Software Requirements

Specification

for

Group Testing Environment

v3.0

15/04/2022

Prepared by

Filipe Lagrenade,

Sanjay Williams,

Jia Son Pow, and

Isaac Ellis

Capstone Group I3

**Table of Contents:**

[**Introduction**](#_uo2pradetltg) **5**

[Purpose](#_8x964nkwbjdo) 5

[Document Conventions](#_g7aioo8kgxis) 5

[Intended Audience and Reading Suggestion](#_8uedakq32uoj) 5

[Project Scope](#_zg0r3mfr1ct) 6

[References](#_bcqnjtfha5pw) 6

[**Overall Description**](#_3iql6dvjthx2) **7**

[Product Perspective](#_lw16t3bcvjav) 7

[Product Features](#_9d027pf2vsr) 7

[User Classes and Characteristics](#_ditjg6nw7qhz) 8

[Operating Environment](#_hi9lgwye34mf) 8

[Design and Implementation Constraints](#_s7w3y0kzz1z6) 9

[User Documentation](#_982c82xvmh0o) 9

[Assumptions and Dependencies](#_tsdeo3wfxac2) 9

[**System Features**](#_yzwwgyf9tu90) **11**

[**Account Management**](#_ldocooepmmrj) **11**

[Signing Up - Student](#_sl81ih766ezt) 11

[Signing up - UC/Invigilator](#_2m5e2d9tp3tu) 12

[Sign In - Student](#_jz2m8mgb9w8) 13

[Sign In - Assessment Supervisor](#_awifkqb9rbro) 15

[Password Reset - Student](#_axt7a98yhswe) 15

[Password Reset - UC/Invigilator](#_3vscgl8gsoth) 16

[**The Assessment Process**](#_toech9es1ebc) **17**

[Starting an Assessment](#_20wpaki03qrx) 17

[Sitting an Assessment](#_fe8jox8hdjk4) 19

[Changing computer during an Assessment](#_rxiljqdq9kzv) 20

[Test Time Alteration (During the test)](#_38mfzb4ok3ei) 21

[UC Send Alert to Students](#_wzag4njzoys2) 22

[UC/Invigilator View Student Messages](#_lhnshs6hr0go) 23

[Create Incident Notes During Test (Invigilator)](#_qhm1vugf38vd) 24

[Ending an Assessment](#_8d1lkc7piz0o) 25

[Viewing Work Folder after an Assessment](#_z6ewhaya4cmx) 26

[**Assessment Management**](#_s99krnoyasn8) **27**

[Uploading an Assessment](#_e8289pr7oj9o) 27

[Downloading Test Files](#_f05db49nwngo) 28

[System Management](#_tbd7ucmvmwoc) 30

[**Use Cases**](#_kyj2rulatgpg) **32**

[Student Use Cases](#_71kztzjlmsn2) 32

[Student Signup](#_4yuo4axg7phb) 32

[Student Sign In](#_oj0cflmut6so) 33

[Student starts assessment](#_eul7zsx9nzmy) 33

[Ask a question to UC/Invigilators](#_9vvnqm6y89t9) 34

[Assessment timer](#_rfebwt15jv93) 35

[Ability to end assessment early](#_a4rssg5e8qdh) 35

[Assessment runs out of time](#_75pir0rj3kra) 36

[Altered on 15 & 5 minutes left](#_9pm1n5ev48qi) 36

[Alerted to any UC/Invigilator notification](#_nyk0ogzhumv7) 36

[The ability to change computers during an assessment](#_8dcmixt4caj2) 37

[Student Password reset](#_1ozmu7t7x2dg) 37

[Assessment file syncing](#_mmcmq16b801h) 38

[Ability to force a save of work folder](#_k1in9ib7isss) 38

[The ability to view files after a test](#_e35866oo5562) 39

[UC/Invigilator Use Cases](#_z3retsvdejla) 39

[UC Signup](#_szee5evr9n70) 39

[Test Time Alteration](#_u5a0a7yepqsq) 40

[Password reset UC/Invigilator](#_cl4oelvthmi5) 40

[Assessment Creation](#_np1dk2tu3rmo) 41

[Sending Alerts](#_bjbn4yywjmu3) 41

[View Student Messages](#_wldx5rtxosal) 42

[Download Test Files (One Student at a time / All at once)](#_74skqifsi4y6) 42

[Create notes during the test](#_ahs4acigg0hr) 43

[View/Edit upcoming assessment](#_36jg2p7yceoy) 43

[Admin Use Cases](#_86ft7fkrnh) 44

[Health Monitoring](#_m2gkrd19k8xj) 44

[Performance Monitoring](#_awvkzb4yn3u9) 44

[Admin Sign In](#_xyarzxqpr5zu) 45

[View/edit all assessments](#_t81pteku6zhq) 46

[**External Interface Requirements**](#_z331pehwgths) **47**

[User Interfaces](#_422o0ecxso2e) 47

[Colour Schemes](#_k20xi3ldde5f) 47

[Student Client](#_6go08wws1tzg) 49

[Invigilator Client](#_mz90mqmozvzv) 59

[Hardware Interfaces](#_kslfku5ek7dk) 64

[Software Interfaces](#_jbt3v5tcmvjh) 64

[Communications Interfaces](#_3piqee3xft9) 64

