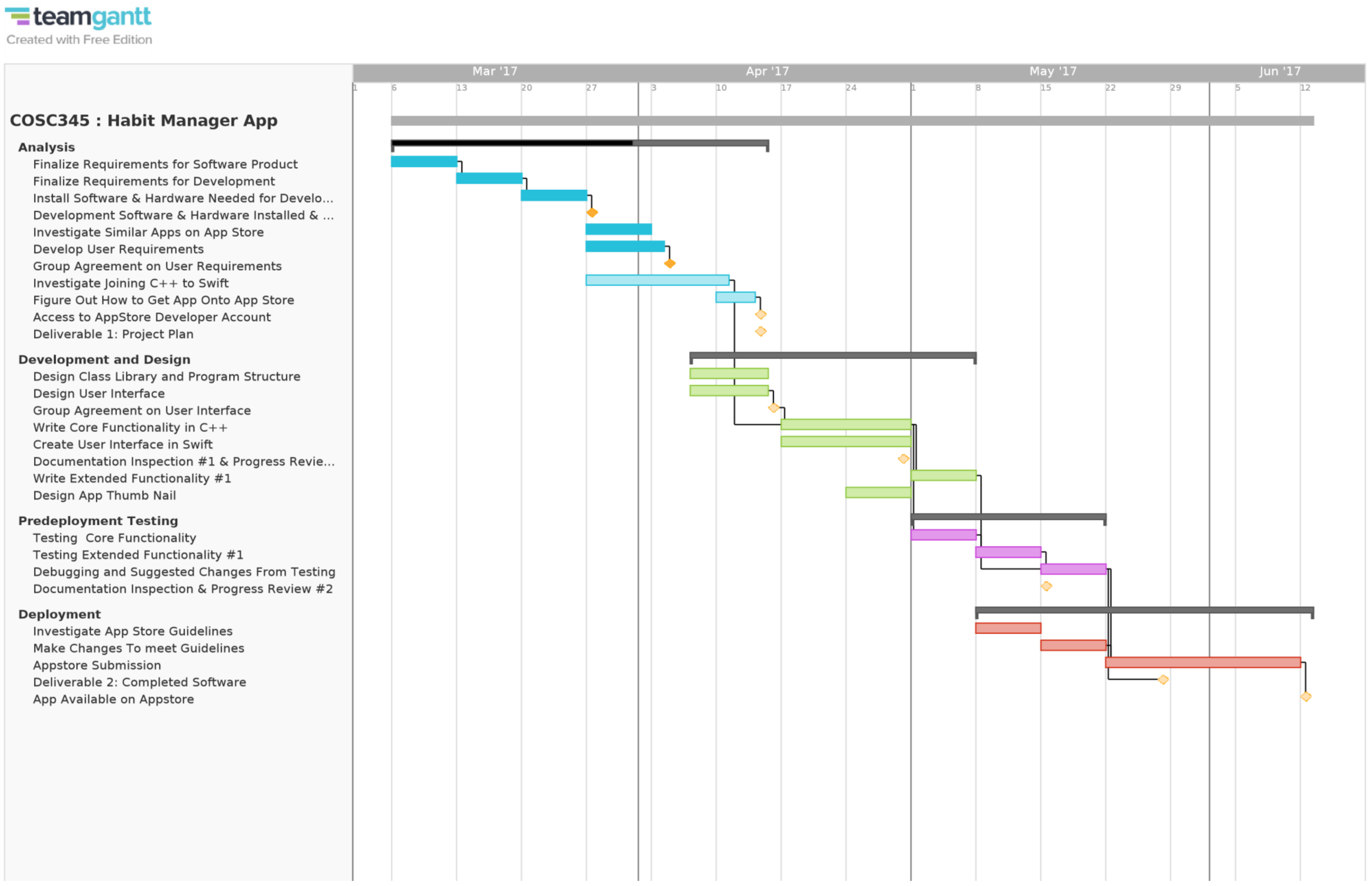


Figure 2. Gantt chart showing project schedule for first semester.

