Pollett, Luke - BetterHSCTimer Documentation

1.1 Problem definition

As an HSC student, exams are undoubtedly one of, if not, the most stressful aspects of our schooling and it is throughout these nerve-wracking times where students need all the time they can get in their tests. During my year 11 end-of-year exams, I noticed that the timing system my school used was both inefficient, and often confusing to students like me who have to sacrifice precious seconds interpreting the abomination on the projector. The setup, featuring an analogue clock and a separate notepad for exam details, often caused confusion, and required unnecessary amounts of setup time, including painfully waiting for that second hand to tick over to finally put pen to paper.

My application aims to simplify this process for both students and invigilators. It will provide clear exam details including start and finish times and total time for specific NSW HSC subjects chosen for evaluation, streamlining the setup process. However, it will not accommodate interstate or state-specific courses due to time and resource constraints as a part of this project.

During development, potential challenges such as automating details and subject selections for new examinations may arise. However, the focus will remain on creating a user-friendly interface that includes analogue and digital clocks, automatic exam timers, and dropdown menus for subject selection. This integrated approach aims to reduce setup time and improve clarity for both students and invigilators in times of great stress like the HSC.

1.2 Legal and ethical considerations

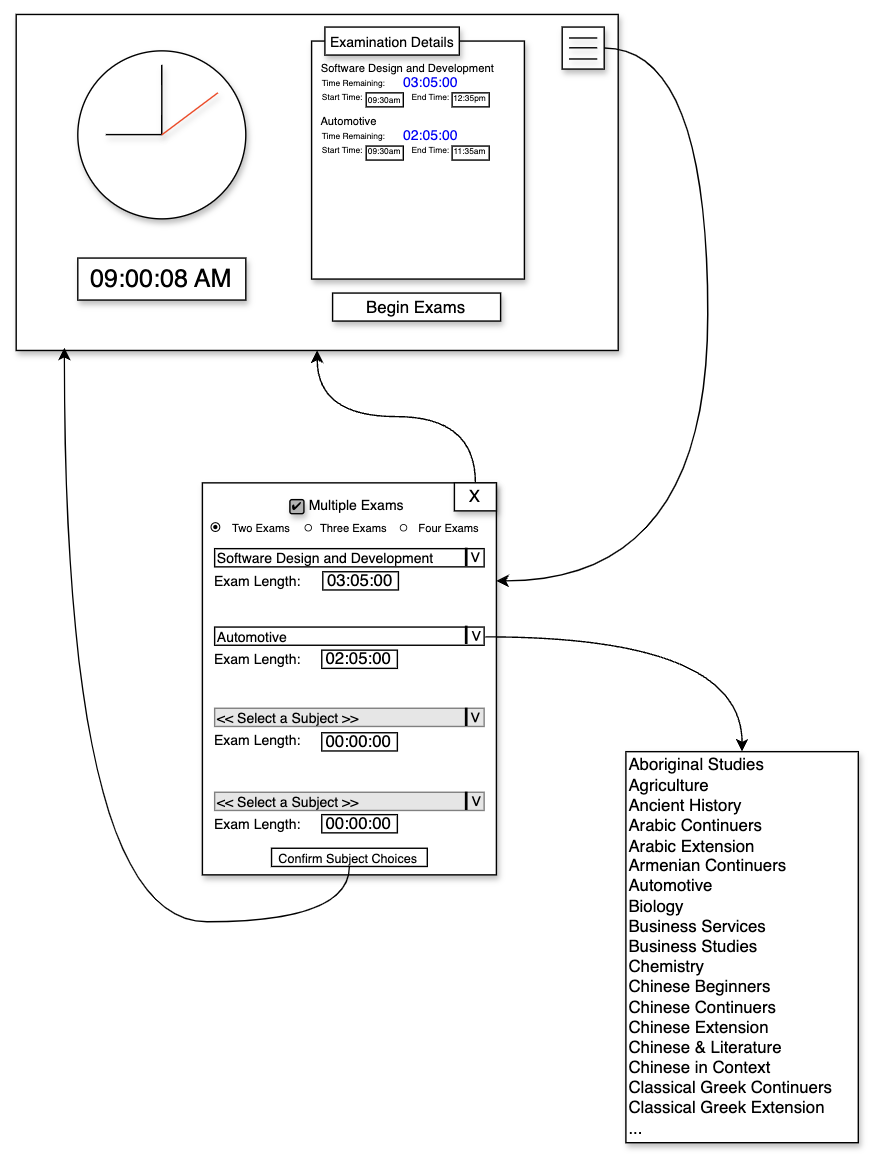
1. As this is an application used for timing and managing HSC examinations, the functionality of my application is required to exhibit high levels of reliability and consistency. If this application proves otherwise, it has the potential to severely impact students in several ways. For example, technical glitches during exams such as the desynchronisation of the analogue and digital clock could disrupt the testing process, causing stress and anxiety among students who rely on the app for allocating time to individual questions.
2. When considering distributing my application to schools across NSW, it is crucial to consider the possible licensing options I could adopt. Open-source licenses, despite accommodating source code flexibility and the benefit of transparency towards the community, can be subject to security vulnerabilities such as individuals with malicious intent and copyright concerns. Alternatively, Closed-source licenses such as a commercial license provide sources for financial compensation as well as ongoing development support, however they lack the benefit of transparency as the source code is restricted to the public. This lack of transparency means that other developers cannot access the source code, potentially limiting opportunities to further develop the application to benefit Students and Invigilators alike.
3. My application must also comply with the educational regulations outlined by bodies such as NESA. This compliance is imperative for the adoption and integration of this application in schools throughout NSW, as without the alignment of this project with the regulations potential sources for bias and unequal evaluation amongst students begin to rise.

