Table of Contents

Analysis **2**
- Key Terminology 2
- Project Background 3
- Description of Current System 3
- Problem Identification 5
- Discussion of Research Techniques and Results 6
- Identification of Client 7
- Identification of Prospective End Users 7
- Key Equations 8
- Data Sources and Destinations 8
- Analysis Data Dictionary 9
- Outline of Proposed Solution 9
- Final Objectives 10
- Acceptable Limitations 11
- Discussion of Hardware and Software 12
- Entity-Relationship Model 13
- Log of Research Collection 13
- Use Cases 13

Design **15**
- High Level System Overview 15
- Inputs, Outputs, Processes, and Storage 16
- System Flowcharts 17
- Hierarchy Charts 18
- Key Formula and Equations 19
- Algorithm Design 19
- File Structure and Organization 20
- Data Dictionary 21
- Object Diagrams 22
- Data Structures 23
- Explanation of Library Software 23
- Design of user interface 23
- Naming Conventions 25
- Hardware Selection 25
- Security and Integrity of Data 26

Testing **26**
- Completeness of Solution: Objective Testing 26
- Testing for Complex Algorithm 1 30
- Testing for Complex Algorithm 2 31
- Testing for Complex Algorithm 3 31
- Testing for Complex Algorithm 4 32
- Testing Screenshots 32

Implementation **40**
- Completeness of Solution 40
- Evidence of Coding Styles 42
- Techniques Used – Data Models 47

Techniques Used – Algorithms **52**

Evaluation **55**

Objective Analysis **55**

User Feedback Interview **62**

Analysis of User Feedback **64**

Possible Extensions **64**

Appendix **66**

Appendix 1 – Client's Current System **66**

Appendix 2 – Client Interview **66**

Appendix 3 – Initial Student Survey **67**

Appendix 4 – Task List Page (TaskList.swift) **69**

Appendix 5 – Daily Timetable Page (Timetable.swift) **75**

Appendix 6 – Task Class (Task.swift) **81**

Appendix 7 – Application Delegate Class (AppDelegate.swift) **83**

Appendix 8 – Revision Task Class (RevisionTask.swift) **85**

Appendix 9 – Edit Task Page (EditTaskViewController.swift) **87**

Appendix 10 – View Task Page (DetailsViewController.swift) **89**

Appendix 11 – New Task Page (AddTaskViewController.swift) **90**

Appendix 12 – Profile Page (ProfilePage.swift) **93**

Appendix 13 – Edit Profile Page (EditProfileViewController.swift) **95**

Appendix 14 – User Class (User.swift) **97**

Appendix 15 – Task Priority Queue Class (RevisionPriorityQueue.swift) **99**

Analysis

Key Terminology

- Google Calendar – an online calendar which my client's school uses to schedule tasks and appointments
- Google Classroom – an online forum where teachers of specific classes can post homework for their students. Any updates on Google Classroom get synced to Google Calendar
- Revision timetable – a daily schedule containing a list of subjects and topics to revise, for a set duration, in order to prepare for exams
- Task – a piece of work defined by the user which they will complete during a set period of time
- Tanglin Trust School – a British-international school in Singapore where my client is based
- CAS – a compulsory part of the Tanglin curriculum which all Sixth Formers must participate in every week. During this lesson, students often help support the local community through volunteering work and other projects.
- Co-Curricular Activity (CCA) – an activity done by my client which is outside of his school syllabus but needs to be allocated time for *e.g. weekly football training*
- AS Exam – an exam some students sit at the end of Year 12 which will form a critical part of their final A-Level grade
- Relative priority – a ranking which can be used to find the importance of one task relative to another
- iOS – an operating system used for mobile devices manufactured by Apple

Project Background

Revision timetables are a key part of exam preparation for many students. These timetables are typically quite complicated to make as students need to account for many factors. For example, students will need to plan their revision timetable around not only their homework timetable for school but also their CCA timetable. This is a time-consuming process, usually requiring multiple days at the start of their already intense exam season.

Timetables are often made in the lead up to the exam season so that students are adequately prepared for them. From the surveys I have conducted on the matter (as can be found in Appendix 3), a Year 12 student will spend on average 4.5 hours a day revising for AS Exams – this shows that it is a stressful time for students, especially when you consider the fact that exam seasons often last for months at a time. Timetables are therefore necessary to structure this time effectively. Out of my cohort of 200 students, my surveys show that a considerable number of students use revision timetables to help them revise. As such, a system to help them create these would be greatly beneficial to the students within Tanglin Trust School.

Description of Current System

My client currently uses an informal system on paper where he writes down all his tasks at the start of the day and organizes them using the system above. Images of this can be found in Appendix 1. Using this system, he is able to roughly organize his day and sort out what tasks he will do in what order. I will go into more detail about how my client generates his timetables below.

There are a number of different methods students use to generate revision timetables. For the purposes of this project, I will be focusing on the system my client uses to create his timetables. As part of his process, he first assigns each topic he wants to cover a relative priority. From here, he sets aside a number of hours during which he can go through his revision list each day. Finally, he divides the time he has for revision by the number of tasks he has to cover in order to know how much time to devote to each topic. This lets him go through each task in order of ascending priority for the calculated period of time, leaving roughly five minutes between each task as a break. Any tasks he doesn't manage to finish in the set period of time will be carried over to the next day. These tasks cannot be carried out sequentially, however, as my client has other activities such as CAS which need to take place at a fixed time for a fixed length. Therefore, he plans in advance where he can slot each task in by seeing where he has adequate time. In an interview with my client, he states that he creates his 'timetable on a day-to-day basis' each morning. The full interview can be found in Appendix 2.

To give a worked example of this, I have carried out the algorithm above using a sample list of tasks below. I have also assigned a relative priority for each of them, with 1 being the most urgent task. This can be seen in Figure 1 below.

Task	Relative Priority
Write analysis	1
Revise databases	4
Revise mechanics	3
Revise integration	2

Figure 1 - An example task list with relative priorities

This is the start of my client's process. From here, he will look through his daily timetable and see how much time he can devote to going through his tasks. This timetable includes all the fixed tasks he must do at a certain time. Normally, this is done very informally, either on a sheet of scrap paper or mentally. However, to illustrate how the algorithm works, I will plan out his afternoon using a spreadsheet as seen below in Figure 2.

Time	Schedule
16:00 - 16:30	CAS
16:30 - 17:00	CAS
17:00 - 17:30	-
17:30 - 18:00	-
18:00 - 18:30	-
18:30 - 19:00	Walk Dog
19:00 - 19:30	-
19:30 - 20:00	-
20:00 - 20:30	-
20:30 - 21:00	-
21:00 - 21:30	Gym
21:30 - 22:00	Gym
Total Free Time	***3.5 Hours***

Figure 2 - An example of a typical afternoon for my client

As you can see from the schedule above, my client might have 3.5 free hours during which he can complete his tasks on a typical day. This will let him determine how much time he should spend on each task as he can divide the total free time he has by the number of tasks he has. For the above example, this means he should spend 52.5 minutes per task. This time includes a five-minute break which my client has at the end of each task, so in actual fact he is spending 47.5 minutes per task along with a five-minute break afterwards. From here, he will simply slot in all his tasks into his timetable in priority order, from highest to lowest priority. This is so that he can ensure he gets his most important work done first. My client likes to make sure that he spends an appropriate amount of time on each task. To this end, he tries to spend no more than 2 hours on a single task, and no less than 30 minutes.

Finally, at the end of his day he will go through all his tasks and see whether he has not managed to complete any of them. If so, he will add these tasks to the next day's task list. These tasks are given the highest relative priority so that he can finish them first on the next day.

Figure 3 below shows the data flow for the current system.

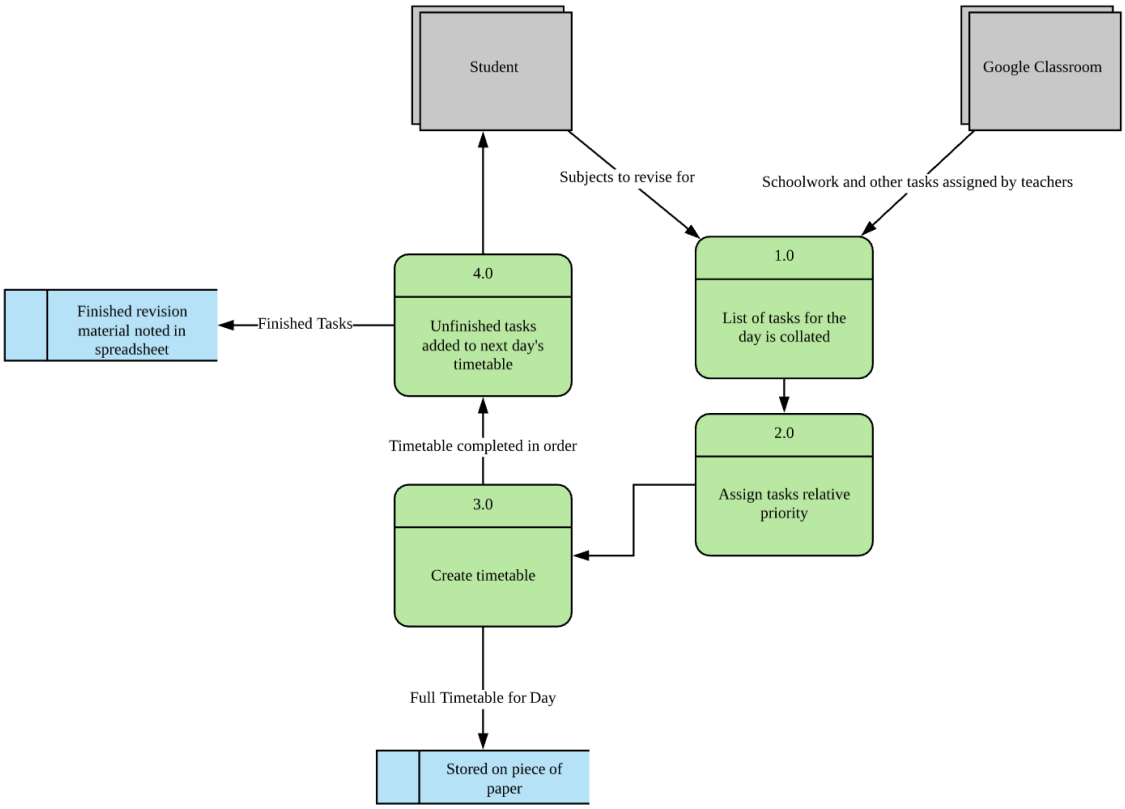


Figure 3 - DFD for Current System

Problem Identification

The current system above has helped my client with his revision so far. However, he has also faced a few issues with this current system which is what I will be hoping to fix.

The main issue with this system is that my client often forgets to add prior commitments to his schedule, resulting in him miscalculating how much free time he has. This mostly happens as my client writes his timetable and homework on scrap pieces of paper; however, school work gets assigned on Google Classroom. Since these two systems aren't always in sync, this results in him wasting time as he has to recalculate his revision timetable after realizing what tasks he is missing. This is a frustrating and inefficient process as he should be able to get on with his work rather than planning out what he needs to do.

Another issue my client faces is that he is often not aware when he should switch tasks. Since he writes his tasks on a piece of paper, he sometimes forgets how long he should spend on each task as he is in the middle of doing work. This results in him missing out certain tasks as he spends too long on the earlier tasks, allowing him to cover less revision material over the course of the day. This makes his revision less effective for a number of reasons. Firstly, he has to spend less time on later tasks as they get pushed back and he has to get them done. Secondly, he often has to keep checking his watch when revising to make sure he hasn't overshot his allotted time which makes the time he does have for revision

less effective. This creates a lot of stress for my client and makes his time spent less productive.

Finally, my client can't keep track of prior tasks since he writes it all on scrap paper. For example, he may not remember whether he has already covered a certain topic as he may have lost his timetable from that day. This makes his time less efficient as he is redoing work that he has already finished.

Discussion of Research Techniques and Results

Interviews

One of the first things I did was interview my client about his specific needs and issues. This can be found in Appendix 2. This helped me understand what problems he faces on a day to day basis and what he really wants from this product. From these interviews, I found out that one of the main issues my client faces is that he doesn't know when to switch tasks as he is too focused on his work – this would not have occurred to me had I not asked for his personal experiences with revision timetables. He also told me that he would like the timetable to sync up with Google Calendar as his school uses it heavily. I plan on conducting more interviews once I start building my product with him to make sure it suits his needs.

Questionnaires

I have sent out a survey to a small sample of students, which should be representative of my cohort, asking them about their scheduling habits. This is so that I can gain a sense of what other students besides my client do to plan their revision to see if there is a more efficient way to approach the problem. This is also so that I can understand how big the problem is within my school and who my potential end users could be. The results of this survey can be found in Appendix 3. The survey revealed a number of things about students at my school. Firstly, it showed that the majority of students at my school used revision timetables to help plan out their work which means they could be viable end users. It also revealed how pressing the problem was as students spent, on average, 4.5 hours a day revising and most have at least 7 exams to revise for which shows they need a method to structure their time. Finally, from my sample I could see that most of the students at my school had at least 1 weekly commitment every week and some found these hard to balance around their work schedule. This is all useful information for me to plan and build my product. I plan on doing more of these surveys once I start building my application in order to gain valuable user insights into it.

Online Research

My client based his algorithm for generating revision timetables off of research he had done online using various different websites and educational boards. Therefore, to understand how to create effective timetables better, I not only used the information my client told me but also conducted some of my own research. I plan on incorporating these details into my app to make the generated timetables easier to use and more effective. For example, one of the websites told me to implement frequent breaks in timetables to avoid students getting burnt out – I plan on adding this to my app to ensure that the time my client spends revising is as effective as possible. I've also done some research into a similar application which the Scottish Qualifications Authority uses to help students study by managing their revision; in

their application, they keep track of what a student has already covered in order to motivate them and keep them going which I plan to do as well.

Personal Experience
As someone who studies very similar subjects to my client and who also has a lot of commitments outside of school, I have experience myself with how difficult revision timetables can be to create and how quickly students can digress from them. I think with this proposed system I could have made my study time more effective and been able to manage my workload far more effectively.

Identification of Client

My current client is Oliver Ogden, a Year 12 A-Level student at my school. Being involved in many different activities both in school and outside of it, he needs to plan his days around his prior commitments and work. He also has multiple AS exams which he will be sitting at the End of Year 12, so is in need of a system to better help plan his schedule for him. He uses an iPhone 7 currently and is hoping to have an application on his phone so that he can view his timetable at any time.

Identification of Prospective End Users

I have two ideal end users, one of which uses the app directly while the other influences it indirectly. My first prospective end user is a student in Sixth Form who has a lot of commitments both within and outside of school. As they are nearing their final years in high school, they have a lot of exams but are unsure how to plan their days accordingly. This student also attends a school that uses Google Classroom heavily to aid learning. This user uses the app directly on their phone.

My second end user is a teacher who uses Google Classroom heavily. They would have a lot of students in Sixth Form and would use Google Classroom to assign them homework and required reading. This end user does not use the app directly, but they influence it through posting tasks on Google Classroom.

In terms of what both users should be able to accomplish with the app, I have listed out their user needs below. My client's user needs will also be the same as the student's user needs.

Students:

- Should be able to view all of their tasks and subjects which they've added for the current week
- The new system should take out all of the manual work involved in making timetables, so that certain tasks which the user specifies should be automatically placed into a revision timetable which is generated
- Students should be able to change details about a task at any time and the timetable should adapt around these changes
- Students should be able to delete tasks
- Students should be able to access tasks on an Apple mobile device

Teachers:

- Should be able to add tasks in Google Classroom which automatically gets synced to each user's local timetable

Key Equations

The calculations for my project are not very complex. Therefore, there are not a lot of key equations associated with it. The majority of my project will be focused on my client's algorithm for generating revision timetables which can be found in the 'Description of Current System' section. This process does not have any key equations or formula as it is mostly organizing a list of tasks in priority order. As such, my main algorithms used will be with creating queue objects and ensuring that the most important tasks get popped first from this queue.

Data Sources and Destinations

Inputs

- Users will be able to enter in a task by pressing a button. Entering in a task involves entering in the following pieces of information into a task input page:
    - Task name, which can be entered using a text field
    - A due by date using a date and time picker
    - Any special notes for the task, which will be entered using a text field
    - The importance of the task, which will be entered using a drop down list asking the user whether the task is very important, important, or not important
    - Whether the task should be included in an automatically generated revision timetable for the day, which will be selected using a checkbox
- Users can edit any of the above fields of a task by clicking on a task, which will take them to the task input page for that specific task
- A task can be marked complete by pressing the task
- Users can sync their Google Calendars with the app by pressing a button
    - They will enter in their username and password for Google using the Google API input fields

Outputs

- The timetable for the current day will be displayed on a table which can be found on the home page
    - Tasks will be color-coded by importance (red is most important, orange is important, and green is not important)
    - The default task colour will be green
    - Automatically scheduled revision tasks will be displayed in grey
- The automatically generated revision timetable will be displayed in this same table
- The user will receive a notification on his phone when a task or subject on his revision timetable has started
- The users profile and how many tasks they have completed can be found on their profile
- When buttons are clicked sound effects should be heard

Analysis Data Dictionary

Field Name	Data Type	Length	Description
Task Name	String	50	Stores the name of the task e.g Revise Differentiation
Task Notes	String	255	Any additional information needed about the task e.g remember to do end of chapter questions
Priority	Integer	10	Stores the relative position of how important a task is e.g 5 (5th most important task)
Time Period	String	50	Time frame during which the task will happen e.g 12:50 – 1:50
Time Per Task	String	20	How many minutes should be spent on each task e.g 45 minutes
Number of Tasks Done	Integer	15	The number of tasks which have been finished in one day e.g 5

Outline of Proposed Solution

In order to solve my client's issue, I plan on making an iPhone application using Swift and Objective-C which will generate his revision timetable and manage his day for him. My application will also use Google Calendar to sync my client's timetable with his schools. This will give my client one central location where he can find his work for the day, solving one of his main problems that his school work and personal revision are stored on two separate mediums.

This application will allow a user to create a new task and enter in certain details (which can be found in the Data Sources and Destinations section). From here, the application will slot it into his planner for the day and, if indicated by the user, will automatically add it to a dynamic revision timetable. This will be done using a queue. This solves another one of his issues that making timetables is a time-consuming process. The application will also automatically keep track of which tasks have been completed at the end of the day and which have not in order to add them to the next day's timetable.

The application will automatically notify my user when they should switch tasks via a notification on their phone, solving my client's final issue that he cannot keep track of when to switch tasks. This system should therefore streamline my client's revision process and make it much easier for him to balance all of his various commitments. Figure 3 shows how data flows in my proposed system.

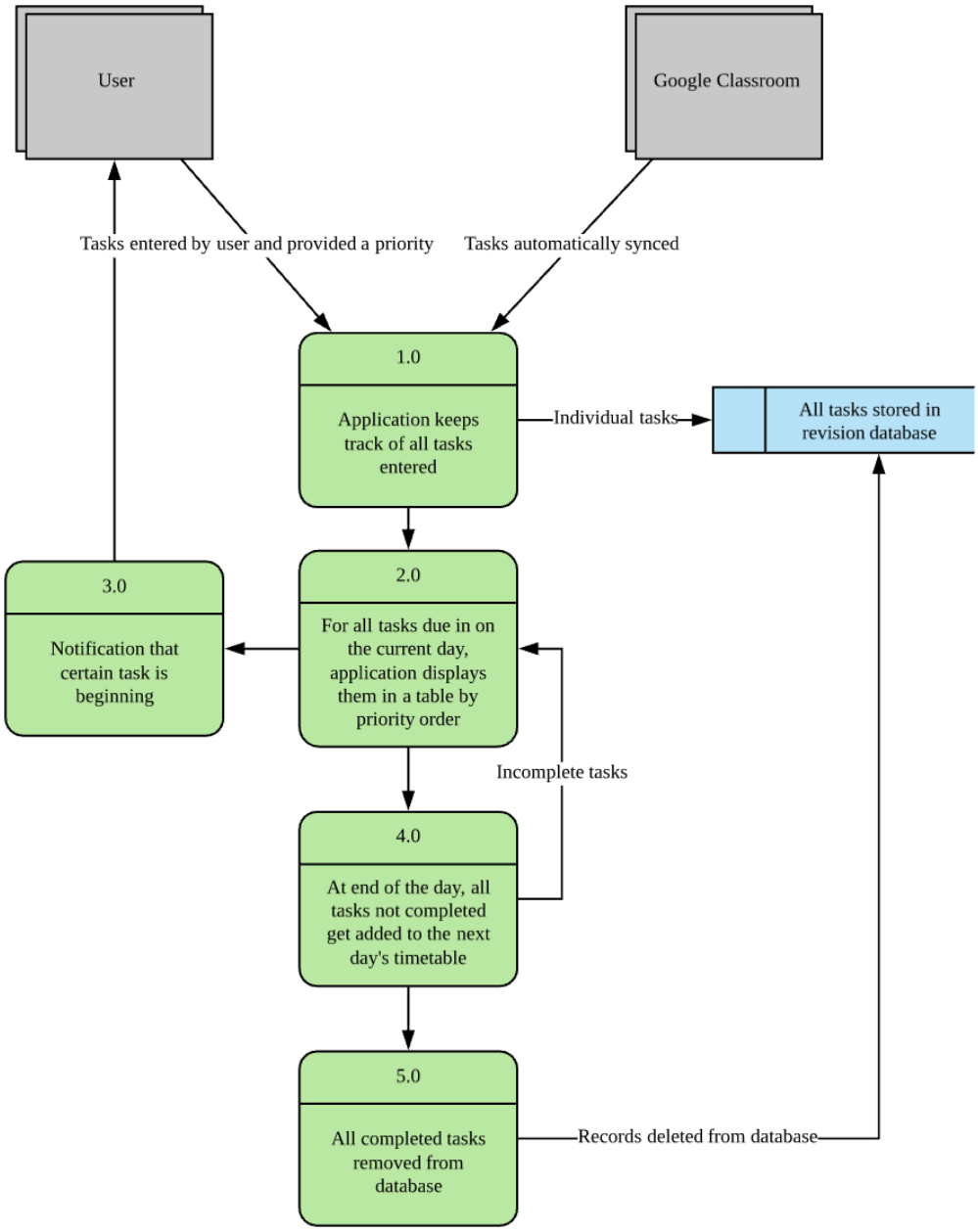


Figure 4 - DFD for proposed system

Final Objectives

1. The database must store all tasks which the user enters into the application
2. The database must be able to record key information about tasks which is stated in the Data Sources section
3. The proposed solution must be accessible by iPhone 8's

4. The system must be able to connect with Google Calendar
5. The system must sync the user's local timetable with that on their Google Calendar account
6. The system should play a 'click' noise when a button is pressed
7. Tasks should be able to be entered using a form
8. Tasks should be colour-coded in the timetable depending on priority
9. The system should be able to handle a high number of tasks and still be functional
10. The system should automatically adjust the revision timetable when new tasks are inputted
11. The application should automatically notify the user with a notification on their phone when a task has begun
12. The system should notify the user when two tasks are inputted which occur at the same time
13. The user should be able to view all tasks which have been entered into the system
14. The user should be able to view the full timetable for their current day
15. Students should be able to choose whether they want a task to be part of the automatically-generated revision timetable or not
16. The system should show how many tasks the user has completed on a separate profile page
17. The user should be able to edit any attribute of a task by pressing on the task object which will bring up the task edit screen
18. The user should be able to easily mark a task as completed by pressing a button on a task box
19. The system should send notifications to the user even when the application is not open on their device
20. The system should automatically set aside time between different revision subjects as a break

These objectives may be revised following further interviews with my client.

Acceptable Limitations

1. The system will only be expected to work on iPhone 8 devices; it will not be expected to work on other Apple devices, Android devices, or as a webpage
2. The system will only generate revision timetables using my client's algorithm for creating them; users will not be able to adapt the algorithm of my system in any way
3. The tasks saved to a database will only be saved to a specific device; users will not be able to access their inputted tasks on another device
4. There will be no login system as users will use the system on their own devices
5. The user will not be able to edit the color-coding system or any part of the UI of the system
6. The system will only send 1 notification per task; users cannot change the notification frequency of the system
7. Users will have no direct access to the database and should instead edit it via a form
8. Users will only be able to view automatically-generated revision timetables for the current day; the system will not be expected to produce timetables for any day other than the present
9. All access rights should be the same for this system

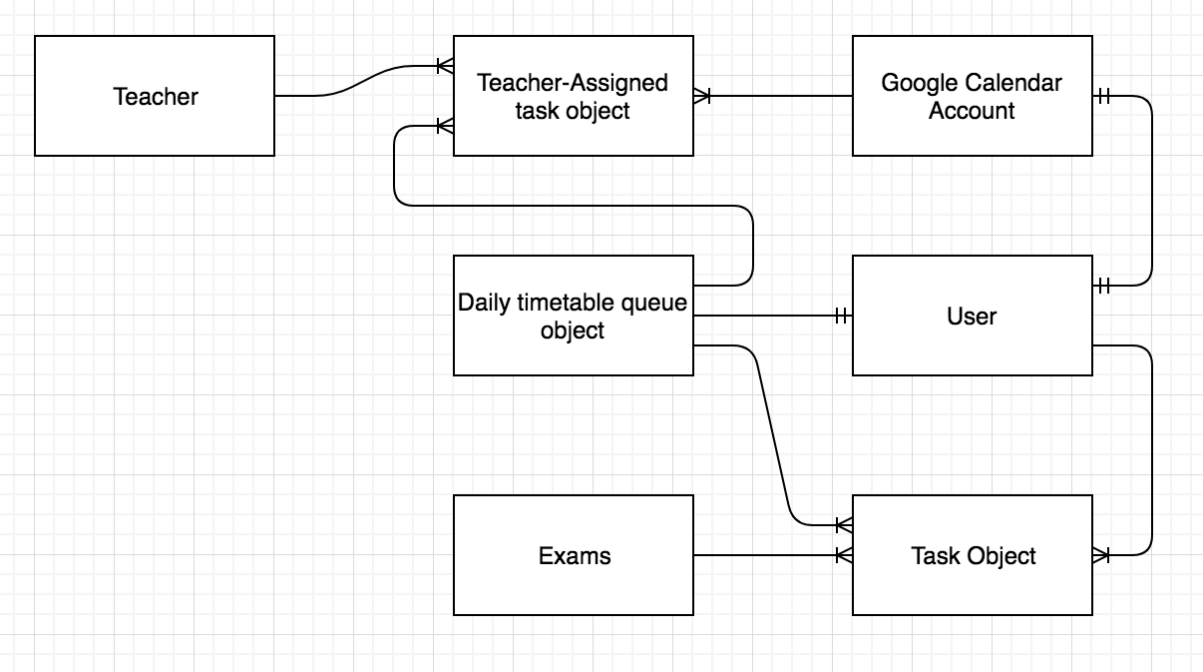
10. The system is not expected to sync with calendar applications other than Google Classroom
11. The system will not be expected to create timetables from scratch by itself – it will only arrange a timetable for the subjects the user enters into the system
12. The system will not be available on the Apple App Store, due to the costs of uploading an app

Discussion of Hardware and Software

Since my proposed system will be running as an iPhone application, I will be using an iPhone to test and deploy my solution. This piece of hardware is appropriate for my project as my client has an iPhone 7 so it will be suited to his needs. For testing purposes, I can connect my iPhone to my MacBook directly to run the app locally. Other than this, I will not be using any other hardware.