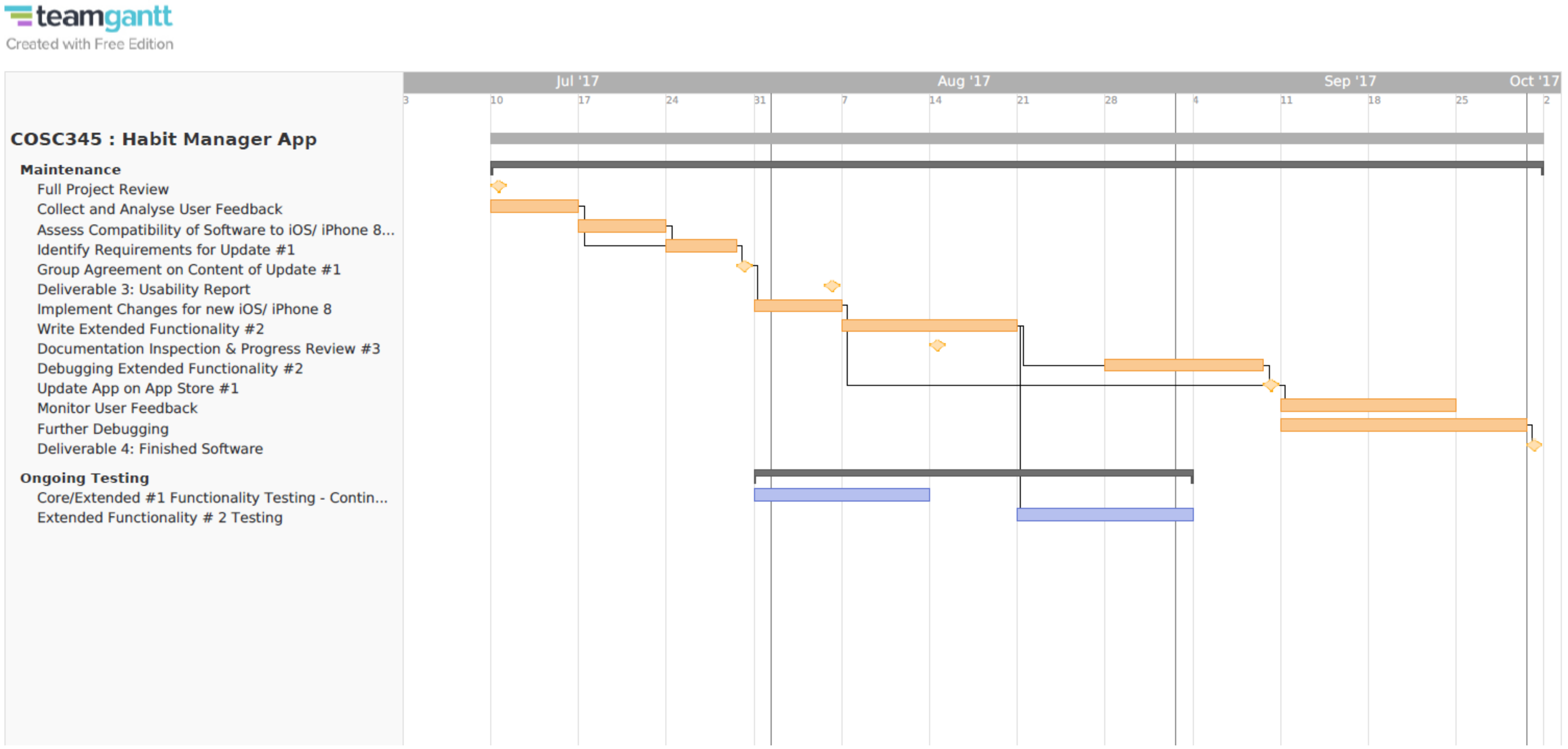


Figure 3. Gantt chart showing project schedule for second semester.

Risk Analysis

There are several factors that could delay the development of the application, they are summarised in Table 3. As part of progress reviews we will reassess these risks and how they will affect the project. Any new risks will also be raised and dealt with in these reviews.

The largest risk to this project lies in the inexperience of the group in developing software. We hope to mitigate this by using the best practice that we have learned in smaller projects. Practices such as incremental implementation, version control, continuous integration and writing tests and documentation before instead of after the code is finished. Regular progress reviews will help us to learn whether we have under or overestimated the difficult of our project and allow us to make the best use of the time before it gets out of hand.

Specialisation of work within the group will help us to make use of the little time that we have. Unfortunately, it also introduces the risk that someone with knowledge critical to the project may become unavailable at a critical time. We have set aside time for those with minimal programming experience in the group to learn C/C++ and Swift while the programmers write the code. Hopefully by the time the project is underway they will have the skills to assist in the maintenance and implementation if the programmers become unavailable. Proper documentation and communication will also help the project progress smoothly in this situation.

Hardware failure of any personnel computer could result in the loss of work and the inability to do further work. Using cloud based version control on GitHub will greatly alleviate the risk of losing work due to hardware failure and to accidental overwriting. All files being worked on personal hardware must be backed up to the GitHub repository or Google Drive. In the event a group member's computer breaks, the computers available to us in the computer science building will be used as a backup. These are maintained by the department with staff and resources devoted to their continued operation. In our group, Alexis regular works at these computers and so is responsible for checking that the version of XCode they run is up to date and will build our app as expected.

Reconfiguring our app to work with iOS 11 and Swift 4, when Apple releases the update, may require a large amount of time. Introducing iOS11 compatibility to our app can only start when we know the details of the developer release in July. This task will get priority over introducing new features to help us avoid underestimating how long it will take.

Table 3. List of risks to the project.