[**Other Nonfunctional Requirements**](#_2rk2mhu8xne4) **64**

[Performance Requirements](#_ychdacb8v7rw) 64

[Persistent Error Messages](#_f3aaq13yxxnj) 64

[Help Alert Response Time](#_lo2l99m83q89) 65

[Ignore Alert Response Time](#_v1yfwyyejig1) 65

[Set up Efficiency](#_g9psbw7tbcpc) 65

[Uploading Assessment Documents](#_92ton1pqhg6) 65

[Folder/Server Backup](#_s9b2wu7h5sup) 65

[Coding Standard](#_ofvovzd3u8a8) 65

[Safety Requirements](#_405y4fai8ux) 65

[Security Requirements](#_tn2dxdf6ebw1) 65

[Software Quality Attributes](#_3afu39n2shx7) 66

[Robustness](#_t5tnwadnbkl) 66

[Maintainability](#_oqfyh5t2nn54) 66

[Availability](#_jiolf6xq183j) 66

[Usability](#_2308dfdf340n) 67

[**Other Requirements**](#_2h2rj5jlovdx) **67**

[Documentation](#_tx20vv878536) 67

[**Appendix A: Glossary**](#_kdibmuq2abbx) **68**

[**Appendix B: Analysis Models**](#_a0mv15ixx60n) **69**

[Top Level Flow Chart](#_binm7z9qna1s) 69

[**Appendix C: Issues List**](#_h9yxpjb6motg) **70**

[**Appendix D: Attributions**](#_2l1k61mrv21w) **71**

[**Appendix E: Agile Report**](#_2gj956ys4y3a) **72**

[Reflection](#_aqbivgh9yeo3) 72

[Group Toggl Report:](#_w1kvzcf7tlmm) 73

[Burndown Chart:](#_6zqrq05dc982) 81

[Meeting Minutes](#_37bkau2294e3) 82

[I3 Project Supervisor Meeting](#_4a27nq32vi5l) 82

# Introduction

The ‘Group Testing Environment’ is a software project that will be designed and implemented by Filipe Lagrenade, Sanjay Williams, Jia Son Pow, and Isaac Ellis on behalf of Dr Johannes Herrmann. The principal goal of the project is to create software that will securely distribute test papers and retrieve testresults within a computer laboratory. The project will be supervised by Arlen Brower.

## Purpose

This document covers the requirements for version 1.0 of the ‘Group Testing Environment’ software project as described above.

## Document Conventions

* ‘Client’ may refer to the product owner, Dr Hannes Herrmann, or in the context of software client
* Product Management refers to Mr Filipe Lagrenade and Mr Arlen Brower
* ‘test’ may refer to a university style test, where students answer questions to measure their performance, or a software assessment, where software is tested to ensure it meets requirements.
* GTE is an acronym for ‘Group Testing Environment’
* UC is an acronym for ‘Unit coordinator’

## Intended Audience and Reading Suggestion

Please refer to the list below for a guide on reading.

| Developers | Developers should read section 2, 3 and 4 |
| --- | --- |
| Project Management | Project Management should read the current document in its entirety and in order. |
| Clients | Clients should read the current document in its entirety and in order. |

## 

## Project Scope

The Curtin computer science department is planning on opening a large-scale computing laboratory which will hereafter be referred to as superlab. Consequently larger scale tests, requiring computers, can be conducted that would previously take several smaller computer laboratories, often at different times. With this in mind the super lab provides an opportunity to combat contract cheating. Furthermore there is a need for a new set of software to administer and manage these testsin a secure and synchronized manner. Additionally this software would need to distribute test papers and synchronously backup student work in a reliable manner.

## References

[1] “Educational Technology Services | Blackboard | North America” Accessed in March. 20, 2022. [Online]. Available: <https://www.blackboard.com/>

[2] R. Bandakkanavar, "Software Requirements Specification document with example - Krazytech", *Krazytech*, 2022. [Online]. Available: <https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>.