1.3 Functional and non-functional requirements

|  |  |
| --- | --- |
| Functional Requirements | Details |
| Local Time Displays | App should Display the user’s local time in the form ‘HH:MM:SS AM/PM’ as well as an analogue clock |
| Exam Countdown Timer | App should Clearly illustrate a countdown timer for Each exam the user has selected, it will update every second and change text colour depending on time remaining. |
| Display HSC Subjects | App should Clearly Communicate a range of HSC subjects that the user can observe and choose from. These subjects will be visible in dropdowns or ‘OptionMenus’ in the settings window |
| Subject Selection | App should Enables user to choose specific HSC subjects from the dropdown menus in the settings window. The chosen subjects will be processed and placed in a clear, organised display on the main window along with their respective exam lengths (‘HH:MM:SS’). |
| Start & End Time Displays | App should Communicate a Start and End Time for each examination so invigilators can keep track of the time allocation and better manage other activities throughout the length of the exam(s). |

|  |  |
| --- | --- |
| Non-Functional Requirements | Details |
| Format User’s local time | App should get the user’s device local time and convert it to the format ‘HH:MM:SS AM/PM’. Calculate placement of hour, minute and second hands on an analogue clock face using mathematical methods such as trigonometry, the unit circle and on-screen coordinates. |
| Update Exam Timer(s) | App should get the length of the exam(s) in seconds as an integer, and convert it to the format ‘HH:MM:SS’ using python operators such as modulo (%). Labels on the main window will be configured to display the exam lengths. This process of calculating time remaining and configuring labels is run every 1 second until all exams have concluded. |
| Get & Display HSC Subjects | App should Retrieve all HSC subjects from a provided text file filled with subjects from NESA, and convert it into an array of subjects that can be referenced with indexes. These subjects are then used as the data range/’options’ for the exam dropdowns. |
| Process User’s Subjects | App should process user’s selected subjects from the dropdown menus and retrieve the specific exam(s) working and reading times from the provided text files. The application should then calculate the total length of the exam and format it into ‘HH:MM:SS’, and update the main window’s examination display. |
| Calculate Start & End Time | App should get the user’s current local time when starting the exams and calculate the specific time in which the exam(s) will end, by adding the length (‘HH:MM:SS’) to the initial starting time. |