Risk ID	Risk	Description	Action
1.	The time required to develop the software is underestimated.	Software development is notorious for going overtime. The group has little experience with software development.	Carry out good practices that we know e.g. version control, continuous integration, document as we go.
2.	Work lost due to hardware failure or deletion.	A computer crash or accident overwrite could result in loss of valuable work.	All work backed up in cloud whether by Google Drive or GitHub.
3.	The ease of joining Swift and C++ is more difficult than expected.	This is something no one in the group has done before and in critical to our project development	Percy to investigate how to do it and teach other group members as the project goes on.
4.	One or both programmers are ill or unavailable at a critical time during development.	At the start only Tyler and Percy have experience with C++. Their absence will reduce the available experience within the group and slow development.	Maggie and Alexis to learn the basics of C++ and swift. Documentation to be kept in a state where the code can be edited easily if one of the programmer is away.
5.	Changes to requirements changed dramatically	The requirements for the project are given to us by the coordinators of the course and could change unexpectedly.	Clear communication and development of skills over the course of the development will hopefully allow us to adapt easily. However without knowledge of what can change it is hard to take action.
6.	Updates to iOS and Swift render the software invalid.	It is expected that a new version of iOS will be released in September 2017 along with a new iPhone and Swift update.	Developers should have access to the iOS sometime in July 2017 before the public release. Updating our app will be a priority at this time.
7.	The personal computers belonging to group members break.	The damage or theft of a group member's property may result in a loss of work and prevent further work on the project.	We will use version control and cloud storage so that all files are available even if property is lost. As a backup all group members have access to the facilities at the computer science building. These facilities have most of the software that we need and have staff employed to keep them working.

Project Schedule

The project has four deliverables with fixed deadlines outlined in Table 4. Our schedule of the tasks discussed in the project break down, and the schedule for these is shown previously in the Gannt Chart in Figure 2 and Figure 3. A short summary of the schedule for each part is given here.

Table 4. Deadlines for deliverable in the project.

Deliverable	Deadline
Project Plan (This report)	13th April 2017
Working Version of Software + Documentation	29th May 2017
Usability and Improvement Report	4th August 2017
Working, Debugged Software + Documentation	29th September 2017

Analysis & Planning

The first six weeks of the project are spent carrying out planning tasks and in the preparation of this report.

Development and Design

The second deliverable is the working version of the software which is due on the 29th of May, giving us six weeks from the first deliverable. The deadline is fixed and we predict the work will expand to fill the time right up to the deadline. The incremental nature of our development plan means that we can fill the time available with extended the functionality or if the project is delayed we can focus on the core functionality and deliver that in a working state.

Testing

The testing of the product will occur simultaneous while it is being developed. This allows us to make efficient use of people in the group with less programming experience and get more work in before the fixed deadline. The design of the program should ensure that the interfaces between core and extended functionalities are rendered invalid by any changes or debugging in the core code.

Deployment

Deployment cannot start until development and testing are well underway but we can start investigating the App Store guidelines earlier on and make the necessary changes in a timely fashion.

Maintenance

Maintenance starts in second semester, the first deliverable is a report on the usability of the software. The focus of the first part Is testing and getting user feedback for the application. After the submission of this we expect the apple update to be occurring and we will have to work on that as well as releasing another update.

Training

Training activities will occur throughout the duration of the project and do not have fixed deadlines but will monitor progress and make sure that relevant knowledge is being shared.

Monitoring and Reporting

In the schedule, we have set milestones as concrete goals to monitor the progress of our project. These milestones can be seen in the schedule of tasks in Table 2 and have been discussed in the project breakdown.

Progress review will take place regularly to monitor the project. These reviews will cover the following points:

- Are we on schedule?
    - Compare the actual and predicted date of completion for each task and milestone.
    - Make any adjustments to the schedule as necessary.
- Are the user requirements being met?
    - Compare the function of the app to our user requirements.
    - Formalise any new requirements or functionality.
- Are there any new risks to consider?
    - Review the management of known risks and address any new ones.
- Are we using our time efficiently?
    - As group members pick up new skills we may be able to distribute tasks more efficiently.
- What new knowledge is there?
    - Any new information that may be important to the project should be shared, as well as questions asked if a group member unsure about something.
- Are there any new issues or changes to the project?
    - Changes to the project restrictions, features or user requirements.

The delivery of reports and deliverables shall be done through the GitHub repository at the following address:

https://github.com/powerofpercy/COSC345-iOS-App

Conclusion

This report details our plan to create a smart phone software product that can help to the user manage and carry out life improving habits. The development will occur in a number of stages with incremental implementation and testing. Once we have a working version of the software, we will maintain it for the duration of second semester and release an update containing bug fixes, compatibility changes and new features. There are many risks that may delay our project, but through regular monitoring and efficient use of resources we should be able to adapt and deal with these risks. Regular monitoring, clear communication and following software engineering best practice, will also ensure that our software product displays the attributes of efficiency, maintainability, dependability and usability.