In terms of software, since I am creating an iOS application I will need to use Swift, a programming language created by Apple for app development. I will also need to use Objective-C as this is also necessary for certain parts of app development. I will need to use XCode, an Integrated Development Environment (IDE) that is shipped with MacOS, to program my application as it contains resources which are necessary for iOS programming. All of these resources have a lot of documentation associated with them which means I should be well-supported when I start programming my app. I will also be using the Google Calendar API to sync my user's tasks from Google Classroom to the app so my language choice is appropriate here as well as Google supports the iOS development environment.

Entity-Relationship Model



- One teacher can assign many tasks to a user (e.g homework) but one of these tasks can only be associated with one teacher
- One Google Calendar account can have many of these objects on it but one object can only appear on one specific account as it is only associated with that one user
- One instance of a task object (used to keep track of a specific task in a system) can only have one exam associated with it but one exam can be split into multiple tasks (e.g the task objects could be Exam Revision Module 1 and Exam Revision Module 2)
- Each Daily Timetable can have multiple task objects and teacher-assigned task objects but since only one daily timetable exists at a time they can only be associated with one of them
- All information about task objects and timetable objects will be saved to a database

Log of Research Collection

- Online Research – conducted from May 15th onwards
- Preliminary Client Interview (Appendix 2) – completed Thursday May 24th
- Photo of client system (Appendix 1) – received May 30th
- Initial Student Survey (Appendix 3) – last response in by July 15th

Use Cases

Title: Enter Task
Actor: Student User
Scenario:

1) Student enters in appropriate information for the task entry form (contains the following fields: title, due date and time, special notes, priority, whether to include in automatic revision timetable)
2) User alerted if any issues with data
3) New task appears in task list

Precondition(s):

- User has pressed the 'New Task' button

Title: Sync Google Calendar
Actor: Student User
Scenario:

1) Student presses 'Sync Google' button on profile page
2) Google login page appears where user can enter in details
3) Tasks on Google Calendar get added to task list

Precondition(s):

- User is on profile page

Title: Edit Task
Actor: Student User
Scenario:

1) Student clicks on a task in the task list
2) An edit form appears which contains all the inputted data for that task
3) The user can edit data fields within the form
4) The user presses a save button to save this updated task

Precondition(s):

- User is on task list page
- User has already entered tasks into the application

Title: Mark Task As Completed
Actor: Student User
Scenario:

1) User holds down a task in the timetable or task list
2) The task disappears with an animation
3) The total number of tasks counter is incremented by page on the profile page

Title: Add Google Classroom Task
Actor: Teacher
Scenario:

1) Teacher enters a task on Google Classroom which they assign to the student
2) Google automatically merges this with the student's calendar
3) Once the student signs in to Google, this task will be merged with their timetable

Precondition(s):

- The task the teacher assigns is a time-sensitive task (i.e. has a due date)

Title: Notify User
Actor: Application Software
Scenario:

1) When student is working, the application should send a notification to the user's phone informing them that the task is over
2) The notification will have a noise attached to it
3) The user can dismiss the notification once read

Precondition(s):

- A task has elapsed

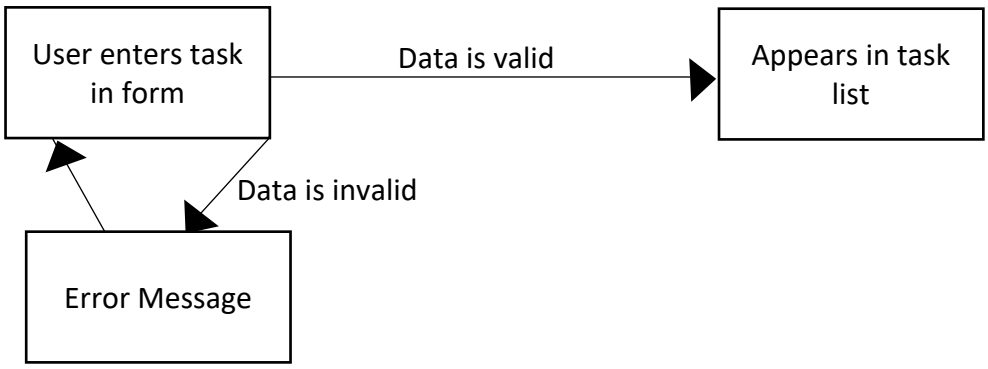
Design

High Level System Overview

In order to ensure that all aspects of my project are functional, I have split my work into four different stages. This way I can easily plan out my work schedule without getting overwhelmed by future functionality which I will have to implement.

Stage 1 – Basic Task List:

The first stage of my project will be in simply setting up my project in Swift and implementing a basic task list. My user will enter some attributes about a task in a form, this will be error checked, and then this task will simply appear in a list. In this stage of the process, I will not be filtering my tasks by date or time.



Stage 2 – Serialized Data:

The second stage of my project is storing my data by serializing it. Serialization is the process by which you store data permanently in IOS devices. In stage 1, whenever I would shut down the app all data in the task list would be lost. By serializing my data, I can store and reload all my data from each session. In this stage I plan to store all the data I will need for my project – this includes all tasks inputted and all data about the user (e.g. name, daily revision times etc.).

Stage 3 – Algorithm Implementation

In this stage, I will make my task list dynamic by implementing my client's algorithm described in the analysis section. I go into further detail on this algorithm later in this section. This algorithm will be implemented on my task list page and will filter tasks by date and time. Every time my user enters a new task or changes some data, this algorithm will be rerun in order to ensure my daily timetable reflects the latest changes.

Stage 4 – Google Calendar Compatibility
The final stage of my project will be linking it up with the Google Calendar API. This will allow my project to retrieve tasks from Google Calendar, merge them with my users existing task list, and slot them into his daily timetable. I have left this stage till the end as we can use the code from stage 1 to add a new task to the task list.

Inputs, Outputs, Processes, and Storage

Inputs:

- Task Attributes:
    - Name
    - Start time
    - End time
    - Priority
    - Task notes
- User attributes:
    - Name
    - Time to start revision
    - Time to end revision
    - Minimum time per revision task
    - Maximum time per revision task
    - Time between tasks as a break
- Task inputs:
    - Button to delete task
    - Button to mark task as complete
    - Button to edit task
    - Button to view task

Processes:

- Processes in order to automatically slot revision tasks into timetable:
    - Filter all tasks by the tasks which are taking place on the current day
    - Calculate how much free time the user has in a day
    - Sort revision tasks by priority order
    - Determine how many tasks can be fit into a gap in the day using the minimum and maximum time per task entered by the user
- Processes for user analytics
    - Determine how many tasks have been entered into system
    - Determine how many tasks have been marked complete
    - Determine how many tasks have not been complete yet
- Processes for entering a task:
    - Ensure data is correct type and format
    - Ensure all fields have been filled in
    - Use Google Calendar API to get relevant information from the user

Storage:

- Every time a valid task is entered, serialize all details of task and store on device
- Every time a change to user data or a task is made, store on device
- Retrieve all data from the last session every time the app starts

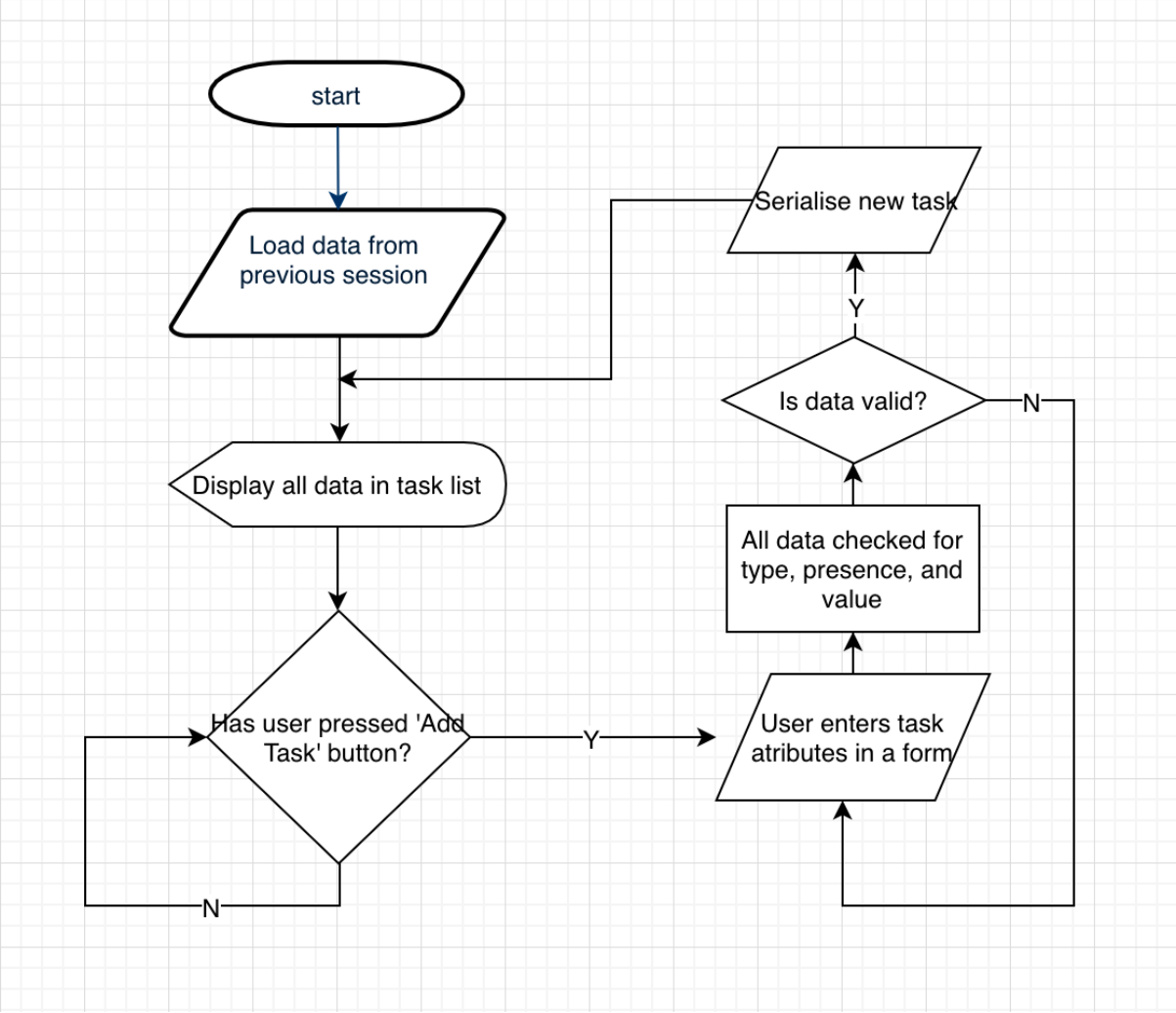
- Store all data on the current session right before app is shut down

Outputs:

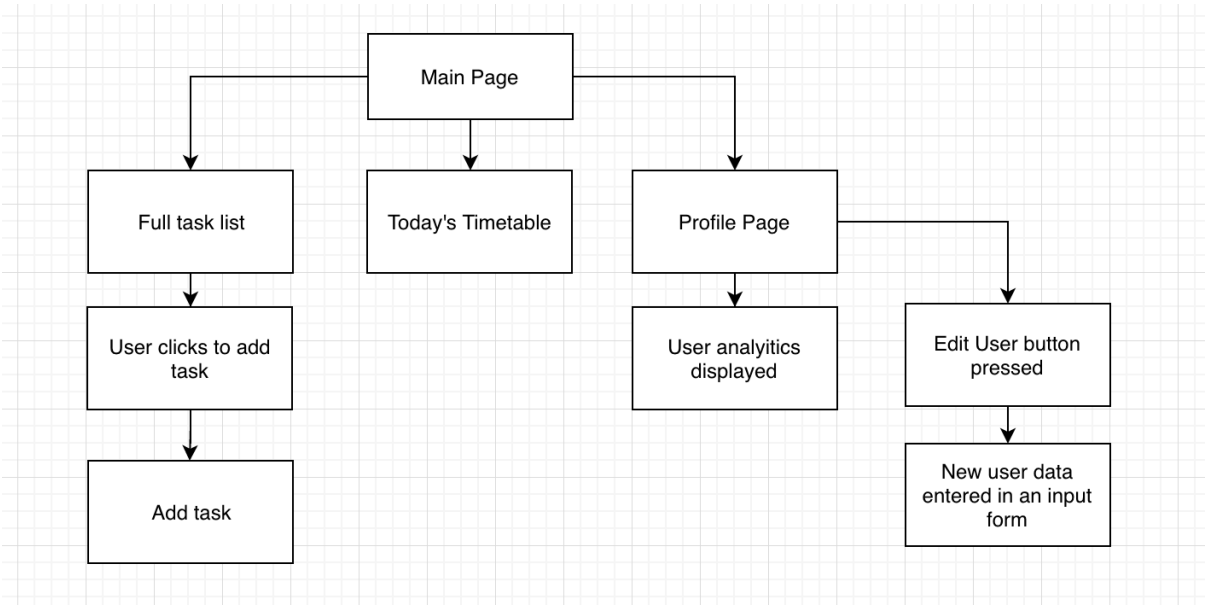
- Task attributes:
  - Name
  - Start date
  - End date
  - Priority
  - Notes
- Alert when task is starting

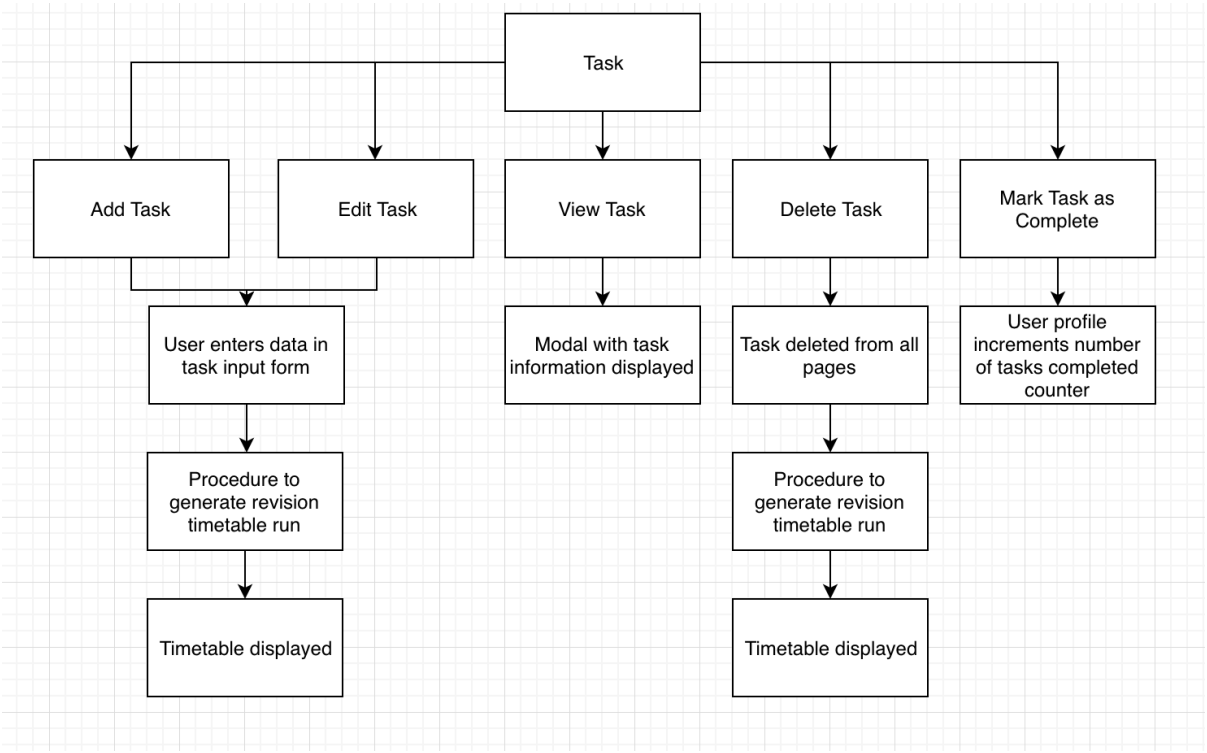
System Flowcharts

Add task flowchart:



Hierarchy Charts





Key Formula and Equations

Most of my project is based on my main algorithm which will slot revision tasks in between a user's daily timetable. In order for this to work, I will need to carry out a few calculations. The first calculation I will need to carry out is to find how much free time I have in a given day. I will do this by finding the maximum free time possible in a day, and subtract the sum of all the task durations for the current day. This can be written out as:

$$Free\ Time = Full\ Day - \sum(Current\ Task\ End\ Time - Current\ Task\ Start\ Time)$$

This is my main equation as the rest of my program uses this number to calculate how many tasks should be done in any given day. Other than this equation, most of the formulas in my program are simple increment operations. For example, I will need to increment a variable to find the total number of tasks entered into my system.

Algorithm Design

Complex Algorithm 1: Pseudocode for slotting tasks into daily timetable:

```
Timetable = Array of Tasks
//The timetable is stored as an array of Task Objects

User = New User()
//Initializes user object and their settings

RevisionTasks = New PriorityQueue() of Tasks
//RevisionTasks will be all the tasks the user wants to revise. It will be made using a priority queue storing the relative priorities of the tasks

freeTime = User.EndTime() - User.StartTime()
//User defined settings for when ending work and starting work to find total working time

For Each task in Timetable
        freeTime -= task.Duration()

TimePerTask = freeTime / RevisionTasks.Length()

finishedTasks = 0
For i = 1 to RevisionTasks.Length()
        For x = finishedTasks to Timetable.Length()
                finishedTasks += 1
                If Timetable(x).EndTime() > User.StartTime() AND TImetable(x).Duration() >= TimePerTask Then
                        Timetable.InsertAt(finishedTasks, RevisionTasks.Pop())
                        Timetable(finishedTasks).SetDuration(TimePerTask)
                End If
        Next x
Next i
```

TESTING FOR ALGORITHM 1 CAN BE FOUND ON PAGE 30

Complex Algorithm 2: Pseudocode for error checking new task added:

```
//Return TRUE if task is fine; return FALSE if task has error
FUNCTION CheckValid(newTask):
        FOR task in TasksArray
                //Checks if new task entered overlaps with another task already in the tasks array
```

```
                //If it overlaps then it is not valid
                IF newTask.StartDate() < task.EndDate() AND newTask.EndDate() > task.StartDate() THEN
                        RETURN FALSE
                ENDIF
        NEXT task
        //Checks that end date is after start date
        IF newTask.StartDate() < newTask.EndDate() THEN
                RETURN FALSE
        ENDIF
        RETURN TRUE
END FUNCTION
```

TESTING FOR ALGORITHM 2 CAN BE FOUND ON PAGE 31

Complex Algorithm 3: Pseudocode for only displaying tasks which are due on current day

```
Timetable = Array of Tasks
//The timetable is stored as an array of Task Objects

currentDate = Date.Today()
//Store the current date in a variable

todayTimetable = Empty Array
//Empty array which will hold today's timetable

FOR task in Timetable
        IF task.GetDueDate() = currentDate THEN
                todayTimetable.Push(task)
        ENDIF
NEXT task

DISPLAY todayTimetable
//Displays the todayTimetable array in a graphical format
```

TESTING FOR ALGORITHM 3 CAN BE FOUND ON PAGE 31

Complex Algorithm 4: Pseudocode for adding alerts on the user's phone

```
FUNCTION AddAlert(NewTask)
        newAlert = NewTask
        //Create new alert object
        newAlert.time = NewTask.GetStartTime()
        Device.AddAlert(newAlert)
END FUNCTION

FUNCTION AddTask(NewTask)
        …
        //Code for add task function

        AddAlert(NewTask)
END FUNCTION
```

TESTING FOR ALGORITHM 4 CAN BE FOUND ON PAGE 32

File Structure and Organization

In my project I will be serializing data to store it in the device. This will be stored as a text file under the 'User Defaults' memory location in IOS. Swift handles this encoding for me

automatically which means I simply define the objects I will need to store in my class initializer and call a function to encode the data when my app is about to shut down.

I will store two different pieces of data. The first piece is user data which includes attributes such as username. The second piece of data is on the tasks entered into the system and includes attributes such as task name. More information on the nature of this data can be found in the data dictionary below.

Data Dictionary

Tasks:

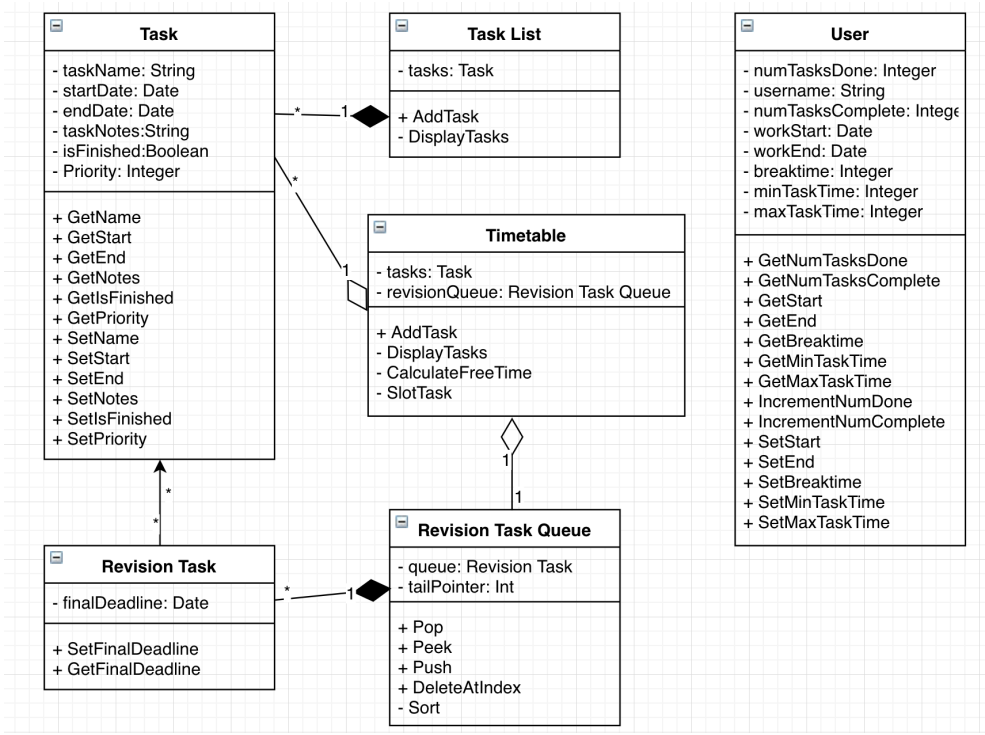
Field Name	Data Type	Length	Validation
Task Name	String	50	Type check
Task Notes	String	255	Check length and type
Priority	Integer	1	Type and range check (must be between 1 and 3)
startDate	Date	50	Type and format check (must be DD/mm/YYYY)
endDate	Date	50	Type and format check (must be DD/mm/YYYY). Must also be after startDate.
isRevisionTask	Boolean	1	Prescence check

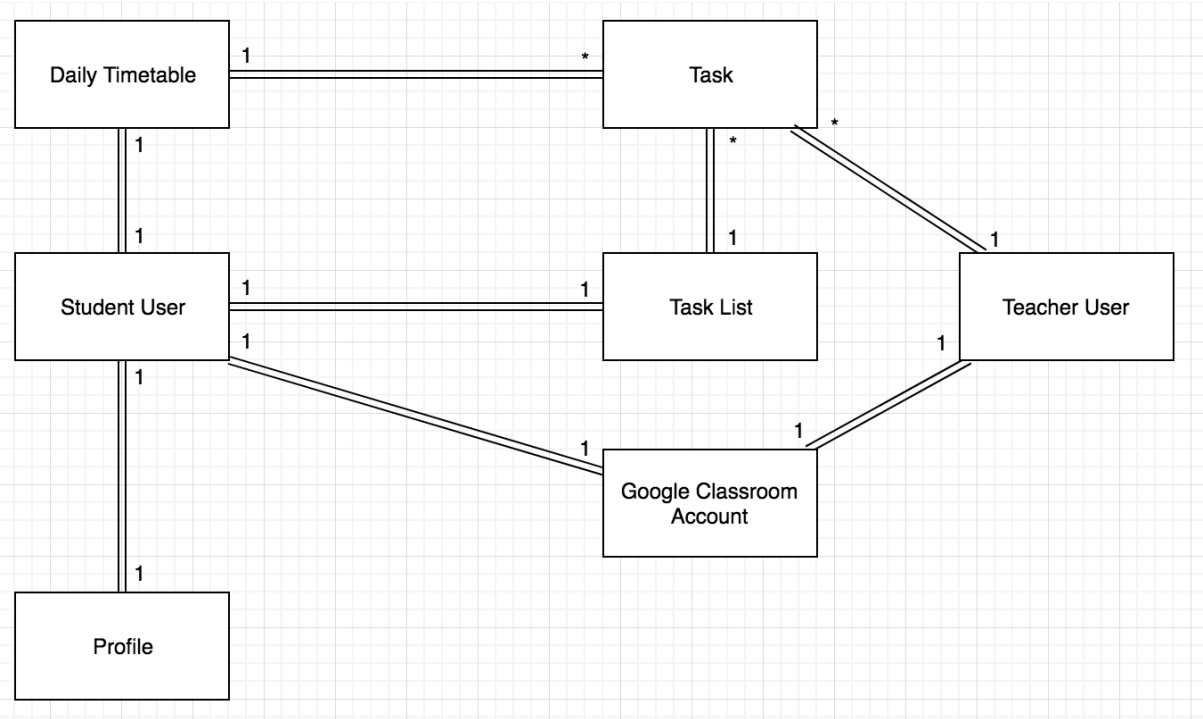
User:

Field Name	Data Type	Length	Validation
Username	String	50	Type and presence check
Number of tasks added	Integer	100	Range check
Number of tasks completed	Integer	100	Range check
Minimum task duration	Integer	100	Range and type check
Maximum task duration	Integer	100	Range and type check
Time between tasks	Integer	50	Type and range check
Work start time	Time	50	Type and format check (must be hh:mm)

Work end time	Time	50	Type and format check (must be hh:mm)

Object Diagrams





- One daily timetable is made up of a number of tasks
- Each task can only appear on one task list

- A student user only has one Google Classroom Account and one Daily Timetable
- A teacher user can assign multiple tasks
- One student only has one profile on the application

Data Structures

Arrays:

I have used arrays throughout my project. For example, in my task list class, I have a task list array which will store Task objects. This will be used to store all the tasks the user has entered into the system. As such, this array is the most important aspect of my program as it stores all the information the user has entered into the system. I also use an array in my Revision Priority Queue class to store the revision tasks as a priority queue.