# 

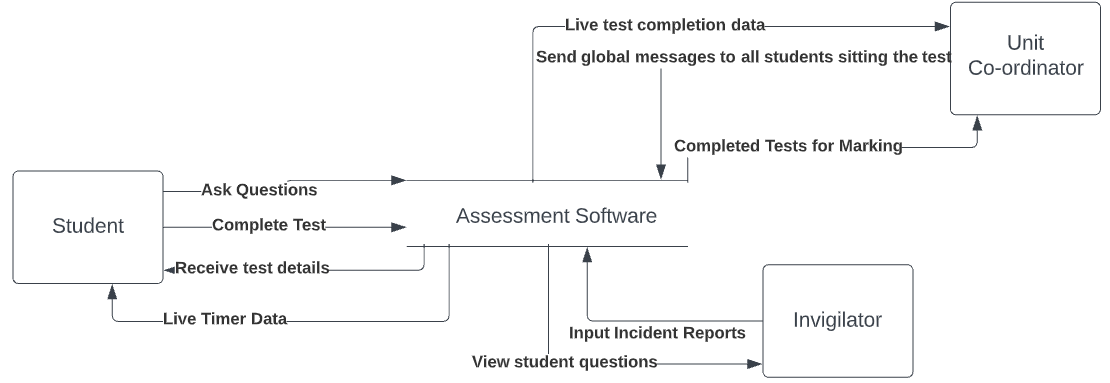
# **Overall Description**

## Product Perspective

The ‘*GTE*’ replaces the existing testing processes that rely on email and the proprietary software BlackBoard[1]. The software project aims to provide a platform for Invigilators and UC’s to securely create, plan and administer tests in the new super-lab. Along with a client for the computers of the super-lab that can receive test papers, synchronously backup student test data throughout tests. The test paper distribution and backing up of students' results tackles the issue of contract cheating by creating an equitable testing environment for students. This ensures that test papers are released in unison and that records of student work are kept. Additionally, the software will provide the ability for students, UC’s and Invigilators to send alerts to and from each other.

## Product Features

This product has the following features:

* Provide an accessible environment for a coding assessment through the distribution of authorised files
* It contains the code files or templates that are required for the assessment.
* Provides automation of time management through an on-screen timer
* This software will also feature auto saving and an intuitive submission system where the student would spend less time submitting their work at the end of the assessment without having the need to go onto an external learning management system, such as blackboard.
* Provides administrators with the ability to:
  + Easily see any questions raised by students during the sessions.
  + Track progress of students
  + Highlighting common issues that students may be having with the test
  + Allow communication across all students for issues such as corrections to assessment questions or emergency information.

## User Classes and Characteristics

The system will support three different kinds of users - student, invigilator and the unit coordinator. Students will have access to student functions, invigilators will have access to the invigilator functions and the unit coordinator should be able to access invigilator and unit coordinator functions.

The student should be able to perform the following functions:

* View a cover sheet outlining administrative details about the test
* Start and view the test.
* Read, add and edit items in the submission folder.
* Ask questions to the unit coordinator/invigilators.
* Submit the test.

The invigilators should have the following functionalities:

* See questions that the students are asking.
* Be notified of the location of a student in the laboratory who needs assistance.
* Create incident reports if there are any suspicions of cheating or academic misconduct.

The unit coordinator should be able to perform the following functions:

* The same functions as the invigilators.
* Send global messages out to all of the students currently undertaking the exam.
* View live completion data of the test (this is low-priority).

## Operating Environment

* The student application client will be running on a linux distribution which will be defined by the client.
* The UC/Invigilator client should be runnable remotely, likely from a browser based system.

## Design and Implementation Constraints

* This student application must run on the linux distribution that will be put on the laboratory computers.
* Must create a suitable environment in which students will be inclined to follow academic integrity guidelines.
* Must be made modular so that the client will be able to make modifications.
* The student view should be application based and not web-based (Though if it is deemed more applicable to be a web-based software then it is possible with client approval).
* Must be scalable.
* Must secure privileged information such as test papers and results.
* Must secure student information such as names, student ID’s, CAP information etc.

## User Documentation

* User manuals will be provided for:
  + Students
  + Administrators/Invigilators
  + Set-up documentation

## Assumptions and Dependencies

* It is assumed that this is to only be made available in a linux version defined by the university.
* It is also assumed that the client will handle the majority of distribution onto the laboratory computers after the development team gives ample demonstration on how to set up the system correctly.
* Currently assuming that any file storage will be based on a cloud-based solution but this can be subject to change.

# 

# **System Feature**s

## Account Management

### Signing Up - Student

* + - 1. **Description and Priority**

**Description:**

Students that do not have an account and are listed as sitting an assessment will receive a signup email. This email will provide them with a signup link that will redirect them to a signup page. The signup page will allow them to create an account.

**Assumption:**

The student is a valid Curtin student or staff member.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* A UC will create an assessment through the UC web portal that will require them to specify a list of Curtin student ids.
* From this list of student ids, the backend server will ascertain which student ids are not related to an existing account. These student ids will be emailed a signup link
* The student will click the link through their email client
* The user will be prompted to enter their Student/Staff ID number and press a ‘Next’ Button to begin the process.
* The user will then be prompted to enter a password of their choice and be asked to re-enter the password into a confirmation bar after which, the user will confirm their password by pressing the ‘Sign Up’ Button and they will be taken to another page.
* The page will likely contain an insert text bar where the user will be prompted to enter an alphanumeric code that was sent to their student email.
* After retrieving the code from their email and entering it into the bar, the user will be taken to a confirmation page displaying that their account was successfully created.
  + - 1. **Functional Requirements:**
    - The backend system must be able to send emails
    - The backend system must be able to ascertain what student ids require accounts based on the upcoming assessments
    - The web page and backend must provide validation for the signup process (emails, names, student ids and passwords must be valid)
    - Signup tokens related to the signup process must be securely stored
      1. **Related Use Cases**
* [Student Signup](#_4yuo4axg7phb)

### Signing up - UC/Invigilator

* + - 1. **Description and Priority**

**Description:**

The user, whether an Invigilator or Unit Coordinator will be able to sign up using the native or web applications. The user will be led through the signup process by the application and will be able to successfully create a new account based on some university/personal details.

**Assumption:**

The user is a valid Curtin student or staff member and must have been made eligible, likely by the unit coordinator, to apply for account activation.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* The user will begin the process by having the application, whether the native or web application, and be open to the initial page.
* There will be a ‘New user? Sign up here’ link under the login panel which, when clicked on, will take the user to a signup window.
* The user will be prompted to enter their Student/Staff ID number and press a ‘Next’ Button to begin the process.
* The user will then be prompted to enter a password of their choice and be asked to re-enter the password into a confirmation bar after which, the user will confirm their password by pressing the ‘Sign Up’ Button and they will be taken to another page.
* The page will likely contain an insert text bar where the user will be prompted to enter an alphanumeric code that was sent to their student email.
* After retrieving the code from their email and entering it into the bar, the user will be taken to a menu page, completing the signup process.
  + - 1. **Functional Requirements:**
    - The backend system must be able to send emails
    - The backend system must be able to ascertain what student ids require accounts based on the upcoming assessments
    - The web page and backend must provide validation for the signup process (emails, names, student ids and passwords must be valid)
    - Signup tokens related to the signup process must be securely stored
      1. **Related Use Cases**
* [UC Signup](#_szee5evr9n70)

### Sign In - Student

* + - 1. **Description and Priority**

**Description:**

A student, when using this software, will be able to log in using their Student ID and their given password. The application will prompt them through the login process, at the end of which they will be taken to the student dashboard.

**Assumption:**

The user must have previously activated their account.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**

The user will initially be shown a blank page with only a text input with the words, ‘Enter Student ID’ and a button with the word, ‘Next’.

If a valid ID number has been typed into the text edit bar when the ‘Next’ button has been pressed, the user will be taken to the next page where they will be prompted to enter their activated app password into an edit text bar, underneath which there will be a ‘Sign In’ button in place of the ‘Next’ button.

If the correct password associated with that particular ID is present in the edit text bar when the sign in button is pressed, the user will be taken to a menu or straight to the dashboard, completing the login process.

* + - 1. **Functional requirements:**
* The system must alert the user if a match is not found for the entered ID with an “Invalid ID” message.
* The system must warn the user when they attempt to log in using an account that hasn’t yet been activated and provide them with a link to begin the account activation process.
* The system must alert the user if their password is incorrect
  + - 1. **Related Use Cases**
* [Student Signin](#_oj0cflmut6so)

### Sign In - Assessment Supervisor

* + - 1. **Description and Priority:**

**Description:**

An Assessment Supervisor, when using this software, will be able to log in using their Staff ID and their given password. They will likely log into a web application which can be opened in a browser. The web application will prompt them through the login process, at the end of which they will be taken to the staff dashboard.

**Assumption:**

The user must have previously activated their account.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**

The login process will be functionally similar to the student login process. (see section 3.1.1)

* + - 1. **Functional Requirements:**
* The functional requirements for this system feature include the functional requirements of the student login process. (see section 3.1.3.3)

### Password Reset - Student

* + - 1. **Description and Priority:**

**Description:**

A student or UC/Invigilator will be able to reset the password for their account. Password can be reset by receiving a password reset email and submitting a new password

**Assumption:**

The user must have previously activated their account.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* The user selects a password reset link on either the UC client or the student client
* The user is prompted to submit their email address.
* When the user submits their email address the system sends a password reset email to their email address.
* When the user follows the link provided in the password reset email they are redirected to the password reset page of the UC client.
  + - 1. **Functional Requirements:**
* The backend system must be able to send emails.
* The backend must be able to verify the student’s identity through the use of a student ID.
* The system must be able to alert the user if the password which has been entered is incorrect.
* The system must notify the student if the password they have entered does not meet the security requirements
  + - 1. **Related Use Cases**
* [Student Password reset](#_1ozmu7t7x2dg)

### Password Reset - UC/Invigilator

* + - 1. **Description and Priority:**

**Description:**

A student or UC/Invigilator will be able to reset the password for their account. Password can be reset by receiving a password reset email and submitting a new password

**Assumption:**

The user must have previously activated their account.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* The user selects a password reset link on either the UC client or the student client
* The user is prompted to submit their email address.
* When the user submits their email address the system sends a password reset email to their email address.
* When the user follows the link provided in the password reset email they are redirected to the password reset page of the UC client.
  + - 1. **Functional Requirements:**
* The backend system must be able to send emails.
* The backend must be able to verify the staff members identity through the use of a staff ID.
* The system must be able to alert the user if the password which has been entered is incorrect.
* The system must notify the staff member if the password they have entered does not meet the security requirements.
  + - 1. **Related Use Cases**
* [Password reset UC/Invigilator](#_cl4oelvthmi5)

## The Assessment Process

### Starting an Assessment

* + - 1. **Description and Priority:**

**Description:**

A student will be able to use the application to set up and begin an assessment. The application will take the student through a simple setting up process in which they will agree to abide by the specific restrictions of the assessment they are about to take.

**Assumption:**

The user must be correctly logged into the system as a student.

**Priority:**

The priority of this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* The user will begin the process by pressing the appropriate assessment for that session, likely found in a folder in a menu or on the student dashboard.
* This will take them to a page outlining the banned materials for that assessment session. The list could include items such as smart watches, mobile phones, earphones, graphical calculators, etc.
* Agreeing to the list, likely by ticking a box on the page, will take the user to a page outlining the websites and native apps banned from that assessment session. The list could include applications such as Discord, Facebook, Instagram, etc.
* Any other rules involving plagiarism will be shown in this process possibly including an originality checklist of statements including a “The work I am submitting is entirely my own…”.
* The user will be taken to a page with a countdown until the assessment will be made available, completing the setup process.
  + - 1. **Functional Requirements:**
* The system must be able workout what assessments the student can take at the current time
* The system must be able to present the user with plagiarism and banned manned material rules.
  + - 1. **Related Use Cases**
* [Student start assessment](#_eul7zsx9nzmy)

### Sitting an Assessment

* + - 1. **Description and Priority**

**Description:**

A student will be able to sit an assessment and be tracked and monitored using the application. Once started, a student will be able to see the time they have left to complete the assessment, have the ability to ask a question of the Unit Coordinator, and have the application tell them how long ago their most recent save was. They will then be able to finish their session, at which point the assessment sitting process will be complete.

**Assumption:**

The user has logged in correctly and started an appropriate assessment session.

**Priority:**

The priority for this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**

After starting an assessment, a student user will likely be taken to their assessment dashboard where they will see a number of things:

* A **Countdown Timer which** will have the current time remaining for that assessment session. This may have feedback for certain events such as, turning another colour when there are 15 and 5 minutes left on the timer, and turning another colour when time is added by the Unit Coordinator.
* A **‘Question/Concern’ Button** which, when pressed, will open a panel / another window containing a text input and a ‘Send’ Button which, when pressed will notify the user that the message has been sent to the Unit Coordinator / Invigilator and that they will be notified when the receive a response.
* A **Notification/Message Window/Panel** which will display any incoming messages from the Unit Coordinator / Invigilator. This will either be hidden during normal use or be a discreet panel on the side, or will open in a separate window.
* The **number of questions completed and/or remaining** which will possibly keep track of where the user is up to, or be manually checked by the student user.
* The **time of the last save** which could also notify the user when the application is backing up the student’s work by turning into the text, “Saving…”.
* The system will **automatically sync** the students work folder to the backend server
* A **‘Save Work Folder’** button which, when pressed, ensures that the current work folder is saved to the backend system.
  + - 1. **Functional Requirements:**
* The system must be able to display an accurate assessment timer
* The user must be able to ask question to a UC/Invigilator via the student client
* The system must be able to present Notification from the UC
* The system must be able to notify students when they have limited time left
* The system must provide a mechanism for saving a students work
* The system must provide the ability to end an assessment early
* The system must be able to sync the student work folder to the backend server
  + - 1. **Related Use Cases**
* [Ask question to UC/Invigilators](#_9vvnqm6y89t9)
* [Assessment timer](#_rfebwt15jv93)
* [Alertered on 15 & 5 minutes left](#_9pm1n5ev48qi)
* [Alerted to any UC/Inviligator notification](#_nyk0ogzhumv7)
* [Ability to force a save of work folder](#_k1in9ib7isss)
* [Assessment file syncing](#_mmcmq16b801h)

### Changing computer during an Assessment

* + - 1. **Description and Priority**

**Description:**

A student may change computers during an assessment and resume on another computer with access to their previous work folder.