2.1 Storyboard



2.2 Context Diagram

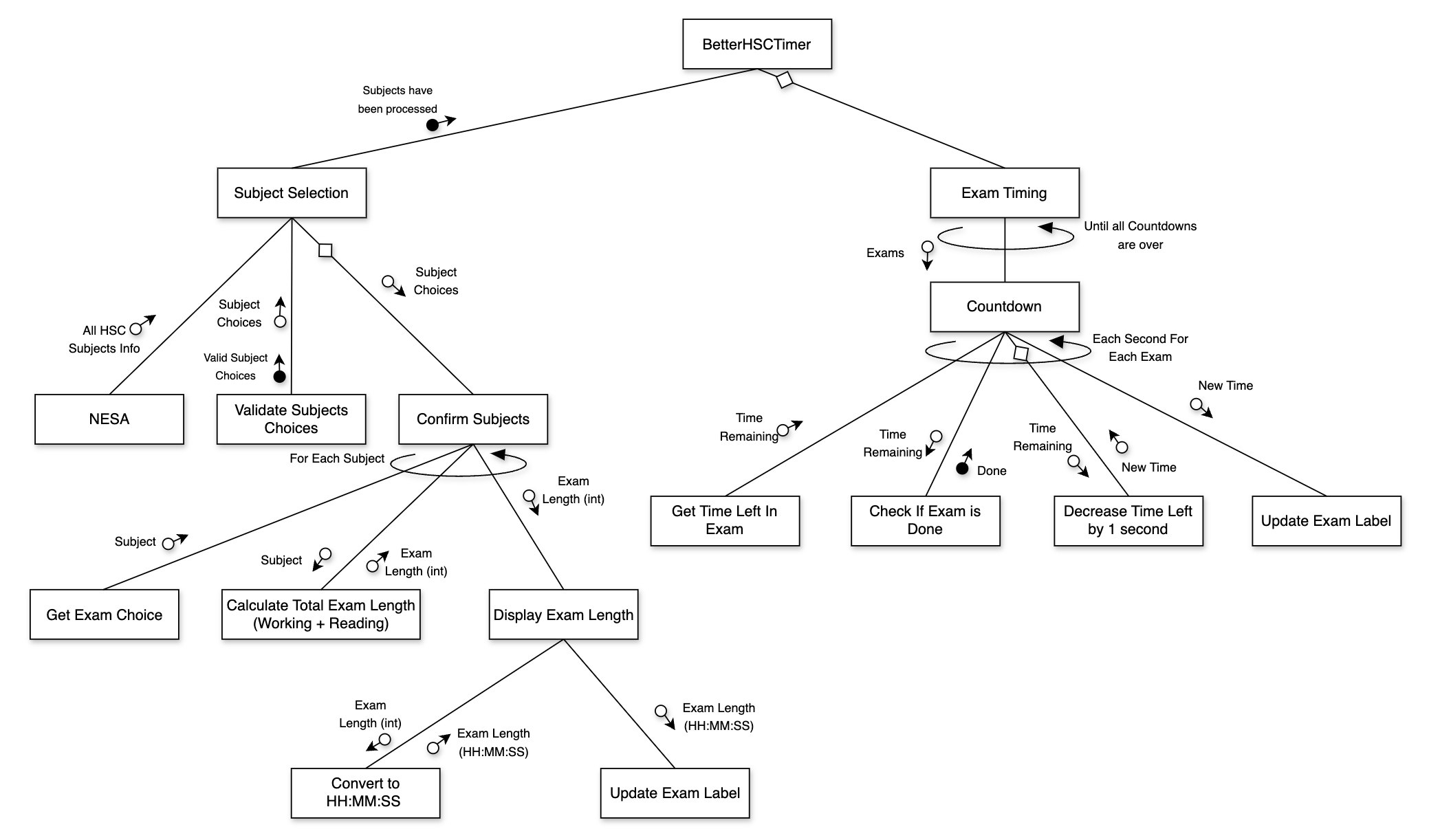
A diagram of a student's exam

Description automatically generated

2.3 Data Flow Diagram

A diagram of a process

Description automatically generated2.4 Structure Chart



A diagram of a computer program

Description automatically generated2.5 Algorithms

Algorithmic description examples:

**Get All HSC Subjects + Exam Details Function:**

1. Retrieve HSC Subjects from NESA as well as their respective exam working and reading times and put them in individual text files.
2. Read the individual text files add the subjects, exam working times and reading times into individual lists.

- (each subject and exam working/reading times share the same index across the lists)

**Get User Subjects Function:**

1. Retrieve number of exams chosen by user
2. Loop through all chosen exams and calculate their total exam length in seconds
3. Add Each exam’s total length to a list.
4. Format each exam’s total exam length in the format: “HH:MM:SS”
5. Configure Each exam’s timer label with the exam’s length in the required format

**Countdown Exam(s) Function:**

1. Retrieve all exams and their lengths.
2. Loop through every exam and check if the exam is over.
3. If not, decrease the total time left on the exam by 1 second.
4. Check if Timer is over again, and if this is the case, update the Exam’s Label to “Time’s Up!”
5. Repeat above loop every second until all exams have concluded.

2.6 GANTT Charts

Project Outset GANTT Chart:A graph with blue and black lines

Description automatically generated with medium confidence

During-Development GANTT Chart:A graph with orange and black lines

Description automatically generated with medium confidence

During this project, my goal was to spend less time refining the subject selection process, and more time on the Examination Timer itself, however as the GANTT chart would show it was this process that inhibited further project development as a whole. During the development stages of this project, I often found myself caught up in the refining of minute portions of code to their utmost degree, rather than dedicating the required time and effort into the core components that made up my application. This was clearly evident in my project’s ‘During-Development’ GANTT chart, where it is visible that major contributions began to pile up towards the back end of the time constraint.

A key takeaway from this project management process that I have learned, is that I should focus less on perfecting everything before moving on, and instead balance the workload throughout development to meet my desired criteria/deadlines.

3.1 Development Log

|  |  |
| --- | --- |
| Development Log Entry 1 | |
| Date | 15h November 2023 |
| Week Number | Term 4 Week 6 |
| Summary of Work Done | Commencing the beginning of the Major Project:   * Created basic tkinter window that holds widgets such as canvas, frame, button, etc. * PROJECT IDEA: Pathfinding algorithm that can navigate through a given area with obstacles. Going to use mathematics in the algorithm such as Pythagoras’ Theorem. * Potential Ideas for Problem Statement: Shape it around students who get lost when coming to St Augs for the first time. Make the pathfinding space 3D to simulate buildings?? |
| Challenges and Solutions | N/A |
| Milestones Achieved | Program Main Window up and running, Project development begins |
| Screenshots | A screenshot of a computer  Description automatically generated |

|  |  |
| --- | --- |
| Development Log Entry 2 | |
| Date | 24th November 2023 |
| Week Number | Term 4 Week 7 |
| Summary of Work Done | Significant Changes from Dev Log 1:   * CHANGE OF IDEA: After talking to Mr. Fong about my idea, we concluded that it would be best if I did something else as this idea was actually very hard: Mr. Fong suggested an exam-timing software because the St Aug method was inefficient and disorganised. * Got a local digital clock ticking in main window: * Implemented OOP for my application as a whole; the app is a class with functions such as the one that updates the digital clock in the main window * Marked down what I plan to do with each widget in the main window |
| Challenges and Solutions | Challenge: Digital clock wasn’t ticking in main window  Solution: It was just getting the current time and putting it on a label without updating at all. I solved this by adding a def function update\_time() which just gets the users current time and adds it to the label then updates that label every second. |
| Milestones Achieved | New Idea: exam timing app  Got a working time clock in the main window |
| Screenshots |  |

|  |  |
| --- | --- |
| Development Log Entry 3 | |
| Date | 23rd December 2023 |
| Week Number | School Holidays (**Week 6** of Development) |
| Summary of Work Done | A lot of progress, but also a lot of really bad problems:   * I have started to make the analogue clock for the main window! So far it’s looking very promising except for some pretty bad errors: * The clock hands update every second and appear to be ticking as a normal clock would, however after a little while the process that updates the second hand falls behind, and it’s not deleting the previous hand quick enough for the new one to be placed cleanly. Not sure how they’re going out of sync because the values are hard coded?? * The program crashes after a little while and I have no idea why. |
| Challenges and Solutions | Challenges: Method of updating analogue clock hands isn’t working; hands start ‘doubling-up’ after some time and hands aren’t getting deleted quick enough.  Solutions: Not yet fixed :( |
| Milestones Achieved | Very close to getting a working analogue clock in main window |
| Screenshots | Normal clock: After ~ 2 min:  A screenshot of a clock  Description automatically generated  Code: calculate where to place the second hand and replace it after 1 second (function is the same for minute and hour hands except for the part of local\_time)  A computer screen shot of a program code  Description automatically generated |