Priority queue:

I have used a priority queue to store all my revision tasks. This is an appropriate data structure for this purpose as I want the most important revision tasks to be scheduled first, with the earlier revision tasks being popped first as well. To create this I have created my own revision priority queue class. This uses an array to store all the revision tasks. I also have a sort class to move the highest priority tasks to the front of the queue. In order to pop the next task, I simply retrieve the element in the first slot, decrement the tail pointer, and shuffle all the elements down by one.

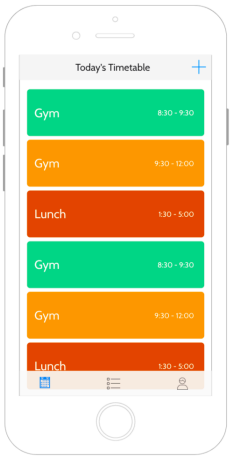
Explanation of Library Software

As stated by my client needs, I have used a IOS Google Calendar API in order to sync Calendar events to my app. This works by having the user sign into Google through the profile page. The app then automatically adds all Google Calendar events to the task list once the user is authenticated. I used the following website (https://blog.hipolabs.com/how-to-use-the-google-calendar-api-on-ios-563ad3abb006) as a reference to help me link up the API to my app.

Design of user interface

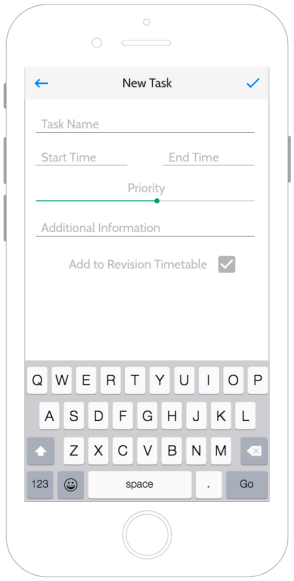
Timetable page:

This page shows all the tasks the user will have to do in chronological order for the current day. As you can see, it is colour coded to reflect the priorities of each task. I have a toolbar at the bottom which contains three buttons. The left button is the current page; the center button takes users to a page showing all tasks they have entered into the system; the rightmost button is the users profile page which shows various statistics about them. At the top right of the screen I have a + button which allows users to enter a new task into the application. This will take them to the input form which I will discuss next.



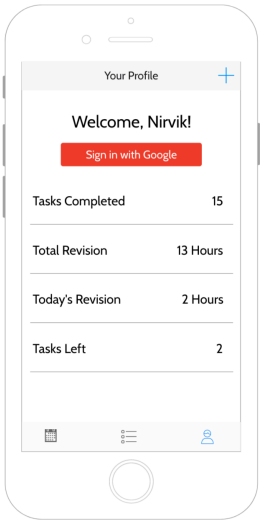
Input form:

This is a design of my task input form to allow my user to enter a new task into the system. The form is made up of a number of text inputs and checkboxes which the user can click to type in information. At the top right is a tick box which will allow the user to enter in a task. If any fields are missing or wrong, the user will be notified through a notification. At the top left is a back button which will cancel the current form.



Profile page:

This is a design of my user's profile page. This page will be used to firstly display analytics about my user. It will also have a button to connect to Google in order to sync their Google Calendar timetable with their app timetable. Like my first design, it features a toolbar at the bottom in order to navigate between the apps various pages. It also has a + button at the top right in order to add a new task. The various text fields in the middle of the page show analytics about my users work habits. Finally, the red button can be pressed in order to take my user to a Google login page. This will allow them to connect to Google.



Naming Conventions

Throughout my program, I will give my variables meaningful identifiers in order to make the code clear. Firstly, I will ensure that all variable names are appropriate for the data they are storing to minimize confusion. I will also use lowercase camel case to store multi-word variable names. For example, a variable for a task list array might be called taskListArray.

Hardware Selection

Since my client wants an easily accessible solution, my project will only be available for iPhone 8 devices made by Apple. This is as my client has an iPhone 8. All data will be stored locally on my user's device. When my user first opens the app, it will have some default values for the user attributes; the user will be able to easily change these details through the profile page.

Input device:

- Touch screen
- iPhone home button

Output devices:

- iPhone screen (1334 x 750 resolution at 326 ppi)
- iPhone speakers

Storage device:

- iPhone main memory

Security and Integrity of Data

Since data will be stored locally on a user's iPhone, many aspects of security are already dealt with for me as only the owner of the phone will be able to access the app. However, I have implemented a few measures to ensure data is not tampered with. First of all, to store my data I will be serializing it and storing it under the User Defaults section of iPhone memory. This memory slot is not accessible by users and is reserved for specific applications. Therefore, users will not be able to tamper with memory when the app is not open. The only way data can be edited is through the input forms I have designed and checked.

These input forms will allow me to normalize data. This is as I can split the task information down into its atomic components to ensure that all data is being stored in its most simple form. For example, rather than ask users to enter their task period I instead ask them to enter in the task start time and end time, allowing me to store these components separately.

I will ensure integrity of data through my various validation measures. Since data can only be entered into the app through two input forms, this is a relatively simple task. I will be checking input data using a number of rules such as range and presence checks. In order to ensure no data is lost, I will be regularly saving data to my user's device during the app's execution. Currently, my app is setup to write data to the user's device after a new task is added, edited, or about to be closed to ensure only the latest session information is saved.

Finally, since I will be using the Google Calendar API, I have a Google login screen to allow only a specific user to sync their calendar with my app's calendar. This secure login screen ensures that only valid users can sync their timetables, and since the user will need to log in every time this means that other individuals cannot view their calendars without logging in.

Testing

Completeness of Solution: Objective Testing

Tests for my complex algorithms are highlighted in grey and shown again further down.

Test No.	***Description of the test***	***Datatype***	***Example data***	***Expected Result***	***Pass / Fail***	***Cross Reference***	***Objective No.***
1	Test if task input form works	Normal	Name: Testing Start: 22/02/2019 End: 22/02/2019 Priority: Low	Task appears in task list	Pass	PAGE 32, Fig. 5	7, 13
2	Test if task input form error	Erroneous	Name: Testing Start: 22/02/2019	Error message displayed	Pass	PAGE 33, Fig. 6	7

	checking works.		End: 21/02/2019 Priority: Low				
3	Test if user notified when two tasks are entered which overlap.	Erroneous	Name: Testing Start: 21/02/2019 End: 23/02/2019 Priority: Low	Error message displayed	Pass	PAGE 33, Fig. 7	7, 12
4	Check iPhone is storing task information on local device.	Normal	N/A	Data dump outputted which shows a change to serialised data file.	Pass	PAGE 33, Fig. 8	1, 2, 3
5	Test tasks input form error checking works.	Erroneous	Name: N/A Start: 21/02/2019 End: 23/02/2019 Priority: Low	Error message displayed	Pass	PAGE 34, Fig. 9	7
6	Test Google Calendar functionality	Normal	N/A	Alert appears when calendars are synced. Tasks appear in task list as red task.	Pass	PAGE 34, Fig. 10	4, 5
7	Test task colour-coding functionality	Normal	Name: Low Priority Priority: Low	Task appears as green task in 'Task List' page	Pass	PAGE 34, Fig. 11	7, 8
8	Test task colour-coding functionality	Normal	Name: Medium Priority Priority: Medium	Task appears as orange task in 'Task List' page	Pass	PAGE 35, Fig. 12	7, 8
9	Test task colour-	Normal	Name: High Priority Priority: High	Task appears as red task in	Pass	PAGE 35, Fig. 13	7, 8

	coding functionality			'Task List' page			
10	Test user notifications	Normal	N/A	User notified when application is not running	Pass	PAGE 35, Fig. 14	11, 19
11	Check 'Daily Timetable' page only shows tasks due on the current day	Normal	Name: Tomorrow Task Start: 25/02/2019 End: 26/02/2019	Daily timetable page is not affected by tasks due on the following day	Pass	PAGE 36, Fig. 15	14
12	Test automatic revision timetable functionality	Normal	Name: Revision Task Is Revision Task: True	Task appears as grey box; task start and end date is automatically calculated in 'Daily Timetable'	Pass	PAGE 36, Fig. 16	15
13	Test functionality for marking task as complete	Normal	N/A	Task disappears from 'Task List'; completed tasks counter incremented on profile page	Pass	PAGE 36, Fig. 17	16, 18
14	Test edit task functionality	Normal	New Priority: Low	Task appears as green task	Pass	PAGE 37, Fig. 18	17
15	Test edit task functionality	Normal	New Name: Edit Task	Task name changed	Pass	PAGE 39, Fig. 24	17
16	Test edit task functionality	Erroneous	New End Date: 1/1/2000	Error message appears	Pass	PAGE 39, Fig. 25	17

17	Test edit task functionality	Erroneous	New Start Date: 25/02/2019 25:38	Error message appears	Pass	PAGE 37, Fig. 19	17
18	Test edit task functionality	Erroneous	New End Date: 25/02/2019 25:38	Error message appears	Pass	PAGE 37, Fig. 20	17
19	Test revision timetable priority calculations	Normal	Name: Important Task Priority: High	New task happens before other revision tasks as most important	Pass	PAGE 38, Fig. 21	10
20	Test revision timetable task slotting algorithm	Normal	Task Break Time: 2 [minutes]	Daily timetable updates revision tasks to leave 2 minutes between tasks	Pass	PAGE 38, Fig. 22	10, 20
21	Test revision timetable task slotting algorithm	Normal	Minimum Task Time: 60 [minutes]	Daily timetable updates to ensure that each revision task is at least 60 minutes long	Pass	PAGE 38, Fig. 23	10
22	Test button click	Normal	N/A	System should play a click every time a button is pressed	Pass	N/A	6
23	Testing functionality when large number of tasks entered	Extreme	10 revision tasks entered into system	System should behave normally and slot tasks in	Pass	PAGE 39, Fig. 26	9

				daily timetable			
24	Testing app can handle a large maximum task time	Extreme	Max Task Time: 100000	System should behave normally and slot tasks in daily timetable using normal algorithm	Pass	PAGE 40, Fig. 27	-

Testing for Complex Algorithm 1

Test No.	***Description of the test***	***Datatype***	***Example data***	***Expected Result***	***Pass / Fail***	***Cross Reference***	***Objective No.***
19	Test revision timetable priority calculations	Normal	Name: Important Task Priority: High	New task happens before other revision tasks as most important	Pass	PAGE 38, Fig. 21	10
20	Test revision timetable task slotting algorithm	Normal	Task Break Time: 2 [minutes]	Daily timetable updates revision tasks to leave 2 minutes between tasks	Pass	PAGE 38, Fig. 22	10, 20
21	Test revision timetable task slotting algorithm	Normal	Minimum Task Time: 60 [minutes]	Daily timetable updates to ensure that each revision task is at least 60 minutes long	Pass	PAGE 38, Fig. 23	10

Testing for Complex Algorithm 2

Test No.	***Description of the test***	***Datatype***	***Example data***	***Expected Result***	***Pass / Fail***	***Cross Reference***	***Objective No.***
1	Test if task input form works	Normal	Name: Testing Start: 22/02/2019 End: 22/02/2019 Priority: Low	Task appears in task list	Pass	PAGE 32, Fig. 5	7, 13
2	Test if task input form error checking works.	Erroneous	Name: Testing Start: 22/02/2019 End: 21/02/2019 Priority: Low	Error message displayed	Pass	PAGE 33, Fig. 6	7
3	Test if user notified when two tasks are entered which overlap.	Erroneous	Name: Testing Start: 21/02/2019 End: 23/02/2019 Priority: Low	Error message displayed	Pass	PAGE 33, Fig. 7	7, 12
5	Test tasks input form error checking works.	Erroneous	Name: N/A Start: 21/02/2019 End: 23/02/2019 Priority: Low	Error message displayed	Pass	PAGE 34, Fig. 9	7

Testing for Complex Algorithm 3

Test No.	***Description of the test***	***Datatype***	***Example data***	***Expected Result***	***Pass / Fail***	***Cross Reference***	***Objective No.***
11	Check 'Daily Timetable' page only shows tasks due on the current day	Normal	Name: Tomorrow Task Start: 25/02/2019 End: 26/02/2019	Daily timetable page is not affected by tasks due on the following day	Pass	PAGE 36, Fig. 15	14

1	Check 'Daily Timetable' page shows all tasks which are due on current day	Normal	Name: Testing Start: 22/02/2019 End: 22/02/2019 Priority: Low	Task appears in task list	Pass	PAGE 32, Fig. 5	7, 13

Testing for Complex Algorithm 4

Test No.	***Description of the test***	***Datatype***	***Example data***	***Expected Result***	***Pass / Fail***	***Cross Reference***	***Objective No.***
10	Test user notifications	Normal	N/A	User notified when application is not running	Pass	PAGE 35, Fig. 14	11, 19

Testing Screenshots

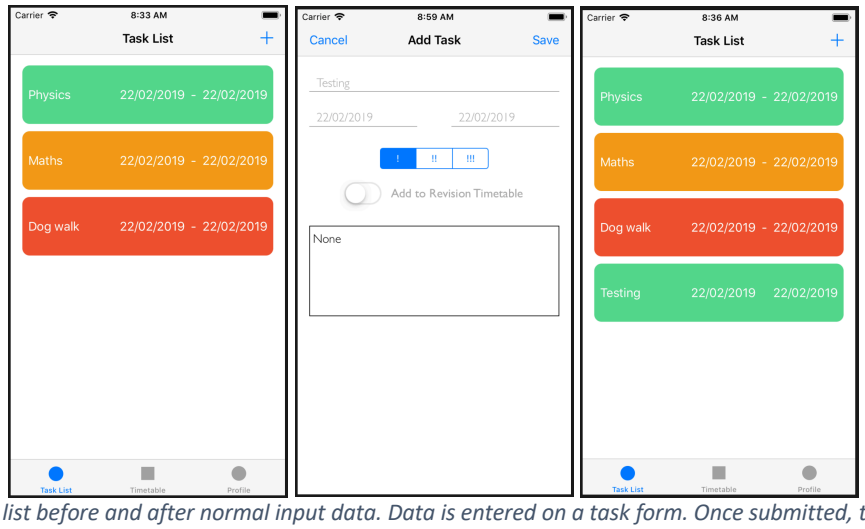


Figure 5 - Task list before and after normal input data. Data is entered on a task form. Once submitted, task appears as coloured box. All tasks are displayed on this page.

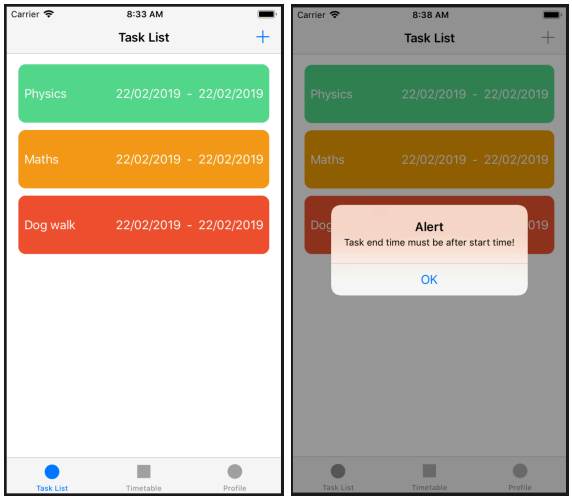


Figure 6 - Task list before and after incorrect task end time entered. Error message shown.

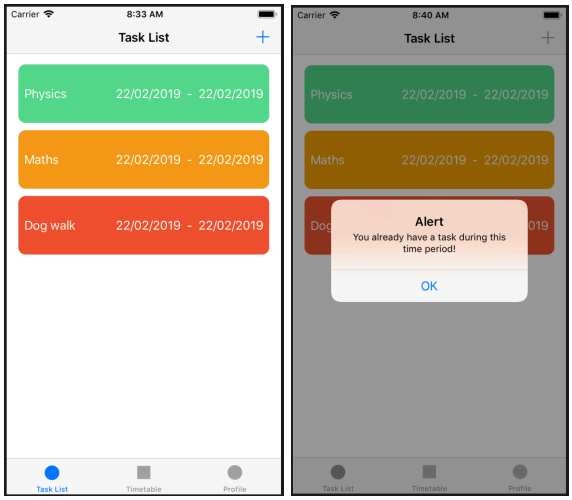


Figure 7 - Task list before and after incorrect task period entered. Error message shown.

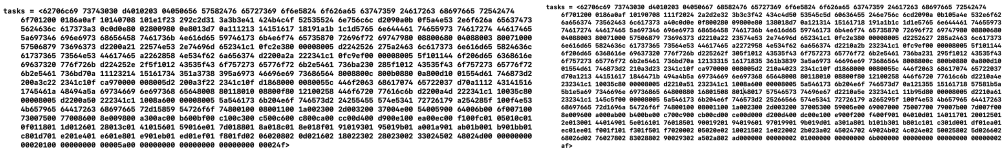


Figure 8 – Serialised task data dump before and after new task entered. Change in the serialised file indicates new task details are being stored in database.

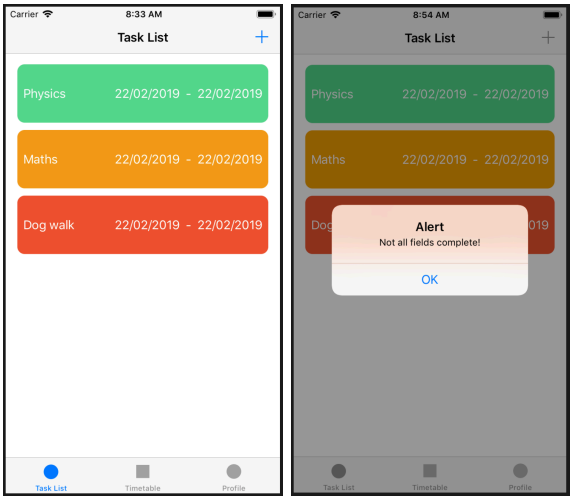


Figure 9 – Task list before and after incomplete form entered. Error message shown.

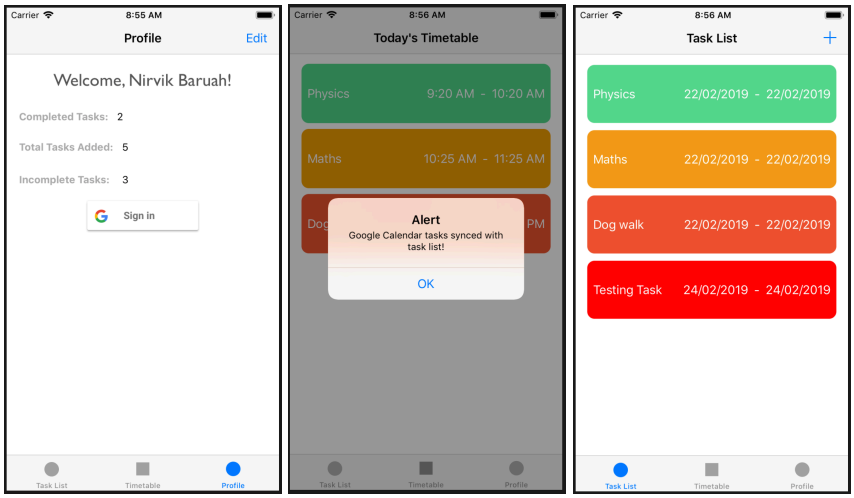


Figure 10 – Before and after 'Sign In' to Google Calendar button pressed. Alert message shown indicating tasks are synced. Google Calendar task then appears in Task List page.

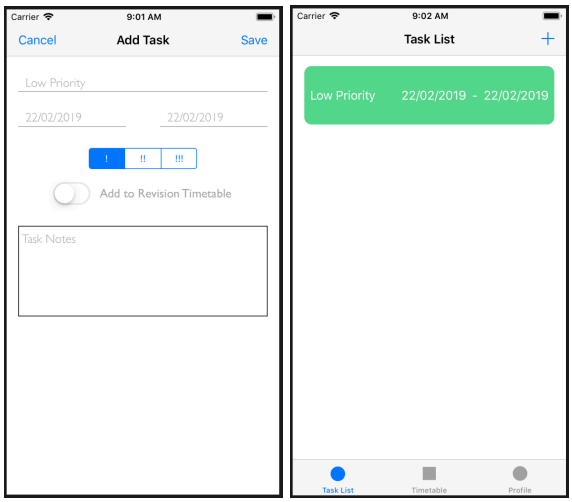


Figure 11 – Low priority task appears in task list as a green box.

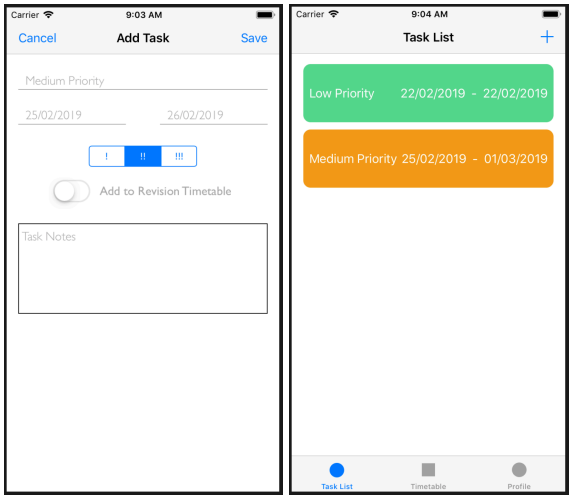


Figure 12 – Medium priority task appears as yellow box.

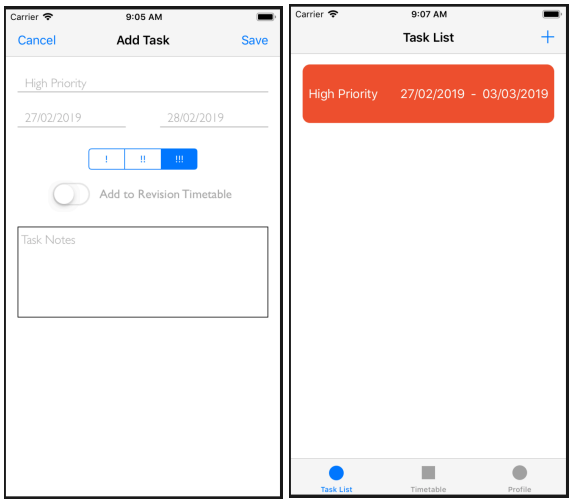


Figure 13 – High priority task appears as red box.

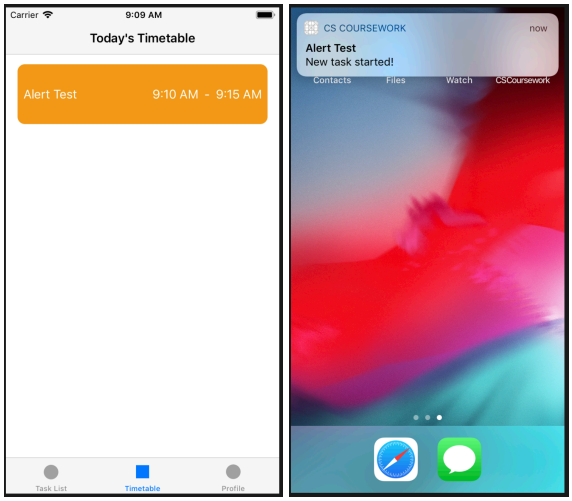


Figure 14 – Task entered for 9:10 AM. Task alert appears at 9:10 AM when app is not running.

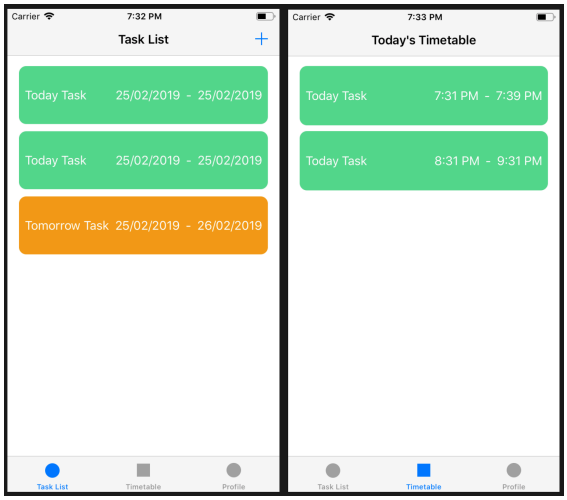


Figure 15 – Timetable page after adding a task for the following day. Timetable page remains unchanged and only displays tasks which terminate on current day.

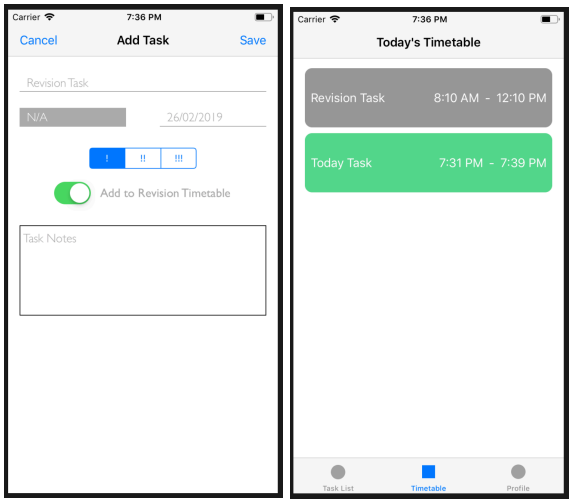


Figure 16 – User indicating task as revision task. It is automatically added to the current timetable as part of a revision timetable for the day.

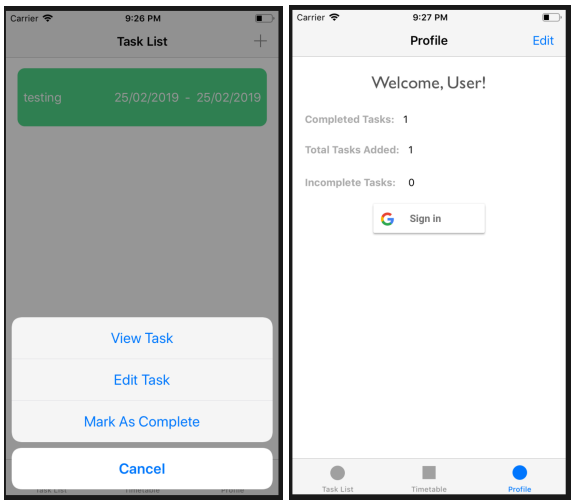


Figure 17 – Profile page after a task is marked as complete. Note how the number of completed tasks, and the total tasks added have been incremented by 1.

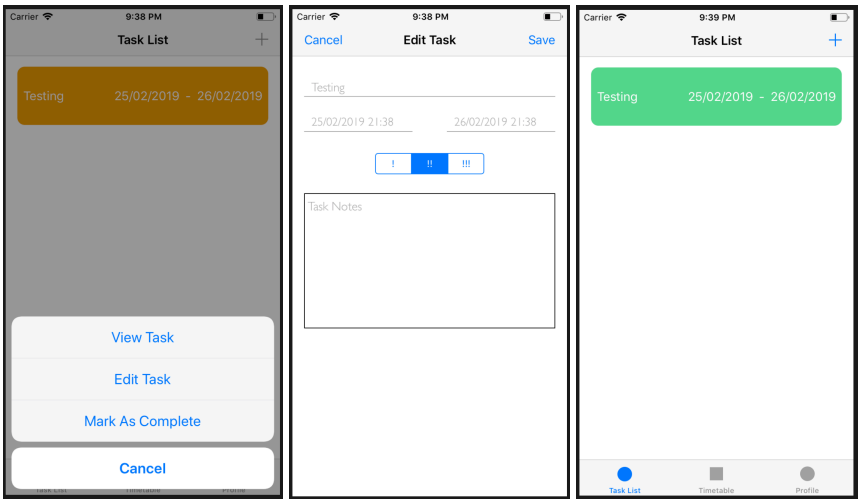


Figure 18 – Edit task form appears when 'Edit Task' button pressed. Using the form, I have changed the task's priority, as shown by the change in box colour.

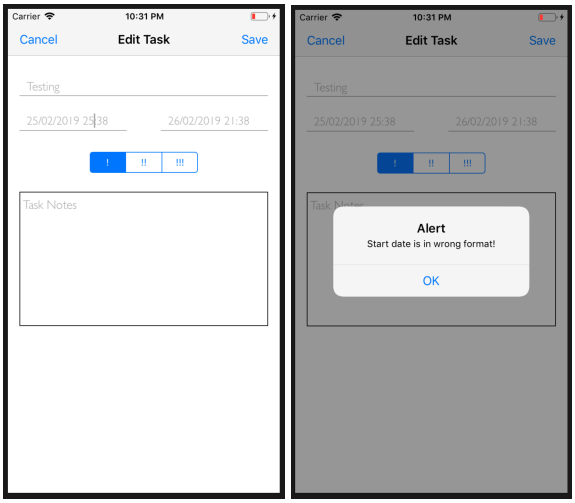


Figure 19 – Edit task form before and after attempting to enter invalid start date. Error message appears.

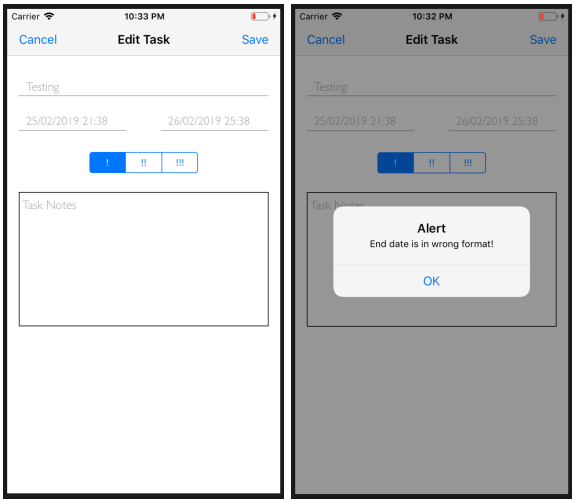


Figure 20 – Edit task form before and after attempting to enter invalid end date. Error message appears.

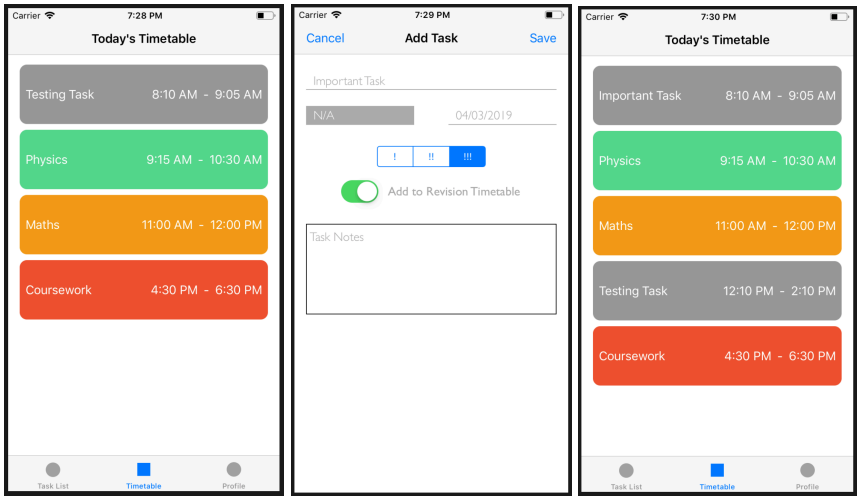


Figure 21 – Before and after task of higher importance added to revision timetable. Note how the daily timetable automatically adjusts the order of tasks to complete them in priority order.

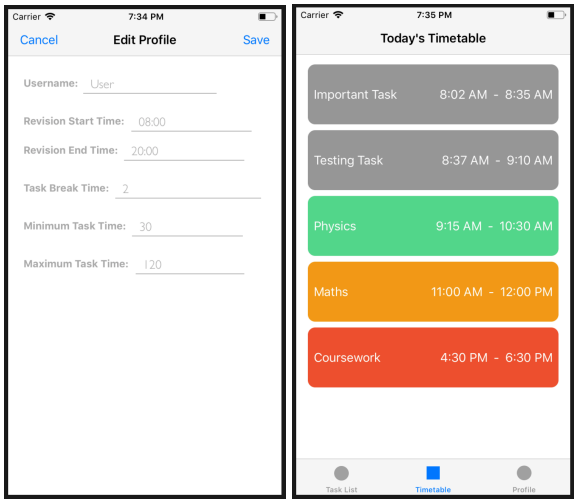


Figure 22 – Task break time set for 2 minutes. Note how timetable adjusts to leave 2 minutes between revision tasks.

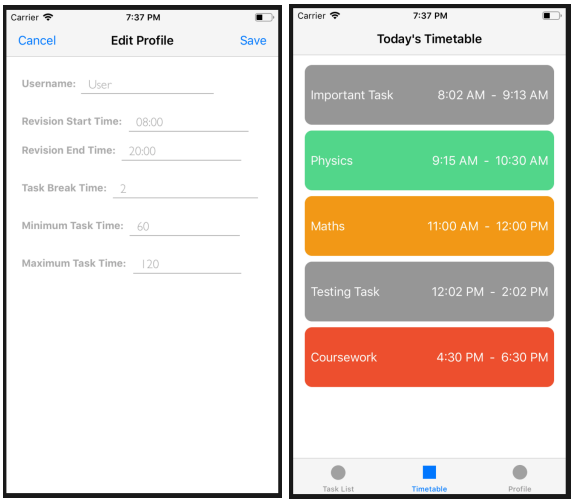


Figure 23 – Minimum revision task time set for 60 minutes. Note revision tasks are now at least 60 minutes long.

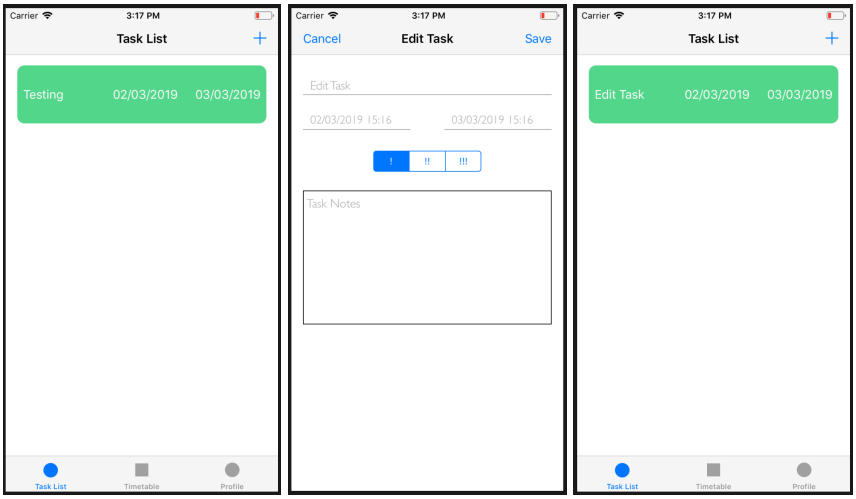


Figure 24 – Changing task name using edit task form.

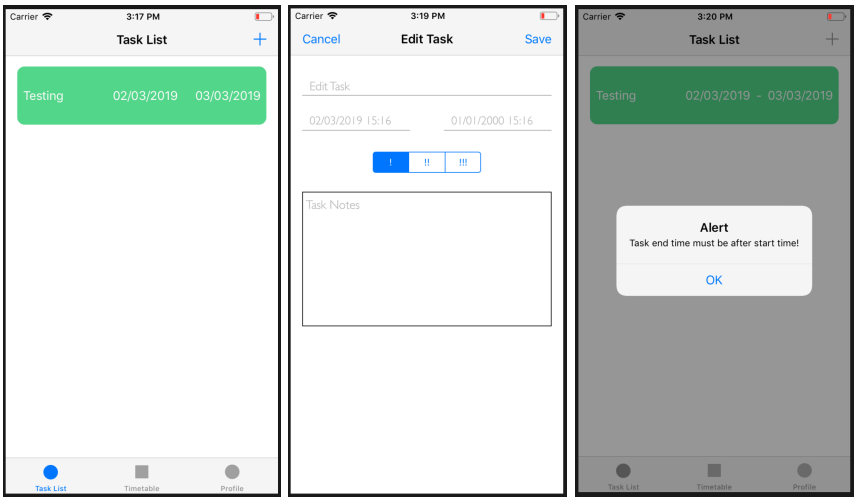


Figure 25 – Erroneous data entered into edit task form. Error message appears.

Figure 26 – Large number of tasks entered into system. Daily timetable works as expected.

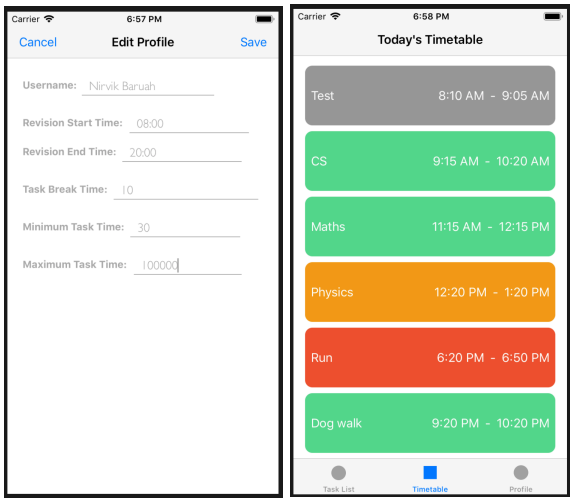


Figure 27 – Max task time set to 100000. App works as expected

Implementation

Completeness of Solution

No.	***Objective***	***Coding Page Reference***	***Testing Reference No.***	***Met/Not Met***
1	The database must store all tasks which the user enters into the application	PAGE 69, lines 101 – 104 PAGE 90	Page 27, Test 4	Met
2	The database must be able to record key information about tasks which is stated in the Data Sources section	PAGE 69, lines 101 – 104 PAGE 90	Page 26 – 27, Test 4, 1	Met
3	The proposed solution must be accessible by iPhone 8's	All code in appendices configured for iPhone 8. Code can be found from PAGE 69 – PAGE 99	All tests being run on iPhone 8	Met
4	The system must be able to connect with Google Calendar	PAGE 93, lines 27 - 44	Page 27, Test 6	Met
5	The system must sync the user's local timetable with that on their Google Calendar account	PAGE 75, lines 31 – 47, 296 - 343	Page 27, Test 6	Met
6	The system should play a 'click' noise when a button is pressed	PAGE 69 line 332	Page 29, Test 22	Met

7	Tasks should be able to be entered using a form	PAGE 69, lines 151 – 153 PAGE 90	Page 26 – 27, Test 1, 2, 3, 5, 7, 8, 9	Met
8	Tasks should be colour-coded in the timetable depending on priority	PAGE 69, lines 182 - 202 PAGE 75, lines 219 - 233	Page 27, Test 7, 8, 9	Met
9	The system should be able to handle a high number of tasks and still be functional	PAGE 69	Page 29, Test 23	Met
10	The system should automatically adjust the revision timetable when new tasks are inputted	PAGE 75, lines 61 - 135	Page 29, Test 19, 20, 21	Met
11	The application should automatically notify the user with a notification on their phone when a task has begun	PAGE 75, lines 270 - 294	Page 28, Test 10	Met
12	The system should notify the user when two tasks are inputted which occur at the same time	PAGE 69, lines 59 - 77	Page 27, Test 3	Met
13	The user should be able to view all tasks which have been entered into the system	PAGE 69	Page 26, Test 1	Met
14	The user should be able to view the full timetable for their current day	PAGE 75	Page 28, Test 11	Met
15	Students should be able to choose whether they want a task to be part of the automatically-generated revision timetable or not	PAGE 90, line 31	Page 28, Test 12	Met
16	The system should show how many tasks the user has completed on a separate profile page	PAGE 93, line 68	Page 28, Test 13	Met
17	The user should be able to edit any attribute of a task by pressing on the task object which will bring up the task edit screen	PAGE 69, line 359 - 362 PAGE 87	Page 28 – 29, Test 14, 15, 16, 17, 18	Met
18	The user should be able to easily mark a task as	PAGE 69, liens 363 - 367	Page 28, Test 13	Met

	completed by pressing a button on a task box			
19	The system should send notifications to the user even when the application is not open on their device	PAGE 75, lines 270 - 294	Page 28, Test 10	Met
20	The system should automatically set aside time between different revision subjects as a break	PAGE 75, lines 98 - 122	Page 29, Test 20	Met

Evidence of Coding Styles

No.	Style	Explanation	Code Reference
1.	Meaningful identifier names	- All my variables have meaningful names using camel case e.g in my Daily Timetable page (Appendix 5), my revision tasks array is called revisionTasks - My functions also have useful names e.g deleteTask in my Task List (Appendix 4) - Classes also have useful names e.g RevisionPriorityQueue (Appendix 15) for my priority queue class	PAGE 69, lines 18 – 34 PAGE 75, line 145
2.	Annotation used effectively when required	- I have comments throughout my code and have annotated my complex algorithms e.g my slot task algorithm from Appendix 5 is described in full using comments - I have comments describing all of my functions e.g my labelling of getter methods in Appendix 4 - All my classes have a comment at the top containing their name and the date the file was created	PAGE 69, lines 1-6, 38, 43 PAGE 75, lines 28 - 37
3.	Well designed user interfaces	- I have tried to follow the initial mock-ups as closely as possible and believe I have done so effectively e.g both my design and testing show the same colour scheme	PAGE 69 PAGE 75

		- I use segmented input buttons and toggle switches where possible to make the application smooth e.g my task input form uses a toggle switch for indicating revision tasks - I have tried to make both my initial designs and final product as clean as possible by having no overlapping elements and tables for data e.g when viewing a task the data is displayed in a table	
4.	Modularisation of code	- I have used functions to split my code up into reusable modules e.g I have a delete task function in Appendix 4 to easily delete a task by passing it as a parameter - All my functions carry out a single task e.g in my Task class (Appendix 6) my GetTask function only returns my task and nothing more - I have used functions to break up long algorithms into smaller steps e.g my SlotTasks function in Appendix 5 calls other functions such as GetMinBreakTime	PAGE 69, lines 55, 207 PAGE 75, lines 68 PAGE 81, lines 61 - 98
5.	Good use of global variables	- I have no global variables and instead pass my class variables around when switching pages e.g in Appendix 4 I have a line of code which passes my current task array at the end of a segue to another page	No global variable code references. Function to handle private variables being passed around can be found in PAGE 69, lines 329 - 343
6.	Minimal use of global variables	- I have no global variables and instead pass variables around through functions as parameters e.g in Appendix 12 I have a function called GetLatestUser to get the newest instance of my user variable from other pages	No global variable code references. Function to handle private variables being passed around can be found in PAGE 93, line 52

7.	Managed casting of types (converting variable type, and possible loss of data)	- Swift is a static-typed language which means I can't cast variables to a different type, so this is managed for me	No variable casting in my program
8.	Use of constants	- I use constants throughout my program to make it more efficient e.g in Appendix 4 I make userDefaults a constant using the keyword 'let' to assign it a constant value - I also use constants locally in functions to temporarily assign a constant value as Swift encourages this e.g in Appendix 4 I use a let statement to assign different types of alerts in my tableView function	PAGE 69, line 41, 73, 80
9.	Appropriate indentation	- All my code is appropriately indented using a tab for each new block of code. This can be seen throughout all my appendices e.g in Appendix 4 I indent all my code by a tab when declaring a new function	PAGE 69, line 17, 46 PAGE 75, line 18
10.	Self-documenting code	- As stated in No. 1 of this table, all my variables, functions, and classes have meaningful names which means they are easily understood independently of comments - As can be seen from my testing table, all my input variables also have meaningful names which describe their purpose e.g in Test 1 I have inputs called name and start date which are easily understandable	PAGE 85, line 57 - 68 PAGE 87, line 16
11.	Consistent style throughout	- I have used lowercase camel case throughout my system e.g my array called RevisionTasks in Appendix 5 and my constant called userDefaults in Appendix 4 both use camel case - My class names use uppercase camel case e.g I have classes	PAGE 75, lines 18 - 26 PAGE 83, line 14

		called TaskList and RevisionPriorityQueue - I have indented by the same amount in every file (one tab for each new block of code)	
12.	File paths parameterised	- I have no file paths in any of my code. However, I do access the user defaults section of my user's iPhone in order to retrieve stored data. I refer to this directly in Appendix 4 by directly referencing the user defaults section, granting me direct access	PAGE 69, lines 121 - 131
13.	Modules (subroutines) with appropriate interfaces	- All my functions take in appropriate variables and don't rely on global variables to function e.g in Appendix 5 my slotTasks function takes in a start time and a queue of revision tasks to determine what tasks can be slot in - All my return values are appropriate for the function e.g in Appendix 5 I have a function called findAvailableTime which returns an integer for the number of minutes	PAGE 75, lines 145 - 146, 194 - 202
14.	Loosely coupled modules (subroutines) - module code interacts with other parts of the program through its interface only	- In my code values are only updated via function return values e.g in my table view function in Appendix 4 I set my cells background colour by calling the function HexStringToUIColor - Functions are also only called using their parameters e.g in my view task page (Appendix 10) task details are updated using a single function with parameters	PAGE 69, lines 189 - 201 PAGE 89, lines 20 - 27
15.	Cohesive modules (subroutines) – module code does just one thing	- All my functions have a single purpose e.g in Appendix 4, my function HexStringToUIColor simply takes in a hex colour value as a parameters and returns a Swift UIColor	PAGE 69, lines 189 - 201 PAGE 75, lines 194 – 202, 270 - 294

		- This can also be seen in my function findAvailableTime in Appendix 5 which finds the time between two tasks - I also have complex functions such as scheduleNotifications in Appendix 5 which simply schedule when my user will be notified of a new task	
16.	Modules (collection of subroutines) – subroutines with common purpose grouped	- All my code which serves a similar purpose is grouped together under a single class to easily manage e.g all my queue management code is grouped under the class RevisionPriorityQueue - I also group modules by which page they act on under the same class e.g all my Task List modules appear on the TaskList class - I have done this for all of my pages to make my application easier to manage	PAGE 99 PAGE 97, lines 75 - 126
17.	Defensive programming General quality - Making the software behave in a predictable manner despite unexpected inputs or user actions.	- I have a number of validation checks on input forms to ensure no invalid data can be entered e.g the design for this validation can be found in my design section under Complex Algorithms - I also have a number of specific error messages for different erroneous data Test 2 and Test 16 in my testing table show different error messages from my input forms - I have also made my code readable using a consistent style throughout to ensure other developers can understand it in the future	PAGE 69, lines 55 - 98
18.	Good exception handling - Useful error messages	- Not only have I taken care of a number of different error cases in my input form but I have also output useful error messages as can be seen in Appendix 4	PAGE 69, lines 55 - 98

		where I specify the error such as if there is an overlap with another task or if a field is missing - This same exception handling happens when I edit a task e.g in Tests 16-18 you can see what happens when invalid information is entered into my edit task form	

Techniques Used – Data Models

Queues and Priority Queues

Throughout my project, I have used both queues and priority queues to store tasks. I have used them as they are a First In First Out structure so they make sense for a daily timetable. All of my tasks are pushed onto a queue so that they are organised in chronological order and can be pushed off easily once completed. Similarly, my revision tasks have been added to a priority queue. This is so that the most important task can be stored at the front of the queue and my daily timetable page can pop off the next task when there is some free time during the day.

An example of my queue object is shown in the code below, taken from my Task List page (Appendix 4). As highlighted, when a new task is added it is pushed to the end of my queue.

```
func addTask(title: String, startDate: Date, endDate: Date, priority: Int, notes: String, isRevisionTask: Bool)
    {
    //Check if append normal Task object or a RevisionTask object
    if (isRevisionTask){
        tasks.append(RevisionTask(taskName: title, start: startDate, end: endDate, final: endDate, taskNotes:
            notes, isFinished: false, priority: priority))
        RevisionTask.SetUser(newUser: user!)
        //Push to priority queue
        revisionTasks.Push(task: tasks.last as! RevisionTask)
    } else{
        //Push to normal queue
        tasks.append(Task(taskName: title, startDate: startDate, endDate: endDate, taskNotes: notes,
            isFinished: false, priority: priority))
    }
    let indexPath = IndexPath(row: tasks.count - 1, section: 0)
    tableView.insertRows(at: [indexPath], with: .left)
    //Increments static user variable for num tasks done
    user!.IncrementTasksAdded()
}
```

To store my priority queue, I have created a separate priority queue class (Appendix 15). This contains all appropriate queue functionality such as pop, peek, push, and sort. Below is a screenshot of my push method. In this method, I push my task to the end of the queue. Once I do this, I increment my tail pointer by one to always point to the next free location. Finally, I sort my queue by priority order so the most important task is always first.

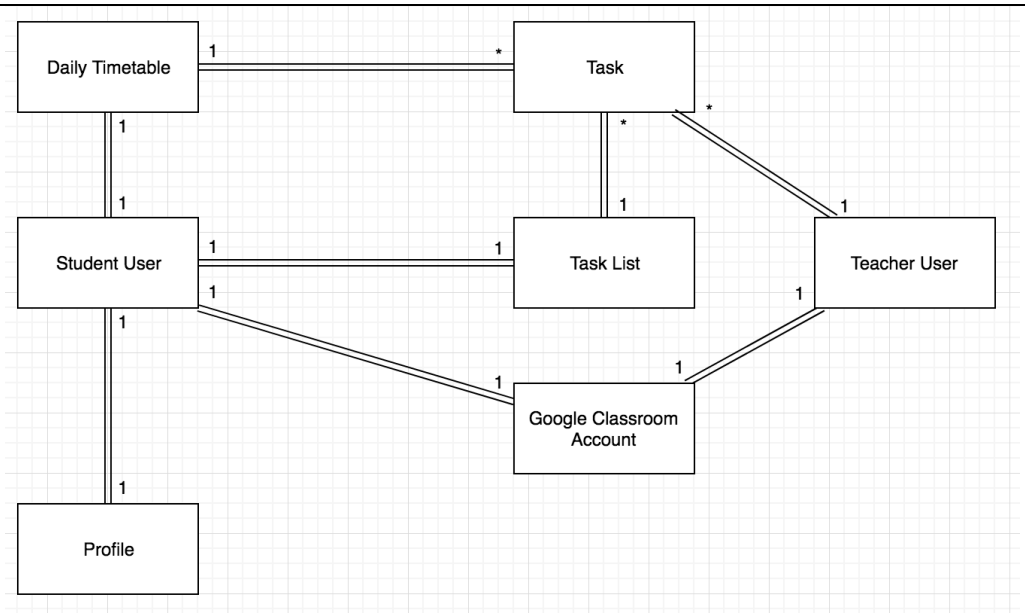
```
//Push
func Push(task: RevisionTask){
    self.queue.append(task)
    self.tailPointer += 1
    Sort()
}
```

This priority queue is essential in other parts of my project. Below is a screenshot of one of my complex algorithms in my daily timetable page (Appendix 5) where I slot tasks into free slots during the day. Since this is a long algorithm, I have highlighted the important line of code. Since I have a priority queue which stores the most important task at the front, I simply need to pop a task from the queue when I have a free slot of time. This is what the algorithm below achieves.

```
func slotTasks(timeAvailable: Int, startTime: Date, taskListCopy: RevisionPriorityQueue) -> RevisionPriorityQueue{
    let minTaskTime = user!.GetMinTaskTime()
    let maxTaskTime = user!.GetMaxTaskTime()
    let breakTime = user!.GetMinBreakTime() * 2
    var numRevTasksToDo = taskListCopy.GetCount()
    if (numRevTasksToDo > 0 && timeAvailable >= minTaskTime + breakTime){
        var timePerTask = timeAvailable / numRevTasksToDo
        while ((timePerTask > maxTaskTime + breakTime || timePerTask < minTaskTime + breakTime) && numRevTasksToDo > 1){
            numRevTasksToDo -= 1
            timePerTask = timeAvailable / numRevTasksToDo
        }
        //Check if successful in dividing time - otherwise just allocate max/min time per task
        if (timePerTask > maxTaskTime + breakTime){
            timePerTask = min(timeAvailable, maxTaskTime + breakTime)
            numRevTasksToDo = taskListCopy.GetCount()
        }
        else if (timePerTask < minTaskTime + breakTime){
            timePerTask = max(timeAvailable, minTaskTime + breakTime)
            numRevTasksToDo = 1
        }
        //Add half the break time in minutes at start for break
        var taskStartTime = combineDateWithTime(date: Date(), time: startTime.addingTimeInterval(Double((breakTime/2) * 60)))
        //Take away half the break time as half the break time used at start and end for break
        var taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask - (breakTime)) * 60))
        for _ in 1...numRevTasksToDo{
            taskListCopy.Peek()!.SetStartDate(date: taskStartTime!)
            taskListCopy.Peek()!.SetEndDate(date: taskEndTime)
            tasks.append(taskListCopy.Pop())
            //Add 5 minutes between tasks as break
            //Take away 5 minutes at end as total time per task includes break
            //Multiply by 60 to convert minutes to seconds
            taskStartTime = taskEndTime.addingTimeInterval(Double((breakTime/2) * 60))
            taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask - (breakTime)) * 60))
        }
    }
    //Must sort after every slotting as can add any number of tasks so not a case of simple swapping of elements
    tasks = tasks.sorted(by: { $0.GetStartDate() < $1.GetStartDate() })
```

Complex scientific/mathematical/robotics/control/business model

My project uses a complex business model with a number of different actors who need to work together to make it successful. Below I have attached my actor diagram.



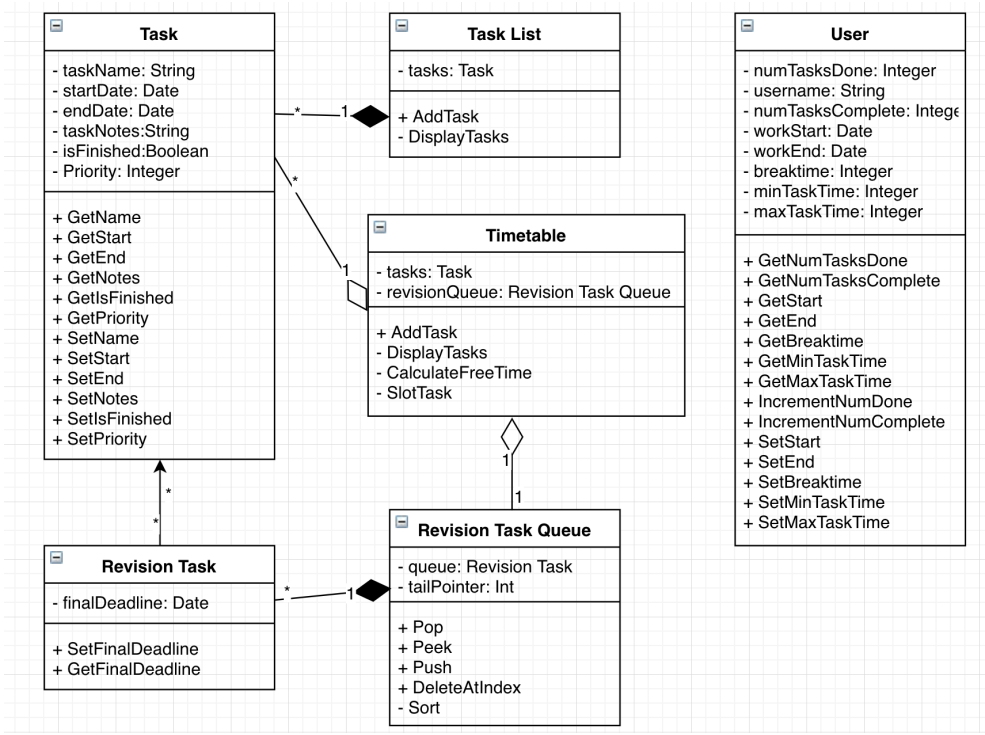
As you can see, my coursework not only relies on external APIs such as Google Classroom but it also relies on student users and teacher users interacting through the Google Classroom interface. I hope to use my application to bridge this gap and allow them to communicate through a second interface, namely my iPhone app. Below I have attached an image of my app connecting to the Google Calendar API. This is taken from my application delegate file (Appendix 7).

```
//Google Sign in
func application(_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
    return GIDSignIn.sharedInstance().handle(url as URL?,
                                             sourceApplication: options[UIApplication.OpenURLOptionsKey.sourceApplication] as? String,
                                             annotation: options[UIApplication.OpenURLOptionsKey.annotation])
}

private func application(application: UIApplication,
                 didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    // Initialize sign-in
    GIDSignIn.sharedInstance().clientID = "750611619941-1s590l62jchqpg8o2if1sp6c3oufvttg.apps.googleusercontent.com"
    GIDSignIn.sharedInstance().delegate = self as? GIDSignInDelegate  Conditional cast from 'AppDelegate' to 'GIDSignInDelegate' always succeeds

    return true
}

func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!,
          withError error: Error!) {
    if let error = error {
        print("\(error.localizedDescription)")
    }
}
```

In addition to my multiple actors, I also have a variety of use cases for my project. For example, some users might use my application as a simple reminders service. Other users can use my application to manage their revision timetables. These varied use cases for my varied actors creates a complex business model for me to manage. This, coupled with my complex user-defined algorithms adds greatly to the complexity of my app.

Complex user defined use of OOP e.g. classes, inheritance, polymorphism

Throughout my project, I have used Object Oriented Programming to manage my various pieces of data. Below I have pasted my class diagram created using UML.



As you can see, I have used complex aspects of Object-Oriented Programming. For example, I have both association and composition aggregation, as well as a inheritance structure. This is as my Task List class is made up entirely of Task objects, resulting in composition aggregation. However, my Timetable is made up of both tasks and revision tasks which means it uses association aggregation with both of these classes.

I also have used techniques such as polymorphism. For example, in the below code taken from my revision priority queue class (Appendix 15) I have defined two different constructors. These will create a different object depending on whether an existing queue is passed to the constructor or not. Polymorphism such as this is essential for me to handle the various different data inputs the user can enter. As such, it is used throughout my program.

```
//Polymorphism of constructors
override init() {
    self.tailPointer = 0
}

init(queue: [RevisionTask], pointer: Int){
    self.queue = queue
    self.tailPointer = pointer
}
```

My OOP model is also user-dependent. Below I have attached code from my Task List page (Appendix 4). This code creates an object. However, depending on the user input it either creates a revision task object or a task object. This is one example of many where I have complex, user defined object-oriented code.

```
func addTask(title: String, startDate: Date, endDate: Date, priority: Int, notes: String, isRevisionTask: Bool)
    {
    //Check if append normal Task object or a RevisionTask object
    if (isRevisionTask){
        tasks.append(RevisionTask(taskName: title, start: startDate, end: endDate, final: endDate, taskNotes:
            notes, isFinished: false, priority: priority))
        RevisionTask.SetUser(newUser: user!)
        //Push to priority queue
        revisionTasks.Push(task: tasks.last as! RevisionTask)
    } else{
        //Push to normal queue
        tasks.append(Task(taskName: title, startDate: startDate, endDate: endDate, taskNotes: notes,
            isFinished: false, priority: priority))
    }
    let indexPath = IndexPath(row: tasks.count - 1, section: 0)
    tableView.insertRows(at: [indexPath], with: .left)
    //Increments static user variable for num tasks done
    user!.IncrementTasksAdded()
}
```

Files organised for direct access

As stated previously, my application does not use a database to store tasks. Instead, it saves data directly on the iPhone as a serialised text file in the User Defaults section of IOS. This file is organised for direct access as it is always stored in the same location in User Defaults and my application simply has to retrieve it when it is starting up. Screenshots of my file access methods have been pasted below. In the first image, I show how I serialise my task data to conform to the IOS storing protocols. This allows my object data to be stored locally.

```
required convenience init(coder aDecoder: NSCoder) {
    let priority = aDecoder.decodeInteger(forKey: "priority")
    let taskName = aDecoder.decodeObject(forKey: "taskName") as! String
    let endDate = aDecoder.decodeObject(forKey: "endDate") as! Date
    let startDate = aDecoder.decodeObject(forKey: "startDate") as! Date
    let taskNotes = aDecoder.decodeObject(forKey: "taskNotes") as! String
    let isFinished = aDecoder.decodeBool(forKey: "isFinished")
    self.init(taskName: taskName, startDate: startDate, endDate: endDate, taskNotes: taskNotes, isFinished: isFinished, priority:
        priority)
}

func encode(with aCoder: NSCoder) {
    aCoder.encode(taskName, forKey: "taskName")
    aCoder.encode(priority, forKey: "priority")
    aCoder.encode(startDate, forKey: "startDate")
    aCoder.encode(endDate, forKey: "endDate")
    aCoder.encode(taskNotes, forKey: "taskNotes")
    aCoder.encode(isFinished, forKey: "isFinished")
}
```

My second image shows how I retrieve the data once it is stored on my device. This is done directly as I know exactly where the data is stored and simply need to retrieve the appropriate object. The highlighted line shows that I know the task file is stored in User Defaults and I am simply retrieving the appropriate file object from the folder. The subsequent code iterates over the retrieved file and copies it into the application.

```
var decoded  = userDefaults.object(forKey: "tasks") as! Data?
if(decoded != nil){
    let decodedTasks = NSKeyedUnarchiver.unarchiveObject(with: decoded!) as! [Task]
    tasks = decodedTasks

    for element in tasks{
        if (element is RevisionTask){
            revisionTasks.Push(task: element as! RevisionTask)
        }
    }
}
```

Techniques Used – Algorithms

Queue Operations

As stated previously, my coursework relies on a number of queues to manage tasks throughout the day. This includes a priority queue which I use to store revision tasks. To manage my queues, I store all my data in an array as a Task object. To store my priority queue, I have a separate priority queue class (Appendix 15). This class manages the queue as a whole, and it is made up of Task objects. An excerpt of this class is pasted below, which shows the implementation for all my queue operations. I call these operations throughout my program to manage my daily timetable.

```
//Pop
func Pop() -> RevisionTask{
    self.tailPointer -= 1
    let firstElement: RevisionTask = self.queue[0]
    self.queue.remove(at: 0)
    return firstElement
}

//Peek
func Peek() -> RevisionTask?{
    return self.queue.first
}

//Push
func Push(task: RevisionTask){
    self.queue.append(task)
    self.tailPointer += 1
    Sort()
}

//Delete
func Delete(task: RevisionTask){
    //Removes revision task element from queue
    self.queue = self.queue.filter {$0 != task}
    self.tailPointer -= 1
}
```

Complex user-defined algorithms

I have a number of complex user-specific algorithms throughout my coursework. One of my main algorithms slots revision tasks in between my user's normal daily timetable. For example, my user might have 5 lessons spread throughout the day. This first algorithm will slot tasks in the gaps between these lessons by priority order. This is a complex algorithm for a number of reasons. Firstly, since my user can input a virtually unlimited

number of tasks and revision tasks for a given day, my algorithm will have to scan all gaps between these tasks in addition to prioritising the revision tasks by both date and priority. My algorithm will also have to account for the duration of time between each task as there is no point spending too little time or too much time on one task. The user can adjust the minimum and maximum task time in the profile page which adds to the complexity of this user-defined algorithm.

I have attached the code of this algorithm below, taken from my Daily Timetable page (Appendix 5). This code automatically slots in revision tasks into the gap between two tasks; the revision tasks are slotted in priority order using the priority queue from above.

```
func slotTasks(timeAvailable: Int, startTime: Date, taskListCopy: RevisionPriorityQueue) -> RevisionPriorityQueue{
    let minTaskTime = user!.GetMinTaskTime()
    let maxTaskTime = user!.GetMaxTaskTime()
    let breakTime = user!.GetMinBreakTime() * 2
    var numRevTasksToDo = taskListCopy.GetCount()
    if (numRevTasksToDo > 0 && timeAvailable >= minTaskTime + breakTime){
        var timePerTask = timeAvailable / numRevTasksToDo
        while ((timePerTask > maxTaskTime + breakTime || timePerTask < minTaskTime + breakTime) && numRevTasksToDo > 1){
            numRevTasksToDo -= 1
            timePerTask = timeAvailable / numRevTasksToDo
        }
        //Check if successful in dividing time - otherwise just allocate max/min time per task
        if (timePerTask > maxTaskTime + breakTime){
            timePerTask = min(timeAvailable, maxTaskTime + breakTime)
            numRevTasksToDo = taskListCopy.GetCount()
        }
        else if (timePerTask < minTaskTime + breakTime){
            timePerTask = max(timeAvailable, minTaskTime + breakTime)
            numRevTasksToDo = 1
        }
        //Add half the break time in minutes at start for break
        var taskStartTime = combineDateWithTime(date: Date(), time: startTime.addingTimeInterval(Double((breakTime/2) * 60))
        //Take away half the break time as half the break time used at start and end for break
        var taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask - (breakTime)) * 60))
        for _ in 1...numRevTasksToDo{
            taskListCopy.Peek()!.SetStartDate(date: taskStartTime!)
            taskListCopy.Peek()!.SetEndDate(date: taskEndTime)
            tasks.append(taskListCopy.Pop())
            //Add 5 minutes between tasks as break
            //Take away 5 minutes at end as total time per task includes break
            //Multiply by 60 to convert minutes to seconds
            taskStartTime = taskEndTime.addingTimeInterval(Double((breakTime/2) * 60))
            taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask - (breakTime)) * 60))
        }
    }
    //Must sort after every slotting as can add any number of tasks so not a case of simple swapping of elements
    tasks = tasks.sorted(by: { $0.GetStartDate() < $1.GetStartDate() })
    return taskListCopy
}
```

Dynamic generation of objects – OOP model

My project relies heavily on OOP principles in order to manage user input and data. In addition to this, my system constantly generates new objects dynamically to keep up to date with new user inputs. For example, when a user enters a new task using the task input form my project will dynamically create a new Task or Revision Task object and push this to the task queue. Another example of this is when my user logs into Google Calendar. My system creates a new User object when this happens. I have attached the code of my task list input below to show how my system creates new objects dynamically.

```
func addTask(title: String, startDate: Date, endDate: Date, priority: Int, notes: String, isRevisionTask: Bool)
    {
    //Check if append normal Task object or a RevisionTask object
    if (isRevisionTask){
        tasks.append(RevisionTask(taskName: title, start: startDate, end: endDate, final: endDate, taskNotes:
            notes, isFinished: false, priority: priority))
        RevisionTask.SetUser(newUser: user!)
        //Push to priority queue
        revisionTasks.Push(task: tasks.last as! RevisionTask)
    } else{
        //Push to normal queue
        tasks.append(Task(taskName: title, startDate: startDate, endDate: endDate, taskNotes: notes,
            isFinished: false, priority: priority))
    }
    let indexPath = IndexPath(row: tasks.count - 1, section: 0)
    tableView.insertRows(at: [indexPath], with: .left)
    //Increments static user variable for num tasks done
    user!.IncrementTasksAdded()
}
```

Calling parameterised web service APIs and parsing JSON/XML to service a complex client-server model

A key feature of my project is that it allows a user to sync their Google Calendar timetable with their daily timetable in order to more effectively plan revision timetables. I achieve this by calling the Google Calendar API and passing it a number of parameters such as how far into the future I want to retrieve Google Calendar tasks from. The API authenticates this call with the user sign-in token and, if approved, returns the user's Google Calendar tasks as JSON. I am then able to parse the JSON and retrieve only the relevant bits of data I need for my project. I have pasted the code below of my API call and my JSON handling. This is taken from my Daily Timetable page (Appendix 5).

```
//Only get events from now until a week later
let startDateTime = GTLRDateTime(date: Calendar.current.startOfDay(for: Date()))
let endDateTime = GTLRDateTime(date: Date().addingTimeInterval(60*60*24*7))
let eventsListQuery = GTLRCalendarQuery_EventsList.query(withCalendarId: calendarId)
eventsListQuery.timeMin = startDateTime
eventsListQuery.timeMax = endDateTime

_ = service.executeQuery(eventsListQuery, completionHandler: { (ticket, result, error) in
    guard error == nil, let items = (result as? GTLRCalendar_Events)?.items else {
        return
    }

    for item in items{
        var isNew = true
        let taskName = item.summary
        let startTime = item.start!.dateTime!.date
        let endTime = item.end!.dateTime!.date
        let priority = 3
        for prevItems in self.taskListReference!.GetTasks(){
            if prevItems.GetName() == taskName{
                isNew = false
                break
            }
        }
        if isNew{
            self.taskListReference?.addTask(title: taskName!, startDate: startTime, endDate: endTime, priority: priority, notes:
                "Task Notes", isRevisionTask: false)
        }
    }

    if items.count > 0{
        let alert = UIAlertController(title: "Alert", message: "Google Calendar tasks synced with task list!",
            preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action -> Void in}))
        self.present(alert, animated: true, completion: nil)
    }
})
```

Evaluation

Objective Analysis

Objective	***Met?***	***Comments***	***Future Considerations***
The database must store all tasks which the user enters into the application	Met	Every time a new task is added or edited successfully using the appropriate form, the task is added to an array. This array is then serialized and all tasks within it are stored as a file in the local User Defaults section of the iPhone.	I could have stored the data on a database online rather than on the local device. This would have allowed users to access their timetable from any device by logging into the application. However, I feel this is beyond the scope of my project and that my implementation was appropriate for my client's needs. Storing it in this way also means the user cannot access the database directly but instead has to go through my app, improving data integrity.
The database must be able to record key information about tasks which is stated in the Data Sources section	Met	All information about tasks is stored in a Task object which is stored in a task array. All the properties of this array are serialized and stored in the local User Defaults section.	I feel my implementation was perfect for the purposes of this objective. This is as all information the user enters is stored so when they access the app next time it reloads all the data as it was before. Serializing my task array is also an efficient method as it means all my tasks are stored under a common identifier, allowing direct access.
The proposed solution must be accessible by iPhone 8's	Met	When the application is opened on an iPhone 8, it loads as per expected with the same UI as can be found in my Testing section.	When my application is opened on other iPhones, it still runs but has an unusable UI. This is as I have configured the placement of my UI elements for an iPhone 8 so they do not adjust for a different sized phone. This is useable for my client as

			he only uses an iPhone 8 and wants his timetable manager on this. However, were I planning on distributing this app wider I might have set it up so that it would alert the user beforehand that the application is not configured for their phone. Alternatively, I could have made my app responsive for all iPhone and iPad sizes. Both of these methods would have required a lot of work though which is why I chose to focus on my client's needs and only program an app for an iPhone 8.
The system must be able to connect with Google Calendar	Met	When the system starts up, the app loads the Google Calendar and Google Sign In API in the background. Once the user visits the profile page, there is a button asking them to sign into Google. If they sign successfully, it connects to their public Google Calendar and allows my app to view their events. I have configured my Google Calendar API to request view-only rights.	I feel my method of connecting to Google Calendar is appropriate for my client's needs. Through my easy-to-find login button, I have allowed my client to easily sync his tasks. The only difference I might consider making is to ensure that the user stays logged in on Google if they access it soon after shutting the app down. I opted not to enable this feature on the API as I wanted to ensure my user's Google account was as secure as possible. However, should my client wish for his login details to be cached I could easily implement this.
The system must sync the user's local timetable with that on their	Met	Once the user has connected to Google Calendar as per the process above, the Task List page class will automatically	I feel my implementation was a good start to fulfilling my user's needs but I could have done more to make it

Google Calendar account		check with Google Calendar whether the user has any tasks in the next week. If they do, the task list will update and send the user an alert.	more useful. I firstly feel I could have taken more user input into this process. For example, I only checked for tasks a week into the future as I felt this was appropriate. However, my client may have other ideas so I could have allowed the user to determine how far in the future the application should check. I could have also had the Task List check for Google Calendar tasks when a button is clicked. This is as when there is no user action to sync a task and it does it automatically, it is hard to tell if there are just no tasks in Google Calendar or the system is malfunctioning. With a button click, the user has an immediate response so they can be sure that there are no Google Calendar tasks.
The system should play a 'click' noise when a button is pressed	Met	Every time a button is pressed on any page of the application, a library is called which plays an audio source file. This audio source is the default Apple keyboard click noise. This results in a soft 'click' being played every time a button is pressed to give the user immediate feedback on their action.	I feel my current implementation works very well in improving my app's user experience as there is no latency between the click event and the sound. However, I could extend it a bit further to improve it. For example, instead of using the same keyboard click noise for every button, I could have used different noises to indicate different actions.
Tasks should be able to be entered using a form	Met	When the user is on the task list page and presses the plus button on the top right of their screen, a modal input form appears with an animation. The	I believe my input form is appropriate and meets this objective as it allows the user to easily and efficiently enter in data

		user can then enter all appropriate task data using the various input fields. Once submitted, the task input form is checked for completeness and correctness using the algorithm that can be found in Appendix 4. Finally, if the task is valid it is added to the Task queue and appears in the task list page.	about his tasks. It is also very user-friendly and gives specific error messages - it will alert the user what the exact problem is with the task. Therefore, I do not think my implementation needs any additional work as it is very suitable for this application.
Tasks should be colour-coded in the timetable depending on priority	Met	When the user is entering task data using the task input form, they can specify task priority using a segmented input controller. This data is passed along with the rest of the task data and means when a low priority task is entered it appears as a green box in both the task list and the daily timetable. An orange box is used for a medium priority task and a red box for a high priority task. Revision tasks are treated separately and appear as grey boxes.	I feel my colour-coding system worked well in easily identifying which tasks were important and which weren't in my task list for the user. The only potential extension I might have added to my implementation was in allowing more user customizability to it. For example, other timetable apps in the market allow users to choose what colors should represent task priorities. I could do the same if I wanted to take this project further.
The system should be able to handle a high number of tasks and still be functional	Met	My system works smoothly when a large number of tasks are entered into the system. I implemented this in a few ways. Firstly, I was efficient with my memory management. Rather than creating separate task variables for each page, I would pass the same array around by reference to ensure only one copy existed at any given time. I would also pass arrays around by reference rather than by value when I could to minimize the number of new objects I was creating. Finally, I reduced app background activities as much	I feel my implementation allowed my app to work for a large number of tasks and it does this very well. I would not change anything with this implementation.

		as possible by running most of them when the application initially started to avoid multiple processes occurring simultaneously.	
The system should automatically adjust the revision timetable when new tasks are inputted	Met	When a new revision task is added, the Daily Timetable page changes if the priority order of tasks is affected. This change happens immediately which means when the user visits the Daily Timetable page next the transition between the new timetable is seamless.	I am very happy with my implementation as the user experiences no delay or lags when visiting the Daily Timetable page after adding a task – they only see the new timetable. The only possible extension I could add to this is adding an animation to the Daily Timetable page so that when a new task appears it is animated and slides into place. However, this is certainly not essential and could even be detrimental as it reduces the efficiency and speed of my system.
The application should automatically notify the user with a notification on their phone when a task has begun	Met	When a task is added to the Daily Timetable, it adds the start-time of the task to an alert queue. This queue pops an alert every time a start-time is reached. This results in my user being notified of a new task starting, even when the app is not currently open on their phone. This queue is cleared every time a new task is added to ensure the alert queue reflects the most recent timetable.	I am very happy with my implementation of my alert system. Not only does it stay up to date with my user's timetable but it does so seamlessly and sends alerts even when the app is not running. The only thing I might change is making notifications more frequent for high-priority tasks. For example, the built-in reminders app on iPhones sends out notifications an hour before a task if they are important. The reason I didn't implement this was as I did not want to spam the user's phone with alerts while they were meant to be revising.
The system should notify the user when two tasks	Met	When the user submits the task input form, an algorithm is carried out to validate the data	I believe my implementation for this objective is perfect as it

are inputted which occur at the same time		that is entered. This can be found in Appendix 4. When two tasks are entered which overlap in time period, an alert message is shown to the user.	cleanly and quickly identifies potential task clashes and notifies the user. I would not change any aspect of it.
The user should be able to view all tasks which have been entered into the system	Met	When a new task is successfully entered into the system, it appears in the Task List page. This is my application's home page. Here, the user can view all tasks which have been entered into the system. A task is only removed from this page once it is marked as complete.	I am happy with my Task List page as it displays all the tasks in a clean, user-friendly fashion. The only thing I might change is the sorting order of my tasks. Right now, tasks are listed by the date of creation which means new tasks are added to the bottom of the page. I could have added an option in the profile page to list tasks in chronological order instead.
The user should be able to view the full timetable for their current day	Met	The second page in my app is a Daily Timetable page. Every time a new task is added, the Daily Timetable page scans through the Task List page and finds all tasks which are due on the current day. It adds these tasks to a daily timetable tasks array. It then sorts these tasks in chronological order to allow the user to view their latest timetable for the day easily.	I feel my implementation is appropriate for my application. This is as it fulfills the objective efficiently by only running the algorithm when a new task is entered. I would not change any aspect of my implementation.
Students should be able to choose whether they want a task to be part of the automatically-generated revision timetable or not	Met	In my task input form, the user can select using a checkbox whether they want the task to be part if the automatically-generated revision timetable or not. If this checkbox is selected, the 'Start Date' input option is disabled as the system will set a start time itself. The task appears as a grey box in the Task List and Daily Timetable pages.	I am happy with my implementation as it allows the user to quickly declare a new revision task. I would not change how I implemented this objective.
The system should show how many tasks the user has	Met	On the profile page of my application, several statistics are shown in a table. This	My implementation meets this objective very well. Not only is the number synced

completed on a separate profile page		includes a number indicating how many tasks the user has completed. When a new task is marked as completed by pressing the task and pressing the complete button, this number increments immediately.	seamlessly with my Task List but it is also very easy to find. The only thing I might have added in conjunction with this objective is to display other analytics as I feel the 3 statistics I have on the profile page right now are quite superficial.
The user should be able to edit any attribute of a task by pressing on the task object which will bring up the task edit screen	Met	When the user presses a task, a panel of buttons slides up from the bottom of the screen. If the user presses the 'Edit Task' button, it brings up an input form with all the selected task details entered in. The user can change any of these properties by pressing on them and then clicking the submit button. Like the new task input form, the data is first validated before being added to the Task queue.	I am quite happy with how my edit task functionality turned out as it is a smooth and clean way of allowing the user to change task data. The only thing I would change with it is making it clearer to the user that they need to click on the task box. This is as the task box could be misinterpreted as being a static element, so some users may be unaware that they may need to click on it to access additional functionality. I could add a symbol on the box to indicate it is a clickable element.
The user should be able to easily mark a task as completed by pressing a button on a task box	Met	When the user presses a task, a panel of buttons slides up from the bottom of the screen. If the user presses the 'Mark Task As Complete' button, the task disappears from both the Task List and the Daily Timetable with an animation. The 'number of tasks completed' statistic on the profile page is also incremented.	I am happy with how I have implemented this objective. As above, the only thing I would consider changing is making it clear that the task box is a clickable element rather than a static one. However, I am very happy with my complete task animation and implementation.
The system should send notifications to the user even when the application is not	Met	When a task is added to the Daily Timetable, it adds the start-time of the task to an alert queue. This queue pops an alert every time a start-time is reached. This results in my	I am very happy with my implementation and would not change any aspect of it. This is as it does its job perfectly and it notifies the user to switch task even

open on their device		user being notified of a new task starting, even when the app is not currently open on their phone as the alert queue is built into the iPhone operating system.	when they shut down the task.
The system should automatically set aside time between different revision subjects as a break	Met	In the profile page, the user can specify how long the break between tasks should be in minutes. Every time this value is changed, or every time a new task is added, the Daily Timetable page runs an algorithm which can be found in Appendix 5. This algorithm finds the best time period for each task and automatically schedules in a break between one task and the next. This ensures the user has some time to relax before the next task.	I am happy with my implementation as it allows the user to easily adjust how long they can relax in between tasks. The only thing I might consider adding is an option to edit the task break time directly from the Daily Timetable page. This is as going through the profile page to change this information is somewhat convoluted. However, I think my system achieves this goal very well.

User Feedback Interview

***NIRVIK*: How easy is the system to use?**
***CLIENT*:** I found the system extremely to use once I downloaded it onto my personal iPhone 8. The design of the app was very intuitive. For example, I knew exactly how to add a task even though I had never seen the system before due to easily recognizable symbols such as the 'plus' button at the top of the app. The system was also very forgiving to my mistakes. When I tried to add a task but forgot to fill in one of the fields in the form, the system immediately notified me with a detailed error saying exactly what was wrong.

The system was also able to integrate seamlessly into my day to day activities. Because of how easily I could sign in [to Google], I could easily get my two calendars to sync together without the need for a new account. I also liked how the system gave me alert messages every time a new task was starting as it allowed me to plan my day without spamming me with alerts unlike other apps.

***NIRVIK*: Great. Next, how does the system meet your objectives?**
CLIENT: Well, the system delivers what I had specified at the start of this project. I find it to be easy to use and it plans my revision timetables for me similar to how I would do it manually. It also solves many of the problems I used to face before as it allows me to edit nearly any part of my revision process and constantly tinker around with it. When testing it out a few weeks ago, I was impressed by how well it handled my input and how it created timetables even when I threw some weird parameters in. So… yeah, I think all in all the system does a pretty good job.

NIRVIK: **How easy was the system to setup?**
CLIENT: It was actually super easy to setup. In fact, it took me less than a minute to have it running on my iPhone! All I had to do was download the app. Then a notification came up asking if I wanted to allow the app to send me alerts and requesting some permissions and stuff. From there on, I could add tasks and the system would spit out revision timetables. The [Google] sign in feature was also easy to setup and it was also optional so I could do it whenever I wanted to. All I had to do was go to my profile and press the button.

NIRVIK: **Do you have any criticisms?**
CLIENT: Yeah, while I am happy with the app as it is, there are a couple of things I would change that would make it easier for me to use. Firstly, I think the Google feature is easy to use, but sometimes I find it hard to tell if my calendars aren't syncing because I've been logged out of Google or just because I have no more tasks on Google Calendar. It would also have been nice to modify my alert frequency. Like, if I wanted to be able to get a notification when a task both ends and starts it would be nice to be able to choose this. Finally, the last thing I might add is a way to customise the font and colour scheme of the app so it feels a bit more, like, personal.

NIRVIK: Fair enough. I hadn't considered a few of those ideas. **What other improvements or extensions would you recommend?**
CLIENT: Aside from the things I mentioned previously, I think it would be nice to have a method to add fixed tasks. Like, with me, I have lunch every day at 12:30. With the current system I would have to add this in at the start of every day as tasks can only happen once. However, if there was a way to indicate this is a task that would always occur it would make my task-adding process a lot quicker. This would be a useful update, though it isn't really essential for me.

Analysis of User Feedback

After chatting with my client, I have received both positive and negative feedback about my system. In terms of what the client liked, I am glad to hear that he found the system easy to use and intuitive (line 3). This was one of my main objectives and I spent a considerable amount of time on my design to ensure it was as smooth as possible. I am also very happy to hear that the app met all of his needs and suited his requirements (line 16).

There are however several things which I could certainly work on to improve my client's experience. One of the chief problems my client faced was with the Google Calendar functionality as he was not sure whether the system was syncing his timetable or not (lines 34 – 36). There are a couple of ways I think I could address this problem. One way would be to have a symbol such as a tick at the top of the screen showing that the user is still signed into Google. Alternatively, I could have a small message in the banner saying 'calendar last synced: 2 minutes ago' or something along those lines to allay my client's worries.

Another problem my client faced was with alert frequency and how he wanted the option to choose when notifications appeared (lines 36 – 38). This is a slightly tougher problem to solve as if I give the client unlimited control over when notifications appear in the app they could theoretically add an infinite number of alerts which would crash the system. Therefore, I think it would be more sensible to allow a maximum of 2 alerts per task – one at the start of the task as already implemented, and one at the end like the client is suggesting. I could also perhaps add a buffer to these alerts which the user can specify so they can choose to have, for example, a 5 minute gap between the start task alert and the actual task starting.

Finally, the third biggest problem my client seemed to have faced was with fixed tasks and having to add repeated tasks at the start of each day (lines 44 – 48). This would be a relatively simple problem for me to solve. In future, I would add an additional checkbox to my new task input form asking whether the task is a fixed task or not. If this checkbox were selected, I would simply prevent the task from being deleted at the end of the day and would instead change the start and end date to the next day to ensure it always appears.

Possible Extensions

There are five main things I would do were I to make this system again for my client. The first thing I would do is make the app responsive to all screen sizes rather than tailoring it just for iPhone 8s. I had planned to do this initially. However, due to the excessive amounts of finetuning this would take, I decided against it in the end as I would rather meet my client's specific needs and create a functional program. Next time though, I would allow all IOS devices to access my solution so that my client could install this app on both his iPhone and his iPad. I would also look into researching ways I could share timetables across devices so my client could access his client on multiple different devices throughout the day. This would give my client more flexibility as he could revise anywhere, not just where his phone was.

The second thing I would do is making my Google Calendar integration smoother as my client seemed to have a big problem with it. An important part of my app is in tying together my client's Google Calendar timetable and his revision timetable so it is essential that I allow

my client to seamlessly connect these two calendars. I would follow through with the steps I proposed above. I would first of all make it very clear to the user that they were signed into Google, through a small tick or otherwise in a part of the screen which will always be visible. I would also have a 'last updated' feature to show when Google Calendar was last checked for tasks. I have seen other apps like my email implement this feature to show when it last refreshed my inbox and it allows me to know when I am up to date or not. I feel this feature would be very useful for my app.

The third extension I would consider adding is uploading the app onto the Apple app store. This is again to ensure that my client has as much flexibility as possible. In order to test the app on my client's iPhone I had to plug his iPhone into my laptop so that the files could transfer over. This means that my client cannot download the app without me present, meaning he is constrained to revising with only his iPhone present as that is the only device he has with the app loaded onto it. Were the app uploaded onto the app store, he could easily download the app on any device with an internet connection, decreasing his reliance on me or his phone to revise. I had initially planned this as one of my objectives. However, in order to upload apps onto the app store one has to follow a lengthy process. This includes paying for a $99 Developer License, and waiting several weeks for Apple to approve your app. These factors are why I decided against implementing this feature. However, given more support in the future I could consider uploading it for my client.

Another feature I would add is a way for my client to change the notification frequency of the system. I discussed this in my previous section as well. I had never considered this before as a potential problem. However, after speaking with my client it is clear to me how some users prefer being notified at certain times to know exactly how much time is left. My client for example prefers being notified 5 minutes before the end of a task so that he can try and get as much done as possible before he needs to stop. I could implement this easily using a buffer in between alerts in the alert queue. This would make my system far easier to use for my client, and allow him to revise as effectively as possible by allowing for more user input.

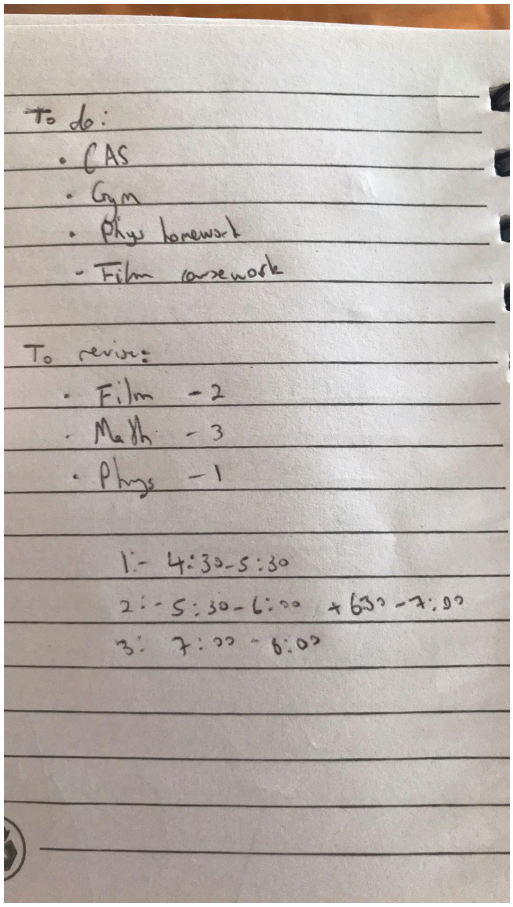
Finally, the last thing I would consider adding onto my system are more graphics and images. Right now, my app has no logo or name – it is simply called 'CS Coursework' when it appears on the home screen as a grey box. This makes my app seem less professional than it is, and while adding images like these don't improve the functionality of my app they improve the user experience which is arguably just as important. The reason I didn't consider giving my app a name or any images is because I wanted to focus solely on the functionality of it. However, next time I would create some custom images and use them in my app to give it a nicer appearance. I would also create some custom images for my navigation bar at the bottom of the screen. Right now, my app uses squares and circles in the navigation bar as these are the default IOS images when no image is specified. Instead, I would use the same images I had specified in my app mock-ups to make it look as clean and professional as possible.

With these five extensions, I believe my app could fulfil my client's requirements even better and make his revision experience more enjoyable. Overall though, I am very happy

with my system and believes my client will greatly benefit from using it as part of his revision process.

Appendix

Appendix 1 – Client's Current System



Example of client's current system

Appendix 2 – Client Interview

Nirvik: So, could you tell me a little bit about your current revision process?

Client: Yeah, so before I start revising I like to make a list of every single topic I need to cover so that it makes my timetables easier to create. I'll usually create my timetable on a day-to-day basis so each morning I'll look at whether I have any important work I need to get done at a certain time. Then I'll just do a certain number of tasks in order of importance depending on how much time I have free. I try spend at least 45 minutes on each task so that's how I gauge how many tasks I can do in a day. Usually, I'll just note down on a piece of paper what I need to do at what time so that I don't forget. This is what I've been doing for a couple years now and usually makes my study leave more productive.

Nirvik: Ok, and how long does this process take? Is it a big part of your day?

Client: Well, mhmmm, it usually takes about 10 minutes every morning to sort out properly. It does take a long time though when I miss something out and need to generate my

timetable again – well, it doesn't really take that long but the time does definitely add up when I could be working instead.

Nirvik: Other than missing tasks out, is there any other part of the process that is time consuming?

Client: Yeah definitely, that's probably the main issue but there are still a couple small problems which are really annoying to deal with. One of those is that I can't really keep track of when I'm supposed to be switching tasks. Basically, the way my revision timetable works is that I do one task for a certain number of minutes before I switch. Since I'm working though, I don't always have a watch with me so I need to keep checking if my time is up yet. This breaks my concentration a bit which is why it isn't brilliant. Sometimes, I'll also repeat a task but only realize once I've spent the time on it. Like, I'll cover one unit of biology in one block, but then the next week I'll accidently cover it again because I've lost my revision timetable from that day.

Nirvik: So, what would you look for in an application to help you solve this?

Client: Well, preferably I'd like to have it on my phone rather than a website or something else because I always have it with me so I can always add another task if necessary. Also –

Nirvik: Sorry, but by your phone do you mean a phone running Android or an iPhone?

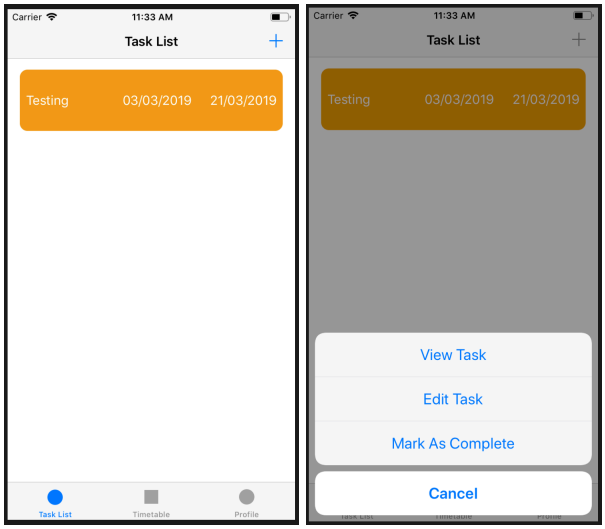
Client: Yeah, I use an iPhone. Anyways continuing on, I want it to be able to sync up with all my school tasks like CAS that I might have scheduled as I sometimes forget about those things in my timetable. It's all on Google Classroom so you should be able to merge it. Also, to solve that problem of me not knowing when to swap tasks it would be cool if it could notify me when I should switch and keep track of what tasks I've done before. The most important thing for me though is that it just creates my timetable for me and adapts if I add another task.

Appendix 3 – Initial Student Survey

Response ID	**How many exams do you have to revise for?**	**How often do you have CCAs during the week?**	**During exam leave, how many hours do you spend revising?**	**What is your main method for tracking revision?**	**What problems do you have with this method?**
1	8	Daily	4	Timetables	Time consuming
2	7	Often (3-4 days)	4	Other (complete one subject at a time)	At end of revision sometimes forget what I covered at start

3	8	Often (3-4 days)	5	Other (go through syllabus)	None
4	10	Rarely (2-3 days)	6.5	Timetables	None
5	6	Often (3-4 days)	5	Timetables	Confusing sometimes as changes everyday
6	4	Rarely (2-3 days)	2	Other (no method)	N/A
7	5	Rarely (2-3 days)	4	Other (follow school timetable)	None
8	9	Never	5	Timetables	Takes time to make
9	7	Daily	5	Timetables	Hard to follow when I have daily CCAs
10	7	Rarely (2-3 days)	4	Other (follow school timetable)	Can't spend more time on worse subjects
11	6	Rarely (2-3 days)	5	Timetables	None
Average	7.0	-	4.5	-	-

Appendix 4 – Task List Page (TaskList.swift)



Left: Task List page. Right: Task List page after pressing a task.

```
1   //  TaskList.swift
2   //  CS Coursework
3   //
4   //  Created by Nirvik Baruah on 3/9/18.
5   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
6   //
7
8   import UIKit
9   import GoogleSignIn
10  import GTMOAuth2
11  import GoogleAPIClientForREST
12  import AudioToolbox
13
14  //TaskList class
15  class TaskList: UITableViewController, GIDSignInUIDelegate {
16
17      //Arrays to manage both types of tasks
18      var tasks = [Task]()
19      var revisionTasks = RevisionPriorityQueue()
20      var user: User?
21
22
23      var inputTask:String!
24      var newTaskAdded:Bool!
25      var startDate: Date!
26      var endDate: Date!
27      var priority: Int!
28      var taskNotes: String!
29      var isRevisionTask: Bool!
30      var wentThroughForm: Bool = false
31      var clickedTaskNotes:String?
32      var clickedTaskTitle:String?
33      var clickedTaskDates:String?
34      var clickedTaskPriority:String?
35
36      var selectedTask: Task?
37
38      //Gets reference to second VC so that can call methods on second table
39      var dailyTimetable : Timetable?
40
41      let userDefaults = UserDefaults.standard
42
43      //Interface to change variables in this class
44      func SetNewTaskDetails(newTask: String!, newStartDate: Date!, newEndDate: Date!,
45  newPriority: Int!, newNotes: String!, isRevisionInp: Bool!){
46          inputTask = newTask
```

```
            startDate = newStartDate
            endDate = newEndDate
            priority = newPriority
            taskNotes = newNotes
            isRevisionTask = isRevisionInp
            wentThroughForm = true
        }

        override func viewDidAppear(_ animated: Bool) {
            if(inputTask != nil && startDate != nil && endDate != nil && priority != nil &&
taskNotes != nil){
                var noDateOverlap: Bool = false
                for task in tasks{
                    //Do not run if revision task
                    if (task is RevisionTask == false && isRevisionTask == false){
                        //Algorithm taken from Ian Nelson on
https://stackoverflow.com/questions/325933/determine-whether-two-date-ranges-overlap
                        if (startDate <= task.GetEndDate() && task.GetStartDate() <= endDate){
                            noDateOverlap = true
                            break
                        }
                    }
                }
                //Alert if date clash
                if(noDateOverlap){
                    let alert = UIAlertController(title: "Alert", message: "You already have a
task during this time period!", preferredStyle: .alert)
                    alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action
-> Void in}))
                    self.present(alert, animated: true, completion: nil)
                }
                //Check if end time is before start time - if so present alert
                else if (endDate < startDate){
                    let alert = UIAlertController(title: "Alert", message: "Task end time must be
after start time!", preferredStyle: .alert)
                    alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action
-> Void in}))
                    self.present(alert, animated: true, completion: nil)
                }
                else{
                    addTask(title: inputTask, startDate: startDate, endDate: endDate, priority:
priority, notes: taskNotes, isRevisionTask: isRevisionTask)
                    dailyTimetable!.updateTable()
                }
            }
            else if wentThroughForm{
                let alert = UIAlertController(title: "Alert", message: "Not all fields
complete!", preferredStyle: .alert)
                alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action ->
Void in}))
                self.present(alert, animated: true, completion: nil)
            }
            wentThroughForm = false

            //Serialise and save data to user defaults
            let encodedData: Data = NSKeyedArchiver.archivedData(withRootObject: tasks)
            userDefaults.set(encodedData, forKey: "tasks")
            userDefaults.synchronize()
        }

        override func viewDidDisappear(_ animated: Bool) {
            inputTask = nil
            newTaskAdded = nil
            startDate = nil
            endDate = nil
            isRevisionTask = nil
            wentThroughForm = false
        }

        override func viewDidLoad() {
            super.viewDidLoad()
            self.tableView.contentInset = UIEdgeInsets.init(top: 10, left: 0, bottom: 0, right:
0);

```

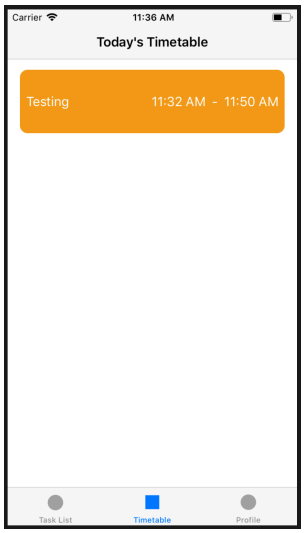
```
        var decoded  = userDefaults.object(forKey: "tasks") as! Data?
        if(decoded != nil){
            let decodedTasks = NSKeyedUnarchiver.unarchiveObject(with: decoded!) as! [Task]
            tasks = decodedTasks

            for element in tasks{
                if (element is RevisionTask){
                    revisionTasks.Push(task: element as! RevisionTask)
                }
            }
        }

        decoded  = userDefaults.object(forKey: "user") as! Data?
        if(decoded != nil){
            let decodedUser = NSKeyedUnarchiver.unarchiveObject(with: decoded!) as! User
            user = decodedUser
        } else{
            //Default values if no user
            let formatter = DateFormatter()
            formatter.dateFormat = "HH:mm"
            let startDate = formatter.date(from: "08:00")
            let endDate = formatter.date(from: "20:00")

            user = User(name: "User", startTime: startDate!, endTime: endDate!, minBreakTime:
10, minTaskTime: 30, maxTaskTime: 120)
        }

        //Gets reference to instance of daily timetable object
        dailyTimetable = tabBarController?.viewControllers?[1].children[0] as? Timetable

        //Create + button in top right
        self.navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .add,
target: self, action: #selector(TaskList.TapAddItem(_:)))

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) ->
Int {
        return tasks.count
    }

    //Format custom cell
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell_task", for: indexPath)
as! CustomTableViewCell
        let newTask = tasks[indexPath.row]
        cell.cellTitle?.text = newTask.GetName()
        let dateFormatter = DateFormatter();
        dateFormatter.dateFormat = "dd/MM/yyyy"
        var strDate = dateFormatter.string(from: newTask.GetEndDate() as Date)
        cell.cellDetail?.text = strDate

        strDate = dateFormatter.string(from: newTask.GetStartDate() as Date)
        cell.cellStartDate?.text = strDate

        if newTask is RevisionTask{
            //Only display end date if revision task
            cell.cellStartDate?.text = ""
            cell.cellHyphen.text = ""
            strDate = dateFormatter.string(from: (newTask as!
RevisionTask).GetFinalDeadline() as Date)
            cell.cellDetail?.text = strDate
            cell.cellBackground.backgroundColor = hexStringToUIColor(hex: "#999999")
        }
        else if (newTask.GetPriority() == 0){
            cell.cellBackground.backgroundColor = hexStringToUIColor(hex: "#58d68d")
        }
        else if (newTask.GetPriority() == 1){
```

```
            cell.cellBackground.backgroundColor = hexStringToUIColor(hex: "#f39c12")
        }
        else if (newTask.GetPriority() == 2){
            cell.cellBackground.backgroundColor = hexStringToUIColor(hex: "#ed532d")
        }
        else{
            cell.cellBackground.backgroundColor = hexStringToUIColor(hex: "#ff0000")
        }
        cell.selectionStyle = .none
        return cell
    }

    func addTask(title: String, startDate: Date, endDate: Date, priority: Int, notes: String,
isRevisionTask: Bool){
        //Check if append normal Task object or a RevisionTask object
        if (isRevisionTask){
            tasks.append(RevisionTask(taskName: title, start: startDate, end: endDate, final:
endDate, taskNotes: notes, isFinished: false, priority: priority))
            RevisionTask.SetUser(newUser: user!)
            //Push to priority queue
            revisionTasks.Push(task: tasks.last as! RevisionTask)
        } else{
            //Push to normal task queue
            tasks.append(Task(taskName: title, startDate: startDate, endDate: endDate,
taskNotes: notes, isFinished: false, priority: priority))
        }
        let indexPath = IndexPath(row: tasks.count - 1, section: 0)
        tableView.insertRows(at: [indexPath], with: .left)
        //Increments static user variable for num tasks done
        user!.IncrementTasksAdded()
    }

    //Perform segue when click + button
    @objc func TapAddItem(_ sender: UIBarButtonItem)
    {
        AudioServicesPlaySystemSound(1306)
        performSegue(withIdentifier: "tapNewItem", sender: sender)
    }
    //Pass current instance of TaskList to AddTaskView class
    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
    {
        if segue.identifier == "tapNewItem"{
            if let navController = segue.destination as? UINavigationController {
                if let childVC = navController.topViewController as? AddTaskViewController {
                    childVC.SetTaskListInstance(obj: self)
                }
            }
        }
        //Set details in view task VC
        else if segue.identifier == "openViewTask"{
            if let navController = segue.destination as? UINavigationController {
                if let childVC = navController.topViewController as? DetailsViewController {
                    if let newTaskNotes = clickedTaskNotes{
                        if let newTaskTitle = clickedTaskTitle{
                            if let newTaskDate = clickedTaskDates{
                                if let newTaskPriority = clickedTaskPriority{
                                    if (newTaskNotes == "Task Notes"){
                                        childVC.SetModalText(newNotes: "None", newTitle:
newTaskTitle, newDate: newTaskDate, newPriority: newTaskPriority)
                                    }
                                    else{
                                        childVC.SetModalText(newNotes: newTaskNotes,
newTitle: newTaskTitle, newDate: newTaskDate, newPriority: newTaskPriority)
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        //Set details in edit task VC
        else if segue.identifier == "editTaskSegue"{
            if let navController = segue.destination as? UINavigationController {
```

```
                if let childVC = navController.topViewController as? EditTaskViewController {
                    childVC.SetTaskInstance(selectedTask: selectedTask, taskListInstance:
self)
                }
            }
        }
    }

    //Getter methods
    func GetNumTasks() -> Int{
        return tasks.count
    }
    func GetTaskAtRow(row: Int) -> Task{
        return tasks[row]
    }

    func GetTasks() -> [Task]{
        return tasks
    }

    func GetUser() -> User?{
        return user
    }

    func GetRevisionTasks() -> RevisionPriorityQueue{
        return revisionTasks
    }

    //For table cell deletion on swipe
    override func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) ->
Bool {
        return true
    }

    //Function to delete row at index path
    func deleteTask(indexPath: IndexPath){
        let deletedTask = tasks[indexPath.row]
        tasks.remove(at: indexPath.row)
        if (deletedTask is RevisionTask){
            revisionTasks.Delete(task: deletedTask as! RevisionTask)
        }

        tableView.deleteRows(at: [indexPath], with: .top)
        let encodedData: Data = NSKeyedArchiver.archivedData(withRootObject: tasks)
        userDefaults.set(encodedData, forKey: "tasks")
        userDefaults.synchronize()

        //Updates other tables in app
        dailyTimetable!.updateTable()
    }

    override func tableView(_ tableView: UITableView, commit editingStyle:
UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
        if indexPath.row < tasks.count
        {
            deleteTask(indexPath: indexPath)

        }
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        //Added to main thread so no delay in presenting UI element
        DispatchQueue.main.async {
            AudioServicesPlaySystemSound(1306)
            //Alert user so can choose what task action to carry out
            let alert = UIAlertController()
            alert.addAction(UIAlertAction(title: "View Task", style: .default, handler: {
action in
                //Set variables so can pass to next VC in prepareforsegue method
                self.clickedTaskNotes = self.tasks[indexPath.row].GetNotes()
                self.clickedTaskTitle = self.tasks[indexPath.row].GetName()
                //Formatter to convert date to string
                let formatter = DateFormatter()
                formatter.dateFormat = "MM/dd HH:mm"
```

```
                self.clickedTaskDates = formatter.string(from:
self.tasks[indexPath.row].GetStartDate()) + " - " + formatter.string(from:
self.tasks[indexPath.row].GetEndDate())
                //Convert priority to text
                if (self.tasks[indexPath.row].GetPriority() == 0){
                    self.clickedTaskPriority = "Low"
                } else if (self.tasks[indexPath.row].GetPriority() == 1){
                    self.clickedTaskPriority = "Medium"
                } else if (self.tasks[indexPath.row].GetPriority() == 2){
                    self.clickedTaskPriority = "High"
                } else{
                    self.clickedTaskPriority = "Google Calendar Task"
                }

                self.performSegue(withIdentifier: "openViewTask",sender: self)
            }))
            alert.addAction(UIAlertAction(title: "Edit Task", style: .default, handler: {
action in
                self.selectedTask = self.tasks[indexPath.row]
                self.performSegue(withIdentifier: "editTaskSegue",sender: self)}))
            alert.addAction(UIAlertAction(title: "Mark As Complete", style: .default,
handler: { action in
                    self.user!.IncrementTasksDone()
                    self.deleteTask(indexPath: indexPath)
            }))
            alert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: { action
-> Void in}))
            self.present(alert, animated: true, completion: nil)
        }
    }
}
```

Appendix 5 – Daily Timetable Page (Timetable.swift)



Daily Timetable page

```
1   //
2   //  Timetable.swift
3   //  CS Coursework
4   //
5   //  Created by Nirvik Baruah on 3/9/18.
6   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
7   //
8
9   import UIKit
10  import UserNotifications
11  import GoogleAPIClientForREST
12  import GTMOAuth2
13  import GoogleSignIn
14
15  //Daily Timetable class
16  class Timetable: UITableViewController{
17
18      //Private variables
19      var tasks = [Task]()
20      //Gets instance of TaskList to access task list
21      var taskListReference: TaskList?
22      var revisionTasks: RevisionPriorityQueue?
23
24      var user: User?
25
26      let userDefaults = UserDefaults.standard
27
28      //Taken from https://stackoverflow.com/questions/50227276/obtaining-google-calendar-in-
29  swift
30      /// Creates calendar service with current authentication
31      fileprivate lazy var calendarService: GTLRCalendarService? = {
32          let service = GTLRCalendarService()
33          // Have the service object set tickets to fetch consecutive pages
34          // of the feed so we do not need to manually fetch them
35          service.shouldFetchNextPages = true
36          // Have the service object set tickets to retry temporary error conditions
37          // automatically
38          service.isRetryEnabled = true
39          service.maxRetryInterval = 15
40
41          guard let currentUser = GIDSignIn.sharedInstance().currentUser,
42              let authentication = currentUser.authentication else {
43                  return nil
44          }
45
46          service.authorizer = authentication.fetcherAuthorizer()
```

```
        return service
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        //Get google calendar events first
        getEvents(calendarId: "primary")
        // Do any additional setup after loading the view, typically from a nib.
        self.tableView.contentInset = UIEdgeInsets.init(top: 10, left: 0, bottom: 0, right:
0);

        updateTable()
    }

    override func viewDidAppear(_ animated: Bool) {
        updateTable()
    }


    //Keeps tasks array consistent across VCs
    //Refreshes tableview once array data changed
    func updateTable(){
        taskListReference = tabBarController?.viewControllers?[0].children[0] as? TaskList
        tasks = taskListReference!.GetTasks()
        user = taskListReference!.GetUser()
        revisionTasks = taskListReference!.GetRevisionTasks()

        //Filters elements which are due today so only they appear and removes all revision
tasks so fresh
        tasks = tasks.filter { Calendar.current.isDateInToday($0.GetEndDate()) }
        tasks = tasks.sorted(by: { $0.GetStartDate() < $1.GetStartDate() })
        tasks.removeAll(where: {$0 is RevisionTask})

        //Only run if revision tasks exist
        if (revisionTasks!.GetCount() > 0){
        //Since my client only wants to do revision tasks in an uninterrupted block of time
which is bigger than 30 minutes but smaller than 2 hours, I will iterate through my tasks for
the day, find when a block like this exists, and pop a revision task

            //Create a copy of revisionTasks array so do not lose original tasks when popping
            //Use copy from NSCopy protocol to allow copy by value
            var taskListCopy = revisionTasks!.copy() as! RevisionPriorityQueue

            //Checks that not all tasks are revision tasks
            if (tasks.filter({$0 is RevisionTask}).count < tasks.count){
                //Creates copy of normal tasks to easily access times between tasks
                var normalTasksCopy = [Task]()
                for element in tasks{
                    normalTasksCopy.append(element.copy() as! Task)
                }

                //For slotting between start of day and first task
                var timeAvailable = findAvailableTime(startDate: user!.GetStartTime(),
endDate: normalTasksCopy[0].GetStartDate())
                if (timeAvailable >= user!.GetMinTaskTime()){
                    taskListCopy = slotTasks(timeAvailable: timeAvailable, startTime:
user!.GetStartTime(), taskListCopy: taskListCopy)
                }
                //For tasks in the middle of the day
                if (normalTasksCopy.count > 1){
                    for i in 0...normalTasksCopy.count - 2{
                        timeAvailable = findAvailableTime(startDate:
normalTasksCopy[i].GetEndDate(), endDate: normalTasksCopy[i+1].GetStartDate())
                        if (timeAvailable >= user!.GetMinTaskTime()){
                            taskListCopy = slotTasks(timeAvailable: timeAvailable, startTime:
normalTasksCopy[i].GetEndDate(), taskListCopy: taskListCopy)
                        }
                    }
                }
                //For slotting between last task and end of day
                timeAvailable = findAvailableTime(startDate:
normalTasksCopy.last!.GetEndDate(), endDate: user!.GetEndTime())
                if (timeAvailable >= user!.GetMinTaskTime()){
```

```
                    taskListCopy = slotTasks(timeAvailable: timeAvailable, startTime:
normalTasksCopy.last!.GetEndDate(), taskListCopy: taskListCopy)
                }
            }
            //Full day is for revision
            else{
                let timeAvailable = findAvailableTime(startDate: user!.GetStartTime(),
endDate: user!.GetEndTime())
                tasks.removeAll()
                taskListCopy = slotTasks(timeAvailable: timeAvailable, startTime:
user!.GetStartTime(), taskListCopy: taskListCopy)
            }
        }
        scheduleNotifications()
        self.tableView.reloadData()
    }

    //Function to slot tasks in a given time period
    //Algorithm:
    //1. Set task Start Time
    //2. Add timePerTask to start time and set this date as end time
    //3. Check if still within users day
    //4. If so, start time = end time
    //5. Pop task
    //6. Loop for the smallest of either numRevTasksToDo or number of revision tasks
    func slotTasks(timeAvailable: Int, startTime: Date, taskListCopy: RevisionPriorityQueue)
-> RevisionPriorityQueue{
        let minTaskTime = user!.GetMinTaskTime()
        let maxTaskTime = user!.GetMaxTaskTime()
        let breakTime = user!.GetMinBreakTime() * 2
        var numRevTasksToDo = taskListCopy.GetCount()
        if (numRevTasksToDo > 0 && timeAvailable >= minTaskTime + breakTime){
            var timePerTask = timeAvailable / numRevTasksToDo
            while ((timePerTask > maxTaskTime + breakTime || timePerTask < minTaskTime +
breakTime) && numRevTasksToDo > 1){
                numRevTasksToDo -= 1
                timePerTask = timeAvailable / numRevTasksToDo
            }
            //Check if successful in dividing time - otherwise just allocate max/min time per
task
            if (timePerTask > maxTaskTime + breakTime){
                timePerTask = min(timeAvailable, maxTaskTime + breakTime)
                numRevTasksToDo = taskListCopy.GetCount()
            }
            else if (timePerTask < minTaskTime + breakTime){
                timePerTask = max(timeAvailable, minTaskTime + breakTime)
                numRevTasksToDo = 1
            }
            //Add half the break time in minutes at start for break
            var taskStartTime = combineDateWithTime(date: Date(), time:
startTime.addingTimeInterval(Double((breakTime/2) * 60)))
            //Take away half the break time as half the break time used at start and end for
break
            var taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask -
(breakTime)) * 60))
            for _ in 1...numRevTasksToDo{
                taskListCopy.Peek()!.SetStartDate(date: taskStartTime!)
                taskListCopy.Peek()!.SetEndDate(date: taskEndTime)
                tasks.append(taskListCopy.Pop())
                //Add 5 minutes between tasks as break
                //Take away 5 minutes at end as total time per task includes break
                //Multiply by 60 to convert minutes to seconds
                taskStartTime = taskEndTime.addingTimeInterval(Double((breakTime/2) * 60))
                taskEndTime = taskStartTime!.addingTimeInterval(Double((timePerTask -
(breakTime)) * 60))
            }
        }
        //Must sort after every slotting as can add any number of tasks so not a case of
simple swapping of elements
        tasks = tasks.sorted(by: { $0.GetStartDate() < $1.GetStartDate() })
        return taskListCopy
    }

    //Finds how many minutes free between two dates
```

```
    func findAvailableTime(startDate: Date, endDate: Date) -> Int{
        var time: Int
        let calendar = Calendar.current
        let startTime = calendar.dateComponents([.hour, .minute], from: startDate)
        let endTime = calendar.dateComponents([.hour, .minute], from: endDate)
        time = max(0, calendar.dateComponents([.minute], from: startTime, to:
endTime).minute!)
        return time
    }

    //Format custom cell
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell_task", for: indexPath)
as! CustomTableViewCell
        let newTask = tasks[indexPath.row]
        cell.dailyListTitle?.text = newTask.GetName()
        let dateFormatter = DateFormatter();
        dateFormatter.dateFormat = "h:mm a"
        var strDate = dateFormatter.string(from: newTask.GetEndDate() as Date)
        cell.dailyListDetail?.text = strDate

        strDate = dateFormatter.string(from: newTask.GetStartDate() as Date)
        cell.dailyListStartDate?.text = strDate

        if newTask is RevisionTask{
            cell.dailyListBackground.backgroundColor = hexStringToUIColor(hex: "#999999")
        }
        else if (newTask.GetPriority() == 0){
            cell.dailyListBackground.backgroundColor = hexStringToUIColor(hex: "#58d68d")
        }
        else if (newTask.GetPriority() == 1){
            cell.dailyListBackground.backgroundColor = hexStringToUIColor(hex: "#f39c12")
        }
        else if (newTask.GetPriority() == 2){
            cell.dailyListBackground.backgroundColor = hexStringToUIColor(hex: "#ed532d")
        }
        else{
            cell.dailyListBackground.backgroundColor = hexStringToUIColor(hex: "#ff0000")
        }

        cell.selectionStyle = .none
        return cell
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) ->
Int {
        return tasks.count
    }

    //Function to combine date with time
    //Taken from https://gist.github.com/justinmfischer/0a6edf711569854c2537
    func combineDateWithTime(date: Date, time: Date) -> Date?{
        let calendar = Calendar.current

        let dateComponents = calendar.dateComponents([.year, .month, .day], from: date)
        let timeComponents = calendar.dateComponents([.hour, .minute, .second], from: time)


        var mergedComponments = DateComponents()
        mergedComponments.year = dateComponents.year
        mergedComponments.month = dateComponents.month
        mergedComponments.day = dateComponents.day
        mergedComponments.hour = timeComponents.hour
        mergedComponments.minute = timeComponents.minute
        mergedComponments.second = timeComponents.second

        return calendar.date(from: mergedComponments)
    }
```

```
268
269         //Schedules notifications depending on start time
270         @objc func scheduleNotifications() {
271             let center = UNUserNotificationCenter.current()
272             //Remove all previous requests
273             center.removeAllPendingNotificationRequests()
274
275             for task in tasks{
276                 let content = UNMutableNotificationContent()
277                 content.title = task.GetName()
278                 content.body = "New task started!"
279                 content.categoryIdentifier = "alarm"
280                 content.sound = UNNotificationSound.default
281
282                 var dateComponents = DateComponents()
283                 dateComponents.hour = Calendar.current.component(.hour, from:
284     task.GetStartDate())
285                 dateComponents.minute = Calendar.current.component(.minute, from:
286     task.GetStartDate())
287                 let trigger = UNCalendarNotificationTrigger(dateMatching: dateComponents,
288     repeats: false)
289
290                 let request = UNNotificationRequest(identifier: UUID().uuidString, content:
291     content, trigger: trigger)
292                 center.add(request)
293             }
294         }
295
296         //Get events from calendar
297         func getEvents(calendarId: String) {
298             guard let service = self.calendarService else {
299                 return
300             }
301
302             //Only get events from now until a week later
303             let startDateTime = GTLRDateTime(date: Calendar.current.startOfDay(for: Date()))
304             let endDateTime = GTLRDateTime(date: Date().addingTimeInterval(60*60*24*7))
305             let eventsListQuery = GTLRCalendarQuery_EventsList.query(withCalendarId: calendarId)
306             eventsListQuery.timeMin = startDateTime
307             eventsListQuery.timeMax = endDateTime
308
309             _ = service.executeQuery(eventsListQuery, completionHandler: { (ticket, result,
310     error) in
311                 guard error == nil, let items = (result as? GTLRCalendar_Events)?.items else {
312                     return
313                 }
314
315                //Parsing JSON from request
316                 for item in items{
317                     var isNew = true
318                     let taskName = item.summary
319                     let startTime = item.start!.dateTime!.date
320                     let endTime = item.end!.dateTime!.date
321                     let priority = 3
322                     for prevItems in self.taskListReference!.GetTasks(){
323                         if prevItems.GetName() == taskName{
324                             isNew = false
325                             break
326                         }
327                     }
328                     if isNew{
329                         self.taskListReference?.addTask(title: taskName!, startDate: startTime,
330     endDate: endTime, priority: priority, notes: "Task Notes", isRevisionTask: false)
331                     }
332                 }
333
334                 if items.count > 0{
335                     let alert = UIAlertController(title: "Alert", message: "Google Calendar tasks
336     synced with task list!", preferredStyle: .alert)
337                     alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action
338     -> Void in}))
339                     self.present(alert, animated: true, completion: nil)
340                 }
341             })
```

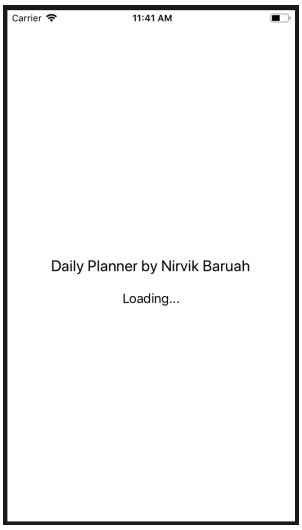
```
342         }
343     }
```

Appendix 6 – Task Class (Task.swift)

```
1   //
2   //  Task.swift
3   //  CS Coursework
4   //
5   //  Created by Nirvik Baruah on 7/9/18.
6   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
7   //
8
9   import Foundation
10
11  //Needs to conform to NSCoding in order to be saved to local UserDefaults
12  class Task: NSObject, NSCoding, NSCopying{
13      //Private variables initialized
14      private var taskName : String
15      private var endDate : Date
16      private var startDate : Date
17      //Optional value
18      private var taskNotes : String?
19      private var isFinished : Bool
20      //Highest priority is most important
21      private var priority : Int
22
23      //Constructor - set object values
24      init(taskName: String, startDate: Date, endDate : Date, taskNotes : String, isFinished :
25  Bool, priority : Int) {
26          self.taskName = taskName
27          self.startDate = startDate
28          self.endDate = endDate
29          self.taskNotes = taskNotes
30          self.isFinished = isFinished
31          self.priority = priority
32      }
33
34      required convenience init(coder aDecoder: NSCoder) {
35          let priority = aDecoder.decodeInteger(forKey: "priority")
36          let taskName = aDecoder.decodeObject(forKey: "taskName") as! String
37          let endDate = aDecoder.decodeObject(forKey: "endDate") as! Date
38          let startDate = aDecoder.decodeObject(forKey: "startDate") as! Date
39          let taskNotes = aDecoder.decodeObject(forKey: "taskNotes") as! String
40          let isFinished = aDecoder.decodeBool(forKey: "isFinished")
41          self.init(taskName: taskName, startDate: startDate, endDate: endDate, taskNotes:
42  taskNotes, isFinished: isFinished, priority: priority)
43      }
44
45      func encode(with aCoder: NSCoder) {
46          aCoder.encode(taskName, forKey: "taskName")
47          aCoder.encode(priority, forKey: "priority")
48          aCoder.encode(startDate, forKey: "startDate")
49          aCoder.encode(endDate, forKey: "endDate")
50          aCoder.encode(taskNotes, forKey: "taskNotes")
51          aCoder.encode(isFinished, forKey: "isFinished")
52      }
53
54      /*
55       Overloaded constructor for Google Classroom tasks
56      convenience init(test : String){
57          self.init(...)
58      }
59       */
60
61      //Getter methods
62      func GetName() -> String{
63          return self.taskName
64      }
65      func GetEndDate() -> Date{
66          return self.endDate
67      }
68      func GetStartDate() -> Date{
69          return self.startDate
70      }
```

```
71          func GetNotes() -> String{
72              //Unwrapping optional
73              return self.taskNotes!
74          }
75          func GetIsFinished() -> Bool{
76              return self.isFinished
77          }
78          func GetPriority() -> Int{
79              return self.priority
80          }
81
82          //Setter methods
83          func SetName(name : String) -> Void{
84              self.taskName = name
85          }
86          func SetEndDate(date : Date) -> Void{
87              self.endDate = date
88          }
89          func SetStartDate(date : Date) -> Void{
90              self.startDate = date
91          }
92          func SetNotes(notes : String) -> Void{
93              self.taskNotes = notes
94          }
95          func SetIsFinished() -> Void{
96              self.isFinished = !self.isFinished
97          }
98          func SetPriority(enteredPriority : Int) -> Void{
99              self.priority = enteredPriority
100         }
101
102         //Conform to NSCopying protocol to allow copy by value
103         func copy(with zone: NSZone? = nil) -> Any {
104             let copy = Task(taskName: GetName(), startDate: GetStartDate(), endDate:
105 GetEndDate(), taskNotes: GetNotes(), isFinished: GetIsFinished(), priority: GetPriority())
106             return copy
107         }
108 }
```

Appendix 7 – Application Delegate Class (AppDelegate.swift)



App loading screen which displays while AppDelegate.swift runs setup procedures.

```
1   //
2   //  AppDelegate.swift
3   //  CS Coursework
4   //
5   //  Created by Nirvik Baruah on 3/9/18.
6   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
7   //
8
9   import UIKit
10  import UserNotifications
11  import GoogleSignIn
12
13  @UIApplicationMain
14  class AppDelegate: UIResponder, UIApplicationDelegate, GIDSignInDelegate{
15
16      var window: UIWindow?
17
18
19      func application(_ application: UIApplication, didFinishLaunchingWithOptions
20  launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
21          registerForPushNotifications()
22
23          GIDSignIn.sharedInstance().clientID = "750611619941-
24  1s590l62jchq9g8o2if1sp6c3oufvttg.apps.googleusercontent.com"
25          GIDSignIn.sharedInstance().delegate = self
26
27          return true
28      }
29
30      func applicationWillResignActive(_ application: UIApplication) {
31          // Sent when the application is about to move from active to inactive state. This can
32  occur for certain types of temporary interruptions (such as an incoming phone call or SMS
33  message) or when the user quits the application and it begins the transition to the
34  background state.
35          // Use this method to pause ongoing tasks, disable timers, and invalidate graphics
36  rendering callbacks. Games should use this method to pause the game.
37      }
38
39      func applicationDidEnterBackground(_ application: UIApplication) {
40          // Use this method to release shared resources, save user data, invalidate timers,
41  and store enough application state information to restore your application to its current
42  state in case it is terminated later.
```

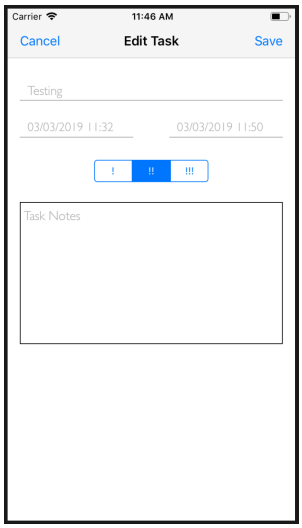
```
        // If your application supports background execution, this method is called instead
of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here you
can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was
inactive. If the application was previously in the background, optionally refresh the user
interface.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See
also applicationDidEnterBackground:.
    }

    //Get approval to send notifications
    func registerForPushNotifications() {
        UNUserNotificationCenter.current().delegate = self as?
UNUserNotificationCenterDelegate
        UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound,
.badge]) {
            (granted, error) in
            print("Permission granted: \(granted)")
            // 1. Check if permission granted
            guard granted else { return }
            // 2. Attempt registration for remote notifications on the main thread
            DispatchQueue.main.async {
                UIApplication.shared.registerForRemoteNotifications()
            }
        }
    }

    //Google Sign in
    func application(_ app: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        return GIDSignIn.sharedInstance().handle(url as URL?,
                                                 sourceApplication:
options[UIApplication.OpenURLOptionsKey.sourceApplication] as? String,
                                                 annotation:
options[UIApplication.OpenURLOptionsKey.annotation])
    }

    private func application(application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) ->
Bool {
        // Initialize sign-in
        GIDSignIn.sharedInstance().clientID = "750611619941-
1s590l62jchq9g8o2if1sp6c3oufvttg.apps.googleusercontent.com"
        GIDSignIn.sharedInstance().delegate = self as? GIDSignInDelegate

        return true
    }

    func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!,
              withError error: Error!) {
        if let error = error {
            print("\(error.localizedDescription)")
        }
    }

    func sign(_ signIn: GIDSignIn!, didDisconnectWith user: GIDGoogleUser!,
              withError error: Error!) {
        // Perform any operations when the user disconnects from app here.
        // ...
    }

}
```

Appendix 8 – Revision Task Class (RevisionTask.swift)

```swift
//
//  RevisionTask.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 26/12/18.
//  Copyright © 2018 Nirvik Baruah. All rights reserved.
//

import Foundation

//Subclass of Task
class RevisionTask: Task{
    //Static variable for all revision tasks
    static var user: User?

    //Final deadline is when task must be done by e.g when exam is
    //end date set in initialiser is to process when the task should be done
    var finalDeadline: Date?
    convenience init(taskName: String, start: Date, end : Date, final: Date, taskNotes : String, isFinished : Bool, priority : Int) {
        //Call superclass constructor
        self.init(taskName: taskName, startDate: start, endDate: end, taskNotes: taskNotes, isFinished: isFinished, priority: priority)
        self.finalDeadline = final
    }

    required convenience init(coder aDecoder: NSCoder) {
        let priority = aDecoder.decodeInteger(forKey: "priority")
        let taskName = aDecoder.decodeObject(forKey: "taskName") as! String
        let endDate = aDecoder.decodeObject(forKey: "endDate") as! Date
        let startDate = aDecoder.decodeObject(forKey: "startDate") as! Date
        let taskNotes = aDecoder.decodeObject(forKey: "taskNotes") as! String
        let isFinished = aDecoder.decodeBool(forKey: "isFinished")
        let finalDeadline = aDecoder.decodeObject(forKey: "finalDeadline") as! Date
        self.init(taskName: taskName, start: startDate, end: endDate, final: finalDeadline, taskNotes: taskNotes, isFinished: isFinished, priority: priority)
    }

    override func encode(with aCoder: NSCoder) {
        aCoder.encode(self.GetName(), forKey: "taskName")
        aCoder.encode(self.GetPriority(), forKey: "priority")
        aCoder.encode(self.GetStartDate(), forKey: "startDate")
        aCoder.encode(self.GetEndDate(), forKey: "endDate")
        aCoder.encode(self.GetNotes(), forKey: "taskNotes")
        aCoder.encode(self.GetIsFinished(), forKey: "isFinished")
        aCoder.encode(self.GetFinalDeadline(), forKey: "finalDeadline")
    }

    //Conform to NSCopying protocol
    override func copy(with zone: NSZone? = nil) -> Any {
        let copy = RevisionTask(taskName: GetName(), startDate: GetStartDate(), endDate: GetEndDate(), taskNotes: GetNotes(), isFinished: GetIsFinished(), priority: GetPriority())
        return copy
    }

    //Make class-level setter in order to set a class-level variable
    static func SetUser(newUser: User) -> Void{
        user = newUser
    }
    static func GetUser() -> User?{
        return user
    }

    func GetFinalDeadline() -> Date{
        return finalDeadline!
    }

    func SetFinalDeadline(newDate: Date){
        finalDeadline = newDate
```

```
70          }
71
72          //Other Getters and Setters defined in super class so no need to redefine
73      }
74
```

Appendix 9 – Edit Task Page (EditTaskViewController.swift)

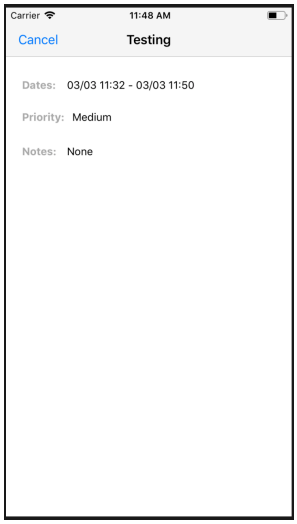


Edit Task page

```
//
//  EditTaskViewController.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 8/1/19.
//  Copyright © 2019 Nirvik Baruah. All rights reserved.
//

import UIKit

class EditTaskViewController: UIViewController {

    //Private variables
    var task: Task?
    var taskList: TaskList?
    var wasOriginallyRevisionTask: Bool = false
    let formatter = DateFormatter()

    //Inputs from page
    @IBOutlet weak var taskNameInp: TextField!
    @IBOutlet weak var startTimeInp: TextField!
    @IBOutlet weak var endTimeInp: TextField!
    @IBOutlet weak var priorityInp: UISegmentedControl!
    @IBOutlet weak var notesInp: TextView!

    @IBAction func saveTask(_ sender: Any) {
        formatter.dateFormat = "dd/MM/yyyy HH:mm"
        var completeForm: Bool = true
        var alert: UIAlertController
        //Only potential error source is date is wrong format so only need to check that
        print(startTimeInp.text!)
        if (formatter.date(from: startTimeInp.text!) != nil){
            task!.SetStartDate(date: formatter.date(from: startTimeInp.text!)!)
        }
        else {
            //Invalid format
            completeForm = false
            alert = UIAlertController(title: "Alert", message: "Start date is in wrong
format!", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action ->
Void in}))
            self.present(alert, animated: true, completion: nil)
        }

```

```
            if (formatter.date(from: endTimeInp.text!) != nil){
                task!.SetEndDate(date: formatter.date(from: endTimeInp.text!)!)
            }
            else {
                //Invalid format
                completeForm = false
                alert = UIAlertController(title: "Alert", message: "End date is in wrong
format!", preferredStyle: .alert)
                alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action ->
Void in}))
                self.present(alert, animated: true, completion: nil)
            }

            if (completeForm){
                task!.SetName(name: taskNameInp.text!)
                task!.SetPriority(enteredPriority: priorityInp.selectedSegmentIndex)
                task!.SetNotes(notes: notesInp.text!)
                taskList!.tableView.reloadData()
                self.dismiss(animated: true, completion: nil)
            }
        }

        override func viewDidLoad() {
            super.viewDidLoad()
        }

        override func didReceiveMemoryWarning() {
            super.didReceiveMemoryWarning()
            // Dispose of any resources that can be recreated.
        }

        //Sets details of input form
        func SetDetails(){
            taskNameInp.text = task!.GetName()
            formatter.dateFormat = "dd/MM/yyyy HH:mm"
            startTimeInp.text = formatter.string(from: task!.GetStartDate())
            endTimeInp.text = formatter.string(from: task!.GetEndDate())
            notesInp.text = task!.GetNotes()
            priorityInp.selectedSegmentIndex = task!.GetPriority()

        }

        //Pass data on once form complete
        func SetTaskInstance(selectedTask: Task?, taskListInstance: TaskList){
            _ = self.view
            task = selectedTask
            taskList = taskListInstance
            wasOriginallyRevisionTask = false
            SetDetails()
        }

    }
```

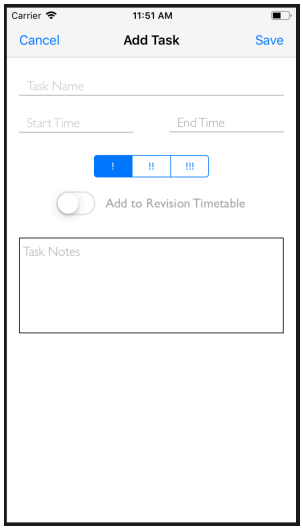
Appendix 10 – View Task Page (DetailsViewController.swift)



View details for a task.

```
1   //
2   //  DetailsViewController.swift
3   //  CS Coursework
4   //
5   //  Created by Nirvik Baruah on 15/11/18.
6   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
7   //
8
9   import UIKit
10
11  class DetailsViewController: UIViewController {
12
13      //Output references
14      @IBOutlet weak var taskTitle: UINavigationItem!
15      @IBOutlet weak var taskNotes: UITextView!
16      @IBOutlet weak var taskDate: UITextView!
17      @IBOutlet weak var taskPriority: UITextView!
18
19      //Change view controller outputs
20      func SetModalText(newNotes: String, newTitle: String, newDate: String, newPriority:
21  String){
22          //Force view to refresh so that it is added to hierarchy and IBOutlets are set
23          let _  = self.view
24          taskNotes.text = newNotes
25          taskTitle.title = newTitle
26          taskDate.text = newDate
27          taskPriority.text = newPriority
28      }
29
30      override func viewDidLoad() {
31          super.viewDidLoad()
32          // Do any additional setup after loading the view, typically from a nib.
33      }
34
35      override func didReceiveMemoryWarning() {
36          super.didReceiveMemoryWarning()
37          // Dispose of any resources that can be recreated.
38      }
39
40  }
```

Appendix 11 – New Task Page (AddTaskViewController.swift)



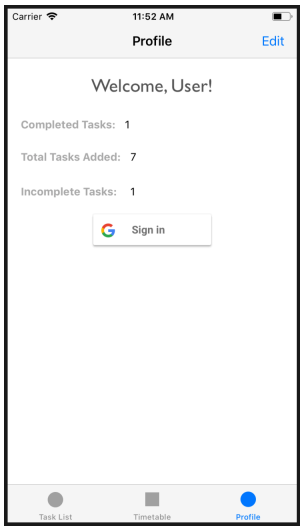
New task input form.

```
1   //
2   //  AddTaskViewController.swift
3   //  CS Coursework
4   //
5   //  Created by Nirvik Baruah on 21/9/18.
6   //  Copyright © 2018 Nirvik Baruah. All rights reserved.
7   //
8
9   import UIKit
10  import AudioToolbox
11
12  class AddTaskViewController: UIViewController {
13
14      //Private variables
15      var taskList : TaskList?
16      var formComplete: Bool = true
17
18      //Inputs
19      @IBOutlet var AddTaskController: UIView!
20      @IBOutlet weak var taskTitleInp: TextField!
21      @IBOutlet weak var priorityInp: UISegmentedControl!
22
23      @IBOutlet weak var taskNotesInp: UITextView!
24      @IBOutlet weak var startDateInp: TextField!
25      private var startDate: Date?
26      private var startDatePicker: UIDatePicker?
27      @IBOutlet weak var endDateInp: TextField!
28      private var endDate: Date?
29      private var endDatePicker: UIDatePicker?
30
31      @IBOutlet weak var revisionToggleInp: UISwitch!
32
33      @IBAction func revisionTaskToggled(_ sender: Any) {
34          if revisionToggleInp.isOn{
35              //Disable start date input if revision task selected
36              startDateInp.isEnabled = false
37              startDateInp.allowsEditingTextAttributes = false
38              startDateInp.backgroundColor = UIColor.lightGray
39              startDateInp.textColor = UIColor.white
40              startDateInp.text = "N/A"
41          }
42          else{
43              startDateInp.isEnabled = true
```

```
            startDateInp.allowsEditingTextAttributes = true
            startDateInp.backgroundColor = nil
            startDateInp.textColor = UIColor.lightGray
            startDateInp.text = "Start Date"
        }
    }
    //Set instance of TaskList
    func SetTaskListInstance(obj: TaskList){
        taskList = obj
    }

    @IBAction func SaveClick(_ sender: Any) {
        AudioServicesPlaySystemSound(1306)
        if formComplete{
            performSegue(withIdentifier: "ReturnTaskList", sender: sender)
        }
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        //Creates DatePicker which will open when textField pressed
        startDatePicker = UIDatePicker()
        startDatePicker?.datePickerMode = .dateAndTime
        //Adds done button to DatePicker
        let startToolbar = UIToolbar().ToolbarPiker(mySelect:
#selector(AddTaskViewController.startDateChange(gestureRecognizer:)))
        startDateInp.inputAccessoryView = startToolbar
        //Dismiss if click viewcontroller
        let startTapGesture = UITapGestureRecognizer(target: self, action:
#selector(AddTaskViewController.startDateChange(gestureRecognizer:)))
        //Bind tapGesture to current view
        view.addGestureRecognizer(startTapGesture)
        //Binds datePicker to textField
        startDateInp.inputView = startDatePicker

        //Creates DatePicker which will open when textField pressed
        endDatePicker = UIDatePicker()
        endDatePicker?.datePickerMode = .dateAndTime
        //Adds done button to DatePicker
        let endToolbar = UIToolbar().ToolbarPiker(mySelect:
#selector(AddTaskViewController.endDateChange(gestureRecognizer:)))
        endDateInp.inputAccessoryView = endToolbar
        //Dismiss if click viewcontroller
        let endTapGesture = UITapGestureRecognizer(target: self, action:
#selector(AddTaskViewController.endDateChange(gestureRecognizer:)))
        //Bind tapGesture to current view
        view.addGestureRecognizer(endTapGesture)
        //Binds datePicker to textField
        endDateInp.inputView = endDatePicker
    }

    @objc func startDateChange(gestureRecognizer: UITapGestureRecognizer){
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "dd/MM/yyyy"
        startDateInp.text = dateFormatter.string(from: startDatePicker!.date)
        startDate = startDatePicker!.date
        view.endEditing(true)
    }

    @objc func endDateChange(gestureRecognizer: UITapGestureRecognizer){
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "dd/MM/yyyy"
        endDateInp.text = dateFormatter.string(from: endDatePicker!.date)
        endDate = endDatePicker!.date
        view.endEditing(true)
    }

    //Pass data back to task list view controller
    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
    {
        if segue.identifier == "ReturnTaskList"
        {
            //Check state of user input toggle switch
            if (revisionToggleInp.isOn){
```

```
                taskList?.SetNewTaskDetails(newTask: taskTitleInp.text, newStartDate: Date(),
newEndDate: endDate, newPriority: priorityInp.selectedSegmentIndex, newNotes:
taskNotesInp.text, isRevisionInp: true)
            } else{
                taskList?.SetNewTaskDetails(newTask: taskTitleInp.text, newStartDate:
startDate, newEndDate: endDate, newPriority: priorityInp.selectedSegmentIndex, newNotes:
taskNotesInp.text, isRevisionInp: false)
            }
        }
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}
//From https://stackoverflow.com/questions/28048469/add-a-done-button-within-a-pop-up-
datepickerview-in-swift
extension UIToolbar {

    func ToolbarPiker(mySelect : Selector) -> UIToolbar {

        let toolBar = UIToolbar()

        toolBar.barStyle = UIBarStyle.default
        toolBar.isTranslucent = true
        toolBar.tintColor = UIColor.black
        toolBar.sizeToFit()

        let doneButton = UIBarButtonItem(title: "Done", style: UIBarButtonItem.Style.plain,
target: self, action: mySelect)
        let spaceButton = UIBarButtonItem(barButtonSystemItem:
UIBarButtonItem.SystemItem.flexibleSpace, target: nil, action: nil)

        toolBar.setItems([ spaceButton, doneButton], animated: false)
        toolBar.isUserInteractionEnabled = true

        return toolBar
    }

}
```

Appendix 12 – Profile Page (ProfilePage.swift)

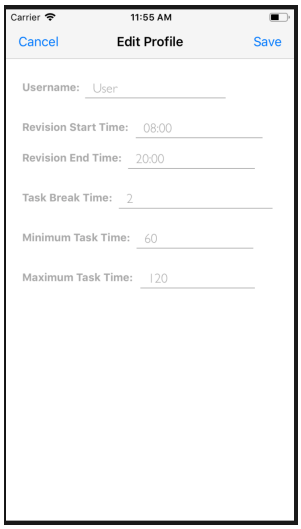


User profile page

```
//
//  ProfilePage.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 3/9/18.
//  Copyright © 2018 Nirvik Baruah. All rights reserved.
//

import UIKit
import GoogleAPIClientForREST
import GTMOAuth2
import GoogleSignIn
import AudioToolbox

class ProfilePage: UIViewController, GIDSignInUIDelegate {

    @IBOutlet weak var googleSigninButton: GIDSignInButton!

    var taskListReference: TaskList?
    var user: User?
    let userDefaults = UserDefaults.standard
    @IBOutlet weak var welcomeGreetingLabel: UILabel!
    @IBOutlet weak var completedTasksLabel: UITextView!
    @IBOutlet weak var totalTasksLabel: UITextView!
    @IBOutlet weak var incompleteTasksLabel: UITextView!

    func initGoogle() {
        // Initialize sign-in
        GIDSignIn.sharedInstance().clientID = "750611619941-
1s590l62jchq9g8o2if1sp6c3oufvttg.apps.googleusercontent.com"
        GIDSignIn.sharedInstance().scopes = ["https://www.googleapis.com/auth/calendar"]
        GIDSignIn.sharedInstance().uiDelegate = self
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        initGoogle()
        GIDSignIn.sharedInstance().signIn()
        //Create + button in top right
        self.navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .edit,
target: self, action: #selector(ProfilePage.TapEdit(_:)))
        loadLatestUser()
    }
```

```
45
46      override func viewDidAppear(_ animated: Bool) {
47          loadLatestUser()
48          updateValues()
49
50      }
51
52      func loadLatestUser(){
53          taskListReference = tabBarController?.viewControllers?[0].children[0] as? TaskList
54          user = taskListReference!.GetUser()
55      }
56
57      func updateValues(){
58          //If logged in, set profile name to google name
59          let currentUser = GIDSignIn.sharedInstance().currentUser
60          if (currentUser != nil){
61              user!.SetUserName(name: currentUser!.profile.name)
62              googleSigninButton.isHidden = true
63          } else{
64              googleSigninButton.isHidden = false
65          }
66
67          welcomeGreetingLabel.text = "Welcome, " + user!.GetUserName() + "!"
68          completedTasksLabel.text = String(user!.GetNumTasksDone())
69          totalTasksLabel.text = String(user!.GetNumTasksAdded())
70          //Can't have negative number so use a max statement
71          incompleteTasksLabel.text = String(taskListReference!.GetTasks().count)
72
73          //Write User attributes to memory so not lost
74          let encodedData: Data = NSKeyedArchiver.archivedData(withRootObject: user!)
75          userDefaults.set(encodedData, forKey: "user")
76          userDefaults.synchronize()
77      }
78
79      override func didReceiveMemoryWarning() {
80          super.didReceiveMemoryWarning()
81          // Dispose of any resources that can be recreated.
82      }
83
84      //Perform segue when click + button
85      @objc func TapEdit(_ sender: UIBarButtonItem)
86      {
87          AudioServicesPlaySystemSound(1306)
88          performSegue(withIdentifier: "editProfileSegue", sender: sender)
89      }
90
91      override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
92
93          if segue.identifier == "editProfileSegue"{
94              if let navController = segue.destination as? UINavigationController {
95                  if let childVC = navController.topViewController as?
96  EditProfileViewController {
97                      childVC.SetDetails(userInstance: user!)
98                  }
99              }
100         }
101     }
102
103 }
```

Appendix 13 – Edit Profile Page (EditProfileViewController.swift)



Edit profile page

```
//
//  AddTaskViewController.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 21/9/18.
//  Copyright © 2018 Nirvik Baruah. All rights reserved.
//

import UIKit
import AudioToolbox

class EditProfileViewController: UIViewController {
    var user: User?

    @IBOutlet weak var nameInp: TextField!
    @IBOutlet weak var revisionStartTimeInp: TextField!
    @IBOutlet weak var revisionEndTimeInp: TextField!
    @IBOutlet weak var taskBreakTimeInp: TextField!
    @IBOutlet weak var minTaskTimeInp: TextField!
    @IBOutlet weak var maxTaskTimeInp: TextField!

    //Event once save button clicked
    @IBAction func saveClick(_ sender: Any) {
        var completeForm: Bool = true
        let formatter = DateFormatter()
        var alert: UIAlertController
        formatter.dateFormat = "HH:mm"
        AudioServicesPlaySystemSound(1306)
        //Only potential error source is with date so only need to check that
        if (formatter.date(from: revisionStartTimeInp.text!) != nil){
            user!.SetStartTime(time: formatter.date(from: revisionStartTimeInp.text!)!)
        }
        else{
            completeForm = false
            alert = UIAlertController(title: "Alert", message: "Start time is in wrong
format!", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action ->
Void in}))
            self.present(alert, animated: true, completion: nil)
        }

        if (formatter.date(from: revisionEndTimeInp.text!) != nil){
            user!.SetEndTime(time: formatter.date(from: revisionEndTimeInp.text!)!)
        }
        else{
```

```
            completeForm = false
            alert = UIAlertController(title: "Alert", message: "End time is in wrong
format!", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: {action ->
Void in}))
            self.present(alert, animated: true, completion: nil)
        }

        if completeForm{
            user!.SetUserName(name: nameInp.text!)
            user!.SetMinBreakTime(time: Int(taskBreakTimeInp.text!)!)
            user!.SetMinTaskTime(time: Int(minTaskTimeInp.text!)!)
            user!.SetMaxTaskTime(time: Int(maxTaskTimeInp.text!)!)
            self.dismiss(animated: true, completion: nil)
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    func SetDetails(userInstance: User){
        //Load view hierarchy first
        _ = self.view
        user = userInstance
        //Sets placeholder text on inputs
        nameInp.text = user!.GetUserName()
        let dateFormatter = DateFormatter();
        dateFormatter.dateFormat = "HH:mm"

        revisionStartTimeInp.text = dateFormatter.string(from: user!.GetStartTime())
        revisionEndTimeInp.text = dateFormatter.string(from: user!.GetEndTime())
        taskBreakTimeInp.text = String(user!.GetMinBreakTime())
        minTaskTimeInp.text = String(user!.GetMinTaskTime())
        maxTaskTimeInp.text = String(user!.GetMaxTaskTime())
    }
}
```

Appendix 14 – User Class (User.swift)

```swift
//
//  User.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 19/12/18.
//  Copyright © 2018 Nirvik Baruah. All rights reserved.
//

import Foundation

//Needs to conform to NSCoding in order to be saved to local UserDefaults
//User class to store user details
class User: NSObject, NSCoding{
    //Private variables initialized
    //Static variable as should be the same regardless of user
    private static var numTasksDone: Int = 0
    private static var totalTasksAdded: Int = 0
    private var userName : String
    private var startTime: Date
    private var endTime: Date
    private var minBreakTime: Int
    private var minTaskTime: Int
    private var maxTaskTime: Int

    //Constructor - set object values
    init(name: String, startTime: Date, endTime: Date, minBreakTime: Int, minTaskTime: Int, maxTaskTime: Int) {
        self.userName = name
        self.startTime = startTime
        self.endTime = endTime
        self.minBreakTime = User.RoundUp(number: minBreakTime)
        self.minTaskTime = minTaskTime
        self.maxTaskTime = maxTaskTime
    }

    //Overloaded constructor for User Defaults loading
    init(name: String, startTime: Date, endTime: Date, minBreakTime: Int, minTaskTime: Int, maxTaskTime: Int, numTasksDone: Int, totalTasksAdded: Int){
        self.userName = name
        self.startTime = startTime
        self.endTime = endTime
        self.minBreakTime = User.RoundUp(number: minBreakTime)
        self.minTaskTime = minTaskTime
        self.maxTaskTime = maxTaskTime
        User.numTasksDone = numTasksDone
        User.totalTasksAdded = totalTasksAdded
    }

    required convenience init(coder aDecoder: NSCoder) {
        let numTasksDone = aDecoder.decodeInteger(forKey: "numTasksDone")
        let totalTasksAdded = aDecoder.decodeInteger(forKey: "totalTasksAdded")
        let userName = aDecoder.decodeObject(forKey: "userName") as! String
        let startTime = aDecoder.decodeObject(forKey: "startTime") as! Date
        let endTime = aDecoder.decodeObject(forKey: "endTime") as! Date
        let minBreakTime = aDecoder.decodeInteger(forKey: "minBreakTime")
        let minTaskTime = aDecoder.decodeInteger(forKey: "minTaskTime")
        let maxTaskTime = aDecoder.decodeInteger(forKey: "maxTaskTime")
        self.init(name: userName, startTime: startTime, endTime: endTime, minBreakTime: minBreakTime, minTaskTime: minTaskTime, maxTaskTime: maxTaskTime, numTasksDone: numTasksDone, totalTasksAdded: totalTasksAdded)
    }

    func encode(with aCoder: NSCoder) {
        aCoder.encode(User.numTasksDone, forKey: "numTasksDone")
        aCoder.encode(User.totalTasksAdded, forKey: "totalTasksAdded")
        aCoder.encode(userName, forKey: "userName")
        aCoder.encode(startTime, forKey: "startTime")
        aCoder.encode(endTime, forKey: "endTime")
        aCoder.encode(minBreakTime, forKey: "minBreakTime")
```

```
            aCoder.encode(minTaskTime, forKey: "minTaskTime")
            aCoder.encode(maxTaskTime, forKey: "maxTaskTime")
        }

        //Getter methods
        func GetNumTasksDone() -> Int{
            return User.numTasksDone
        }
        func GetUserName() -> String{
            return self.userName
        }
        func GetStartTime() -> Date{
            return self.startTime
        }
        func GetEndTime() -> Date{
            return self.endTime
        }
        func GetMinBreakTime() -> Int{
            return self.minBreakTime
        }
        func GetMinTaskTime() -> Int{
            return self.minTaskTime
        }
        func GetMaxTaskTime() -> Int{
            return self.maxTaskTime
        }
        func GetNumTasksAdded() -> Int{
            return User.totalTasksAdded
        }

        //Setter methods
        func IncrementTasksDone() -> Void{
            User.numTasksDone += 1
        }
        func SetUserName(name: String) -> Void{
            self.userName = name
        }
        func SetStartTime(time: Date) -> Void{
            self.startTime = time
        }
        func SetEndTime(time: Date) -> Void{
            self.endTime = time
        }
        func SetMinBreakTime(time: Int) -> Void{
            //Must be a multiple of 2 as my system is only accurate to the nearest minute
            self.minBreakTime = User.RoundUp(number: time)
        }
        func SetMinTaskTime(time: Int) -> Void{
            self.minTaskTime = time
        }
        func SetMaxTaskTime(time: Int) -> Void{
            self.maxTaskTime = time
        }
        func IncrementTasksAdded(){
            User.totalTasksAdded += 1
        }

        //Function for rounding up to nearest even number
        private static func RoundUp(number: Int) -> Int{
            if (number % 2 == 1){
                return number + 1
            }
            return number
        }
    }
```

Appendix 15 – Task Priority Queue Class (RevisionPriorityQueue.swift)

```swift
//
//  RevisionPriorityQueue.swift
//  CS Coursework
//
//  Created by Nirvik Baruah on 28/12/18.
//  Copyright © 2018 Nirvik Baruah. All rights reserved.
//

//Priority queue to represent RevisionTasks

import Foundation

//Needs to conform to NS protocols to enable copying by value
class RevisionPriorityQueue: NSObject, NSCopying{
    //Array representing queue. No need for a priority array as RevisionTask
    //object contains priority attribute
    var queue = [RevisionTask]()

    //Represents next free slot. Also represents array size
    var tailPointer: Int

    //Polymorphism of constructors
    override init() {
        self.tailPointer = 0
    }

    init(queue: [RevisionTask], pointer: Int){
        self.queue = queue
        self.tailPointer = pointer
    }

    func GetCount() -> Int{
        return self.tailPointer
    }

    //Pop
    func Pop() -> RevisionTask{
        self.tailPointer -= 1
        let firstElement: RevisionTask = self.queue[0]
        self.queue.remove(at: 0)
        return firstElement
    }

    //Peek
    func Peek() -> RevisionTask?{
        return self.queue.first
    }

    //Push
    func Push(task: RevisionTask){
        self.queue.append(task)
        self.tailPointer += 1
        Sort()
    }

    //Delete
    func Delete(task: RevisionTask){
        //Removes revision task element from queue
        self.queue = self.queue.filter {$0 != task}
        self.tailPointer -= 1
    }

    //Sort queue by priority first then by final deadline
    func Sort(){

        self.queue.sort { (elem1, elem2) -> Bool in
            //Sort bu priority first
            if elem1.GetPriority() > elem2.GetPriority() {
                return true
            }
```

```
            // Sort bu deadline next
            if elem1.GetPriority() == elem2.GetPriority() {
                return elem1.GetFinalDeadline() < elem2.GetFinalDeadline()
            }
            return false
        }
    }

    func GetQueue() -> [RevisionTask]{
        return queue
    }

    //Conform to NSCopying protocol to allow copying by value
    func copy(with zone: NSZone? = nil) -> Any {
        let copy = RevisionPriorityQueue(queue: GetQueue(), pointer: GetCount())
        return copy
```