**Assumption:**

The student has logged in and resumed an appropriate assessment after doing part of the assessment on another computer.

**Priority:**

The priority for this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* A student resumes their assessment
* The student client displays the progress of the download of the student's previous work folder. Once downloaded it displays that its has download successfully and prompts the user to the location of the folder
  + - 1. **Functional Requirements:**
* The system must be able to retrieve the most recently updated test data from the database
  + - 1. **Related Use Cases**
* [The ability to change computers during an assessment](#_8dcmixt4caj2)

### Test Time Alteration (During the test)

* + - 1. **Description and Priority**

**Description:**

During the test student(s) may need to have the amount of time changed on their computer for various reasons such as interruptions, outages etc. The Unit Coordinator should be able to change the timer on any given student's computer to reflect the actual amount of time which is remaining on the test.

**Assumption:**

The student and unit coordinator must be logged into their respective clients.

**Priority:**

The priority for this system feature is **Low.**

* + - 1. **Stimulus/Response Sequences:**
* In the UC client, the unit coordinator will select the given computers that they want the time changed on
* The backend will be notified of the changes and will push the changes to the respective student clients
* The student clients will show the new correct time
* The system will keep track of changes in a log for auditing purposes.
  + - 1. **Functional Requirements:**
* The system must be able adjust the duration of a students assessment
  + - 1. **Related Use Cases**
* [Test Time Alteration](#_u5a0a7yepqsq)

### UC Send Alert to Students

* + - 1. **Description and Priority**

**Description:**

The UC/Invigilators are able to send global alerts out to all students taking a given assessment.

**Assumptions:**

* The UC/Invigilators have an account which is logged into the UC client
* Students have an account which is logged into the student client

**Priority**

The priority for this system feature is **Medium.**

* + - 1. **Stimulus/Response Sequences:**
* UC Clicks a button which says something similar to “send alert”
* A small pop-up browser will appear and there will be a textbox where the UC will be able to type the alert message in. This pop-up will not interfere with any important screen visibility
* UC types the alert they want to send and clicks the send button
* Backend process send the alert message to all student computers in the form of a small, non-obtrusive pop-up banner
* Students can close the pop-up once they have read the message.
  + - 1. **Functional Requirements:**
* The system must store the messages sent to and from for auditing purposes
* The system must be able to send and retrieve alerts
  + - 1. **Related Use Cases**
* [Sending Alerts](#_bjbn4yywjmu3)

### UC/Invigilator View Student Messages

* + - 1. **Description and Priority**

**Description:**

The unit coordinators and invigilators are able to view messages/questions that students send in regarding the assessment.

**Assumption:**

* The UC/Invigilators have an account which is logged into the UC client
* Students have an account which is logged into the student client

**Priority:**

The priority for this system feature is **Medium**.

* + - 1. **Stimulus/Response Sequences:**
* The student will click a “send question” button
* Student types message and sends to invigilator
* A small pop-up/banner will appear on the invigilator screen which does not interfere with any important screen visibility
* Invigilators to action the question as necessary
* Invigilators to tick off once they have actioned a question
  + - 1. **Functional Requirements:**
* The system must be able to receive messages
* The system must be able to mark the messages as resolved
  + - 1. **Related Use Cases**
* [View Student Messages](#_wldx5rtxosal)

### Create Incident Notes During Test (Invigilator)

* + - 1. **Description and Priority**

**Description:**

The invigilators are able to create notes during the test of potential cheating/collusion as well as any incidents that may have occurred

**Assumptions:**

* Invigilators have an account which is logged in that has access to the relevant test
* Test has begun

**Priority:**

The priority for this system feature is **Medium.**

* + - 1. **Stimulus/Response Sequences:**
* Invigilator clicks on a button named something similar to “notes”
* Invigilator logs which computers or student numbers are related to the incident
* Invigilator logs what events occurred
* Invigilator logs what time the events occurred
* Invigilator may have an option to tick if it is just a note or if it is something that may be of high concern
* Note gets saved in database by system
  + - 1. **Functional Requirements:**
* The system must store the messages sent to and from for auditing purposes
* The system must be able to send and recieve messages
  + - 1. **Related Use Cases**
* [Create notes during the test](#_ahs4acigg0hr)

### Ending an Assessment

* + - 1. **Description and Priority**

**Description:**

An assessment ends when the time to complete it runs out. A student may also choose to end the assessment early.

**Assumption:**

The student is logged in and started an appropriate assessment session.

**Priority:**

The priority for this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* On the student client assessment dashboard , a student will see the **‘Submit Assessment’ Button** which, when pressed, will confirm that the student is sure they would like to end their assessment. After confirmation it will notify the user that their document is being saved for the final time and then will close the application window, ending the assessment sitting process.
* If the time remaining in the assessment reaches zero the student will be notified before closing the application window.
* On an assessment ending a student's work folder will be synced to the backend server a final time before making the folder inaccessible to the student.
  + - 1. **Functional Requirements:**
* The system must be able to sync the work folder with the backend
* The system must be able to restrict access to the work folder
  + - 1. **Related Use Cases**
* [Ability to end assessment early](#_a4rssg5e8qdh)
* [Assessment runs out of time](#_75pir0rj3kra)

### Viewing Work Folder after an Assessment

* + - 1. **Description and Priority**

**Description:**

After an assessment has been completed the students' work folders are inaccessible to them. A UC can choose via the UC client to make these folders accessible to the student.

**Assumption:**

The student has completed an assessment and the UC has chosen to make an assessments work folder viewable to the student.

**Priority:**

The priority for this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* The student client detects that a work folder from a previous assessment has been marked accessible to the student by the UC, via the backend server. The student client then adjusts the permissions of the folder to make the folder accessible to the student.
  + - 1. **Functional Requirements:**
* The student client must be able to sync the work folder with the backend
* The student client must be able to restrict access to the work folder
  + - 1. **Related Use Cases**
* [The ability to view files after a test](#_e35866oo5562)

## Assessment Management

### Uploading an Assessment

* + - 1. **Description and Priority**

**Description:**

A Unit Coordinator or other authorised staff member can upload an assessment using the web application. The process includes providing certain information including, the assessment document(s) itself, lists of staff and students to allow access, information about the assessment session (session start time, duration, banned materials/apps, etc.). At the end of this process, the user will have successfully uploaded the assessment.

**Assumption:**

The user has authorisation to edit the session they are attempting to upload documents for.

**Priority:**

The priority for this system feature is **High**.

* + - 1. **Stimulus/Response Sequences:**
* After clicking on the session that they are attempting to upload documents to, the user will likely be taken to a page where they can upload the assessment document(s) that will be made available to the students taking the assessment. This will likely be done using a choice of a standard file manager upload, or a drag-and-drop method, allowing the user to drag a file from their own instance of a file manager to the application window.
* After completing uploading the necessary files, the user will be taken to a page where they will be prompted to upload a list of students that the assessment will be available for. This will likely be done by uploading a .csv file of the students’ ID numbers. The application will show the user the list of students to acknowledge to upload.
* The next page will be similar to the last except that the user will be uploading (likely by .csv) a list of the IDs of the staff that will have access, or part access, to the assessment documents and/or the admin dashboard.
* After being prompted, the user will then input the start time of the assessment session and the session’s length (this could include multiple sessions for the same assessment).
* Another step in this process is the uploading of a list of physical materials that are banned from that particular assessment/session, as well as a list of the applications and websites that the students will not be able to / allowed to access during the assessment. This will likely have a similar process to the uploading of the assessment document(s).
* Various other pieces of information will be taken during this process such as whether an assessment will be marked anonymously (with student names removed), whether or not an assessment will have a reading time and how long that would be, and if there is a time period at the end of the assessment session after which students will not be permitted to exit the assessment venue.
* After completing this process, the user will be shown a summary of the assessment session(s) to confirm that the information that the application has is correct. If the user confirms, the application will likely take them back to the dashboard or a menu page, completing the assessment upload process.
  + - 1. **Functional Requirements:**
* The system must be able to interpret a list of student ids to student users
* The system must allow for assessment documents to be uploaded
* The system must be able to save and validate assessment information

### Downloading Test Files

* + - 1. **Description and Priority**

**Description:**

Test Markers are able to download the completed student test files from the database

**Assumptions:**

The test must have been completed by the students whose assessments need downloading

Markers must have access to an account which has access to the relevant test

**Priority:**

The priority for this system feature is **High.**

* + - 1. **Stimulus/Response Sequences:**
* Marker logs into the database where the tests are being stored
* Marker selects the test(s) that they want from a list (potentially with a checkbox system), or clicks a select all button.
* Marker downloads selected files
  + - 1. **Functional Requirements**
* The computer that the web application is being accessed on must be connected to the internet.
* The browser that the web application is running on must be installed correctly.
  + - 1. **Related Use Cases**
* [Assessment Creation](#_np1dk2tu3rmo)
  + 1. **View/Edit Upcoming Assessment**
       1. **Description and Priority**

**Description:**

UC’s are able to view and edit any upcoming assessments that they have access to.

**Assumptions:**

* The UC must be logged into an account which has access to the relevant assessments
* The test must not have already started
  + - 1. **Stimulus/Response Sequences:**
* UC Selects assessment which they would like to view and/or edit
* System shows screen to UC with all of the relevant information including date, time, which students are sitting the test and questions being asked
* (Assuming they want to edit) UC updates any of the fields and then clicks save
* System is notified of changes
* If new tests were put in, system updates database with new test papers
  + - 1. **Functional Requirements**
* The system must be able to update assessment information
* The system must be able to upload assessment documents
  + - 1. **Related Use Cases**
* [Download Test Files (One Student at a time / All at once)](#_74skqifsi4y6)

## System Management

* + 1. **Health Monitoring**
       1. **Description and Priority**

**Description:**

Admins are able to view the performance of all of the applications which are active on the lab computers. They should also be notified of any health concerns of either the server or one/many of the applications running.

**Assumptions:**

The admin has high-level access to the server

**Priority:**

The priority of this system feature is **High.**

* + - 1. **Stimulus/Response Sequences:**
* Server sends automated help message to admin via email
* Admin logs into the server
* Admin assesses the cause of the problem
* Once the problem is solved, the admin will then reset the emergency alert mechanism on the server
  + - 1. **Functional Requirements**
* Provide observability to usage of
  + CPU
  + Threading
  + Network usage
* Must be able to send emails, when conditions based of observability metrics are met. E.g when ram usage > 90%
* Display Observability metrics as graphs
  + - 1. **Related Use Cases**
* [Health Monitoring](#_m2gkrd19k8xj)
  + 1. **Performance Monitoring**
       1. **Description and Priority**

**Description:**

Admins are able to monitor and tune the performance throughput of the server to meet the demands from students/UC’s.

**Assumptions:**

* The admin has high-level access to the server
* The students/UC’s must have a communication channel to relay information to the admins

**Priority:**

The priority of this system feature is **High.**

* + 1. **Stimulus/Response Sequences:**
* Students/Uc’s send information about the required performance changes to the admin.
* Admin tunes the throughput of the server's power, balancing between the end users’ demands and the server’s health.
* Once the tuning is completed, admin will send a reply to the students/UC’s about changes made to the server
  + 1. **Functional Requirements:**
* Provide observability to usage of
  + Rsync processes
    - Num processes
    - Mean duration
    - Max duration
  + API usage
* Must be able to send emails, when conditions based on observability metrics are met. E.g when rsync num process > 100
  + - 1. **Related Use Cases**
* [Performance Monitoring](#_awvkzb4yn3u9)

# Use Cases

## Student Use Cases

### Student Signup

| Description | Students are required to setup up an account before accessing the student client. Signup emails will be sent to a student’s Curtin email addresses when they are first registered in an assessment by a UC |
| --- | --- |
| Stakeholder/Actors | * Student * UC |
| Pre-Conditions | * The student must not have an account * A UC must create an assessment that includes the student's Curtin ID number as a participant |
| Details/Flow | * UC creates assessment with the student as a participant via the UC frontend * Backend processes request sending signup emails to students that do not have accounts * Student receives signup email and clicks signup link in the email * Student fills out signup page on UC portal * The backend system processes the request creating an account for the student * The backend sends the student an email confirming their account creation |
| Goal/Post-Condition | * The student now has an account registered with backend system |
| Exception | * Invalid passwords (rules to be specified) * Duplicate accounts * Expired verification token |
| Comments |  |

### Student Sign In

| Description | Students are required to log into the student client before accessing assessments |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * The student has a registered account * The student has access to the student client |
| Details/Flow | * The student opens the student client, which displays a login screen * The student submits their email address and password. * The student client receives an authentication token from the backend server * The login screen is replaced by the student client dashboard/landing page |
| Goal/Post-Condition | * The student gains access to the student client * The student client is able to access APIs on the backend server |
| Exception | * Invalid password * Invalid email address |
| Constraints |  |
| Variants/Alternate | * Email address for the login process could be replaced with student id |
| Comments |  |

### Student starts assessment

| Description | When a student starts an assessment, they are required to acknowledge a page detailing banned materials and a page detailing rules involving plagiarism. |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student is logged in on the student client * Student has an assessment that has started or is starting in the next 5 minutes |
| