[Date]

[Document title]

[Document subtitle]

Alexis Barltrop

Maggie Sun

Percy Hu

Tyler Baker

# Executive Summary

We propose to plan and build 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. We wish to build such a function primarily as it will allow the app to remind 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.

We aim 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 and testing of the app. We aim to have it launched on the App store at the end of first semester. Second semester will be dedicated to further testing 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.

PandaTip: The executive summary should be between 150-600 words and should summarise the remainder of the report without preempting it. It is usually written last of all when you have a clear idea of the proposal as a whole. The aim of this section is to engage the readers’ interest and encourage them to read the rest of the software development proposal. You should include a helicopter view of the project including the software project name and objectives as well as a general mention of the technology. Don’t go into too much detail or include price (as both are included elsewhere). Also remember that the reader may not be technically savvy so avoid losing their interest with too much technical detail which can be included later.

# Contents

Contents

[Executive Summary ii](#_Toc479664728)

[Contents 1](#_Toc479664729)

[Introduction 2](#_Toc479664730)

[Project Description 4](#_Toc479664731)

[Key Attributes 7](#_Toc479664732)

[Resource Requirements 8](#_Toc479664733)

[Hardware 9](#_Toc479664734)

[Software 8](#_Toc479664735)

[Organisation 10](#_Toc479664736)

[Project Break Down 11](#_Toc479664737)

[Risk Analysis 17](#_Toc479664738)

[Project Schedule 19](#_Toc479664739)

[Monitoring and Reporting 20](#_Toc479664740)

[Conclusion 21](#_Toc479664741)

# Introduction

Every day we wake up and brush out teeth because we know it is a habit that will increase our quality of life. There are other habits that also increase quality of life, exercise, eating healthy, breaking sedentary behaviour and while tooth decay has declined diseases such as back pain , diabetes continue to increase.

One growing problem is the increase of extended periods of sedentary behaviour in our lives. For office workers and students most of our time is spent sitting down at a desk or computer with little incentive to change working positions. The effect of prolonged periods of sedantry behaviour has been shown to be detrimental to health in a number of different ways. Studies citing it as a risk factor in heart disease, type 2 diabetes and cancer are increasingly showingthat the more time spent sitting down the more likely one of this conditions is going to develop or the worse it will be.[[1]](#footnote-1)

A solution to this problem is to form a habit of taking regular breaks from sitting down. As simple a task as walking around the office for a few minutes allows the body to get out of a sedantry position and reduce the chance of back injury and other related injuries.

Regular breaks can also improve productivity and assist in concentration. The pomodoro technique favors productive working phases of 20-25 minutes followed by small breaks of 3-4 minutes.[[2]](#footnote-2) Every fourth or fifth break is usually longer, around 20-30 minutes. Taking these breaks as an oppourtunity to stretch and move out of a sedentry position is good example of a habitiual behavour that can imporve both health and saftey in the work/study space and productivity as well.

The personal smartphone is an excellent device for manager such habits. It is almost always on the person and can monitor movement and activity as well as use tones or vibration as a reminder to the user. The ability to easily make applications for it means that making a customizable timer for a Pomodoro routine should also be possible. A user wishing to develop other habits such as exercise, practicing a musical instrument, drinking water, taking supplements or medication could also benefit from using a smart phone to manage these.

In our surveying of the app store we found many app that manage habits and a few timers with nonuniform intervals. However, there is no habit manger that lets you create a customizable timer for a specific activity.

In this report, we propose the development of a smart phone application designed 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 workstation. The main criteria for this project is that the application will be used by the group members every day or at least every week. Our proposed application would be used regularly by all the group members, we all have some sort of habit that we wish to work on. In addition, we also spend most of our time at a computer or desk whether in study or in leisure and would greatly benefit from structured routine that considers breaks from sitting down.

In accordance with the other criteria for this project the application will be written in C/C++ with only the user interface using Swift. Once the application is completed at the end of first semester we plan to maintain it for the remainder of second semester.

Cite some studies to look real professional

$$$ What citation method?