|  |  |
| --- | --- |
| Development Log Entry 4 | |
| Date | 28th December 2023 |
| Week Number | School Holidays (**Week 7** of Development) |
| Summary of Work Done | A LOT of progress from last Dev Log:   * Used AI (CodePilot) to help me try to understand what was wrong with my clock and how I could potentially fix it. This AI code was pretty similar to mine which didn’t work so I scrapped it. * I Solved the error with the previous edition of the analogue clock, I think it was the python time library I was using (ctime()) so I swapped it for time.strftime('%I:%M:%S %p') which works now. * I used more AI to help me draw the clock face and hands. It suggested drawing using trigonometry and screen coordinates which I’m familiar with, so I tried that and it worked.   Summary: Solved previous clock hands issue, Used AI to help make analogue clock and now it works as intended, planning for dropdown menu/sidebar, Made visual rep. of what I want final product to look like. |
| Challenges and Solutions | Challenges: Program was becoming slow because of the previous analogue clock method, Inefficient clock drawing/updating, ctime() too slow  Solutions: Used AI assistance to find a better way to draw and update the clock hands, used trigonometry to draw canvas shapes (clock face + hands), swapped ctime() for time.strftime() |
| Milestones Achieved | Both Time displays are done now, organised a lot of my code, made app look generally better, added a menu button so I can start putting in the exam timing part of my app. |
| Screenshots | A screenshot of a computer screen  Description automatically generated    A clock with a black and white dial  Description automatically generated |

|  |  |
| --- | --- |
| Development Log Entry 5 | |
| Date | 27th June 2024 |
| Week Number | Term 2 Week 9 |
| Summary of Work Done | Major gap in development between last Development Log and April. Development resumed around May 2024.  Significant changes, The Program is now complete and ready for publication:   * Got all HSC Subjects and their exam working + reading times from NESA and added them to individual text files to be used by dropdown menus. * Created Exam selection process in the settings window. Involves several dropdown menus and radio buttons where the user can choose how many exams they want and specific subjects. Process gets the length of the exams chosen by the user and displays them on both the settings and main window. * Researched ways to implement a countdown timer and adapted samples of internet code to facilitate multiple countdown timers simultaneously. * In addition to exam length, Examination start and end times are calculated and displayed to the user when the ‘Begin Exams’ countdown button is pressed. * Ensured Widgets are placed consistently across devices with varying screen resolutions. * Final tweaks and fixes to analogue clock face. * Got rid of the application class as it was causing complications with variables within the code. * Used control structures and other aspects such as OOP to efficiently and concisely place widgets that would become repetitive if hard coded. |
| Challenges and Solutions | Describe specific problems encountered.  Detail how the problems were addressed or solved. |
| Milestones Achieved | Note any significant progress points or project phases completed. |
| Screenshots | A screenshot of a clock  Description automatically generatedA screenshot of a computer  Description automatically generated |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Category | Test Case Description | Input to Provide | Expected Output | Actual Output | Pass/Fail | Action Taken |
| Test 1 | Path coverage | Test that all examinations stop and get removed after ‘Clear’ button is pressed | Button ‘Clear’ pressed | All countdown timers stop and get removed from main window | All countdown timers stop and get removed from main window | Pass | N/A |
| Test 2 | Boundary Value | Verify countdown timer with 0 time remaining/duration | Time remaining in exam == 0 | Timer label = “Time’s Up!” | Timer label = “Time’s Up!” | Pass | N/A |
| Test 3 | Path coverage | Test that countdown timers start after pressing ‘Begin Exams’ | Button ‘Begin Exams’ pressed | Countdown timers start counting down | Countdown timers start counting down | Pass | N/A |
| Test 4 | Abnormal Data | Check if the exam length of a subject is in correct format | ’03:05:111’ from text file | Error pop-up stating “Please Enter A Valid Exam Time In The Format ‘HH:MM:SS’” | Error pop-up stating “Please Enter A Valid Exam Time In The Format ‘HH:MM:SS’” | Pass | N/A |
| Test 5 | Abnormal Data | Check If exams will discontinue if the time remaining is set to “Stopped” | “Stopped” in place of the time remaining for an examination | All examinations will stop | TypeError: unsupported operand type(s) for -=: 'str' and 'int' | Fail | Implement Try, Except function on countdown timer to prevent any invalid operations. |
| Test 6 | Abnormal Data | Check that the analogue clock remains consistent on screen across devices with varying resolutions | Resolution of Device: ‘1920 x 1080’ | Analogue clock face remains consistent on main window | Clock face has deviated left from original position | Fail | Create a ‘clock canvas’ and frame to keep all components of the clock within a confined space. This space is then placed at the clock face’s expected location on the main window. |
| Test 7 | Boundary Value | Validating the option for the maximum amount of exams | ‘Four Exams’ Radio button selected | All Four Subject dropdown menus become available | All Four Dropdowns become active | Pass | N/A |
| Test 8 | Abnormal Data | Not selecting one of the exams from the dropdown(s) | “<< Select a Subject >>” | Error pop-up stating “Please select at least one valid subject” | ValueError: '<< Select a Subject >>' is not in list | Fail | Added an initial check after user submits their chosen subjects where if the subject is == “<< Select a Subject >>” display an error popup |
| Test 9 | Path coverage | Validate exams with minimal subjects chosen | ‘Biology’ | Display the selected exam’s name and length on the settings and main window | Labels with Exam name and length on both windows | Pass | N/A |
| Test 10 | Missing data | Not selecting all required subjects when undergoing multiple exams | ‘Biology’, ‘Chemistry’, ‘<< Select a Subject >>’ | Error pop-up stating, “Please Ensure All Subjects Are Selected”. | Error pop-up stating, “Please Ensure All Subjects Are Selected”. | Pass | N/A |

5.1 GitHub Repository

GitHub Repository URL: <https://github.com/LukePollett/SDD-Major-Project-2024.git>

6.1 Project Reflection

My project, BetterHSCTimer, is an exam invigilation software that assists in the management and timing of large-scale examinations. My project strived to significantly reduce the time and effort required to set up, manage and time examinations, whilst also maintaining an aesthetically pleasing and understandable layouts toward students sitting the exams. My application also aimed to accommodate strictly NSW HSC subjects as interstate studies would occupy too much of the allocated time and resources.

Several aspects of this project were successful and various objectives were achieved such as the clean and efficient local time displays, the subject selection process involving the user being able to select any HSC subject, and countdown timing features. These objectives were essential to the functionality of the application and hence their completion marked large milestones in development. In addition, the implementation of the majority of these objectives exhibited linear development patterns whereas others required separate methods like such as backtracking through repository commits.

There was a wide range of difficulty throughout the development process that saw the hindering of implementation. There were higher levels of difficulty in sections relating to subject selection, as seen in the development GANTT chart, that did not go in accordance with the project outset’s time management strategies, involving aspects such as the implementation of multiple exam choices. Most often, the logic surrounding the subject selection surprised me, as in my project outset I failed to even recognise these possible complications as an inhibiting factor to the project’s progression.

In addition, there were more drawbacks to the flow of development including aspects such as the countdown timers, where I experienced complications in updating the timer labels with the correct value. In many cases this update process saw timers decrease in 2’s (seconds) per second, not counting down at all, and exponential decay (timer decreases by 1, 2, 4, etc…).

This project could be improved in many places that would in turn result in a more successful project development. Aspects of this project that were not implemented that could improve the overall quality of the application include: Using subjects retrieved from other Australian States as well as the use of sound to notify users, including students, of the conclusion of an exam through means of a ‘chime’.

This project was the first of its kind for me, taking on such a large task where we are given 100% freedom for what we wish to create. This project allowed me to take away valuable lessons that I will be able to apply to further studies, such as the importance of time management and achieving the desired goal before the rigorous perfection of minute details begin.