Introduce term for ergonomic breaks

Pomodoro

<https://en.wikipedia.org/wiki/Pomodoro_Technique>

https://www.focusboosterapp.com

Sitting down

<http://www.health.harvard.edu/blog/much-sitting-linked-heart-disease-diabetes-premature-death-201501227618>

Ergonomic calculator

http://www.ergotron.com/tools/workspace-planner

# Project Description

### Purpose

This 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. For example, taking regular breaks during extended periods of computer usage to reduce back pain.

### User Description

The software product is aimed at university students and office workers. A user of this type typically spends most of their productive time, at work or in study, sitting down reading on paper or a monitor screen. 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. 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 smartphone is an ideal device for a habit manager as it is almost always on and always close by to the user. The software will be written for iOS 10 which is supported by the iPhone models 7,6 etc. The limitations of these devices are discussed later. The app will be local and should not need to communicate with a server or cloud system to function completely. The user interface is then restricted to the touch screen input used on the iPhone.

### Description of functionality

The software product will be developed incrementally first developing and testing the core functionality in two parts. The extent to which we implement it depends on the time constraints and difficulty of the implementation. We have included ideas here for extended functionality that would like to try and implment if time and resources permit.

#### Core Functionality

1. 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

1. Adding actions that don’t necessarily require a timer.
2. Track of what day of the week it is and what actions are relevant to that day.

#### Functionality That Would be Super Cool to Have

* Use the motion sensing capabilities of the iPhone to determine if the user is sitting or standing and adjust the breaks accordingly.
* GPS tagging of each habit so that the application can ask you if you wish to action that habit when you get to a certain place.
* Use Siri voice activation kit to schedule habits.

### User Requirements

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

*Rationale*: A segmented timer is necessary for a Pomodoro style timer. While standard times for breaks and work do exist, we decided that the user is the best person to decide what lengths of time to use. This flexibility also allows the timer segments to be used for different activities to study/work. A standard Pomodoro template may be included.

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

*Rationale*: The timer should reliably action 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.

1. Creating a Weekly Habit Reminder (Extended Functionality)
   1. The application should be able to create and schedule a reminder to action a habit.
   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 frist creating the habit. The product is specifically targetted at habits, done regulary every day or week. We are not creating a calender for one off reminders at sepcified dates.

* 1. 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.

* 1. 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 alrams or reminders for that task for that day. The action of crossing a item of the list is also a simple thing that can encourage the user and give a sense of accomplishment.

1. General Requirements
   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 or it is most likely the applciation will be removed and not used as intended.

* 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 occuring during a call should be postponed to the end of the call.

## Key Attributes

The software and any upgrades in functionality 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 other functions and software in the device is not unreasonably slowed down or made inoperable.

### 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 specifcally for making app on Apple operating systems include iOS and includes many useful features. 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 exectuable 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 repositoy 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.

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

A number of web applications shall be used for communciation between team members. In group messaging will be done through email and discordapp.com. Sharing and backup of private documents relating to the project shall be done using Google drive. We will also be using trello 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 ocassion especially when importnt decisions relating to the porject are to be made. A majority of the communcaiton, however, will occur thorugh the digital means to allow for group members to work when and wherever is most appropriate.

Other software may be used for special tasks . 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 applcation TeamGannt to manage the task and scheduling for the project.

## Hardware

### The iPhone

In this project we will use the Xcode iPhone simulator(See below) for a majority of testing, however it will still be neccesary to test our application on a physical iPhone running iOS 10. Testing on a phsical 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.[[3]](#footnote-3) A short description of their hardware is given in table 1.

Table 1. iOS 10 Compatible iPhones and their specifications.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

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 high speed and large memory means that small optimisations ma 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. The programmers and the designer all have personal computers runnnig MacOS Additionally 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 need any hardware other than the personnel computers metnioned.

# 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

* 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.

* 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 project continues.

## Editor

* Alexis Barltrop

Alexis has minimal C/C++ experience but has completed an MSc in Chemistry and is a capable writer. It will be Alexis’ 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

* Maggie Sun

Maggie is a Psychology student, now doing computer science with minimal C/++ experience. Maggie has experience in graphic design and responsible for 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 $$XX which is visualised in the Gannt Chart in Figure XX$$.

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 scheudle if it is not feasible. Most the planning and analysis at this stage will have been completed before the submission of this report.

1. Development & Design

Development of the software product is to be broken down into a number of smaller size projects. 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, Maggie, and the programmers Percy and Tyler will work together to create the user interface and decide how the implementation will proceed. Before work on the implementation starts all members f the group 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.

The Xcode IDE handles the build of the software and we assume that if the software builds without errors and works on the XCode iPhone simulator then the application will work on an actual iPhone.

Checklist code complete page 773

1. Testing

Testing of the software can be divided into two parts – Pre-deployment and Ongoing. 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. Once the core functionality is complete, Maggie and Alexis will begin testing the application using Xcodes built in iPhone simulator. A key assumption of the testing process is that if the software meets the user requirements when running in the iPhone simulator in Xcode then it should behave as expected on a physical iPhone. Xcode has simulators for all the models of iPhone that we plan to design our application for.

Each testing procedure should cover three areas:

* Unit testing
  + Testing the component classes and funcitons of the programs to ensure that each part is operating as expected.
  + Using code coverage tools to see how much of code is actual being run.
  + This is where we want to make sure our product is dependable.
* Performance Testing
  + Establishing a baseline memory and processor use and comparing its performance related to that. This is where we must make sure our product is efficient.
* User interface Testing
  + Are all the user requirements expressed as parts of the user interface?
  + Is the design intuitive and easy to use?

Verification of the core functionality can be carried out by Maggie and Alexis at the same time that the extended functionality is being implemented by Percy and Tyler. Ideally testing of the core functionality and implementation of the extended functionality will be complete at the same time so that the programming team can work on debugging the core functionality while the testing team tests the extended functionality and efficiently use the short amount of time we have.

Xcode has a number of 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 effecitively is scheduled for the testers while the code is being written.

Ongoing testing of the software includes testing new functionality and reciveiing feedback from users. The completetion of the coreOnce the app has been placed on the App store we should get other people using the software and testing it for us.

1. Deployment

Test Flight Beta? Testing App.

The deployment stage commences once we have a working software product that meets our user requirements. The main activity in deployment is launching the Application on the Apple App store. We have budgeted a week to make sure that our application meets the App Store Guidelines. Prior to this we have a deeper investigation of these guidelines, which should feed back into the development and hopefully speed up the submission process. The milestone for this is having our App available on the app store. The second deliverable is a working version of our App and so this should be essentially the same milestone.

1. Maintenance

The maintenance and updating of our software starts at the beginning of second semester. For the first part of June we aim to gather feedback from user testing, as well as develop further functionality for the first update of the software, as described in the functionality overview.

A full review of the software will occur at the beginning as well, comparing the product to our initial user and system requirements and any feedback from the examiners.

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 significant period of time to compatibility testing and adaptation to make sure our application is still valid with the update operating system and coding language. Any changes should be ready and we aim to have an application that should work for the iPhone 8 when it is released in September. The only way we will be able to test it is with the simulators on xCode.

After that period we will attempt to implement any further features that we want to include and continue to debug and test the application as much as possible.

Review user interface and any changes to functionality we want.

Multi stage development with at least one update to the app store. By this stage at least one of the other group members should be able to

6. Training

The project invovles a lot of work that is new to many of the group members. To ensure that the group memebers can do the best that they can and devleope skills and to introduce skill reduncdancy in case something goes wrong we have a extensive triaing plan. The whole project will hopefully be a good lesson in software engineering for all the group.

The Swift language is new to all the group members and we will all aim to have a basci proficiency before the launch of the software. Maggie and Alexis need to get a general proficiency in C++ as well so that they can understand the code and helpe fix it in the debugging stage.

The main tool of Xcode will take some getting used to and has many useful tools for debugging, testing and writing code. We will specfiy certain times to meet and share knowledge on using this new software.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task ID | Description of Task/Milestone | Estimated Days to Complete | People Assigned to Task | Dependencies |
| 1 | 345 |  |  |  |
| 1.1 | Analysis |  |  |  |
| 1.1.1 | Finalize Requirements for Software Product |  |  |  |
| 1.1.2 | Finalize Requirements for Development |  |  | 1.1.1 |
| 1.1.3 | Install Software & Hardware Needed for Development |  |  | 1.1.2 |
| 1.1.4 | Development Software & Hardware Installed & Working |  |  | 1.1.3 |
| 1.1.5 | Investigate Joining C++ to Swift |  | Percy/Tyler |  |
| 1.1.6 | Figure Out How to Get App Onto App Store |  | Alexis |  |
| 1.1.7 | Access to AppStore Developer Account |  |  | 1.1.6 |
| 1.1.8 | First Deliverable Project Proposal |  | Alexis |  |
| 1.2 | Development and Design |  |  |  |
| 1.2.1 | Design Class Library and Program Structure |  | Percy/Tyler |  |
| 1.2.2 | UML Diagram Complete |  |  | 1.2.1 |
| 1.2.3 | Design User Interface |  | Maggie |  |
| 1.2.4 | Agree on User Interface |  |  | 1.2.3 |
| 1.2.5 | Write Core Functionality in C++ |  | Percy/Tyler | 1.1.5, 1.2.2, 1.2.4 |
| 1.2.6 | Documentation Inspection #1 |  | Alexis |  |
| 1.2.7 | Write Extended Functionality #1 |  | Percy/Tyler | 1.2.5 |
| 1.2.8 | Documentation Inspection #2 |  |  |  |
| 1.2.9 | Create User Interface in Swift |  | Maggie , Percy/Tyler |  |
| 1.2.10 | Design App Thumb Nail |  | Maggie |  |
| 1.3 | Predeployment Testing |  |  |  |
| 1.3.1 | Test Core Functionality - Code Coverage |  | Alexis , Maggie | 1.2.5 |
| 1.3.2 | Test Core Functionality - User Interface |  | Alexis , Maggie | 1.2.5 |
| 1.3.3 | Test Extended Functionality #1 |  | Alexis , Maggie | 1.2.7 |
| 1.3.4 | Debugging and Suggested Changes From Testing |  | Percy/Tyler | 1.3.1, 1.3.2, 1.3.3 |
| 1.4 | Deployment |  |  |  |
| 1.4.1 | Launch App on Appstore |  |  | 1.3.4 |
| 1.4.2 | Second Deliverable: Completed Software |  |  | 1.4.1 |
| 1.5 | Training |  |  |  |
| 1.5.1 | Learning Swift - User Interface Design |  | Alexis , Maggie , Percy/Tyler |  |
| 1.5.2 | Learning C++ - Basic Coding |  | Alexis , Maggie |  |
| 1.5.3 | Tutorial on Swift/C++ Linking |  |  |  |
| 1.6 | Maintenance |  |  |  |
| 1.6.1 | Second Semester Begins |  |  |  |
| 1.6.2 | Assess Compatibility of Software to iOS/ iPhone 8¬† |  |  |  |
| 1.6.3 | Implement Changes for new iOS/ iPhone 8 |  |  |  |
| 1.6.4 | Identify Requirements for First Update |  |  |  |
| 1.6.5 | Write Extended Functionality #2 |  |  |  |
| 1.6.6 | Implement Patches/ Bug Fixes |  |  |  |
| 1.6.7 | Update App on App Store #1 |  |  |  |
| 1.7 | Ongoing Testing |  |  |  |
| 1.7.1 | Core/Extended #1 Functionality Testing - Extended Coverage |  |  |  |
| 1.7.2 | Extended Functionality # 2 Testing |  |  |  |
| 1.7.3 | Download¬† |  |  |  |

# Risk Analysis

A number of possible risks to the project are described in Table 2. The largest risk to this project lies in the inexperience of the group as a whole in developing software and writing in C++. Dealing with this is done with specialisation of roles. The two group members with experience in C/C++ are responsible for the implementation of the code. This should leave the other two members to take care of admin and design aspects of the project. Training and education for all group members is important to increase redundancy. People in the group may have other assessments due at critical times and may not be able to attend to important issues. Thus all members of the group aim to develop profienciy in C++ and Swift by the time the software is placed on the App Store. Maggie and Alexis have scheduled weekly time to develop these skills.

The risk of inexperience will be most present in the planning phase. It is easy to imagine that we may underestimate the time it takes to develop the software. Taking this into account we take an incrmental approach to design. The core funcitonality is the first thing to be implemented and extended funcitonality can be pushed back if the core development overshoots.

Version control and storing files on the cloud helps alleviate the risk of conflicting version of software from different people and also due to hardware failure. All files written on personal hardware must be backed up to the github repository or google drive. In the event that personal hardware fails, the back up is the copmuters in the computer science building. These are maintained by the department with staff and resources devoted to there operation. These machiens have mac and Xcode, however we can’t install other software on them.Thus most of communcation and management software a web platform and can be used without installing software.

Apple is expected to realease a new version of iOS and a new iPhone 8 later in September of 2017. Developer releases for this is expected in July 2017 and we aim to update our software to be compatible with these new releases. We don’t know what they will contain but expect a beta release of iOS 11 in June/July. We aim to address this compabiltiy in second semester before triyng ot extend the functionality further.

Table . List of risks to the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | Risk | Description | Action | Monitoring |
| 1. | 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. | Alexis to check every one can do it by second week of development. |
| 2. | 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. | Plan development times aware from holidays or scheduled absences |
| 3. | Changes to requirements changed dramatically | The requirements for the project are given to us by the coordinators of the course and could change unexpectedly. | Cleat 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. | Check emails got to lecture. |
| 4. | The time required to develop the software is underestimated | Software development is notorious for going overtime and we are all inexperienced at making software. | We aim for incremental development to make sure the core functionality is working first before trying to implement more complicated functionality. |  |
| 6. | A new version of iOS is released. | It is expected that a new version of iOS will be released in September 2017 along with a new iPhone. | Developers should have access to the iOS sometime in July 2017 before the public release. Hopefully we will be able to get hold of this and test our application with simulators in xCode. | Investtigate how to get developer access and what new features may help or hinder the use of our application. |
| 7. | The personal computers belonging to group members break. | The damage or theft of a group members 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 stolen. 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. | Alexis works mainly on the computer science computers and so will quickly find out if any problems with them and will alert staff before a critical time. |

# Project Schedule

The project has four deliveralbes with fixed deadlines outlined in Table 3. Our schedule of the tasks discussed in the project break down, and how we aim to meet these deadlines are are shown in a Gannt Chart in $$XX.

Table . 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 prepaation of this report.

#### Development and Design

The second deliverable is the working version of the software which is due on the 29th of May. 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 availbile with extended the funcitonality or if the project is delayed we can focus on the core fucntionality and deliver that in a workign state.

##### Testing

The testing of the product will occur simultaneous while it is being developed. This allows us to make efficeitn 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 neccesary changes in a timely fashion.

#### Maintenance

Maintenacne starts in second semester, the first delievrable is a report on the usabiltiy 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 occuring 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 sur ehtat relevant knwoldege is being shared.

# Monitoring and Reporting

Each phase in the project has a number of milestones that we will use to monitor whether or not we are on track. The TeamGannt web application will be used to manage these milestones and can email reminders to the team when deadlines approaching.

The most critical phase to monitor in the project is the design and development stage where the software will be written and tested. The approach to monitoring this stage is to break in down into a number of incremental stages which can be completed in at most a week. During the development stage Alexis is responsible for monitoring and inspection the code to make sure that style and documentation guidelines are consistent and up to standard for submission. A checklist for style and documentation is attached in the appendix.

For tasks in the analysis stage there are some things which are relevant to all group members and so the reporting for these will be done at a team meeting. Important knowledge such as the baiscs of linking swift and C++ will be taught to everybody in the group to reduce the risk of that knowledge being lost should one of the members become unavailible at a critical stage. Any other crictical information like this will be reported in a similar manner.

The small size of the group lends itself to face to face communication. The number of formal reports will be kept to any absolute minimum instead favouring open commmunication between the group. Discord chat client is being used to communicate and record chat. For milestones where agreement of the whole group is required to finalise an important design or part of a project then a group meeting will be held and mintues kept. The mintues will be stored on the github repository.

In the testing and maintenance phases the important aspect will be in reporting of bugs and monitoring whether the bugs have been fixed. Ulitimately we want everybody to test the software. To this end we use Trello a web application. Bugs and related material i.e. screen shots, test data etc can be added here and made availible to the programmers. This software sends out notificaitons when any new bugs have been reported.

How the project will be monitored

When reports are to be delivered.

Monitoring done via inspections milestones

Progress monitored by dividing up into managable chunks. Looking for milestones not percent completion.

Use standard checklist to inspect code.

How are we going to monitor progress and what reporting will there be?

Milestones along the way.

# Conclusion

Appendix – Checklists

COSC345 Commenting/Documentation Checklist

Style checklist

General

❑Is formatting done primarily to illuminate the logical structure of the code?

❑Can the formatting scheme be used consistently?

❑Does the formatting scheme result in code that’s easy to maintain?

❑Does the formatting scheme improve code readability?

Variables

* Do all variables have meaningful names?
* Are they used only for the purpose they are named?
* Are constants used instead of magic values?
* Are enumerated types used instead of encoded values?
* Any relationships between variables that could be simplified?
* Are variables declraed with correct scope?
* An global variables need to be local or vice versa?

Individual Statements

* Does each line contain at most one statement?
* Is each statement written without side effects?
* Is there at most one data declaration per line?
* Does error handling fix the problem or stop the program?
  + E.g. no continuation of code after catch blocks.
* Is white space used to make logical expressions, array references, and routine arguments readable?
* Do incomplete statements end the line in a way that’s obviously incorrect?
* Are continuation lines indented the standard indentation amount?

Functions/Routines/Methods

* Does each function perform one and only one well defined task?
* Does the name descrive exactly what the rountine does?
* Are the arguments to each routine formatted so that each argument is easy to read, modify, and comment?
* Are blank lines used to separate parts of a routine?

Control Structures

❑Does the code avoid doubly indented begin-end or {} pairs?

❑Are sequential blocks separated from each other with blank lines?

❑Are complicated expressions formatted for readability?

❑Are single-statement blocks formatted consistently?

❑Are case statements formatted in a way that’s consistent with the format- ting of other control structures?

❑Have gotos been formatted in a way that makes their use obvious?

Commenting Checklist

Comments

* Do the comments clarify the intent of the code rather than repeat or try to explain code?
* Do the comments summarise the code and help the reader?
* Are the comments indented the same number of spaces as the code they comment?
* Is the commenting style easy to maintain?
* Is there any disagreement between the code and the comments?
* Any redundent development markers?
* Any out of place single line comments?
* Any unexplained abbreaviations?
* Any parts that are uncommented and need to be?
* Have you been commenting as you write the code?

Variable comments

* Units and allowable ranges in comments?
* Comment limits on input data
* Comment why global variables global
* Comment any relationships between variables if they exist.

Routines

* Any third party routines documented with source?
* All non-trivial rountines commented?

Special Cases

* Document compiler work arounds
* Document any style violations

Classes, Files and Programs

❑Is there a one-to-one relationship between classes and files for most classes and files?

❑If a file does contain multiple classes, are all the routines in each class grouped together and is each class clearly identified?

❑Are routines within a file clearly separated with blank lines?

❑In lieu of a stronger organizing principle, are all routines in alphabetical sequence?

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. http://www.apple.com/nz/ios/ios-10/ [↑](#footnote-ref-3)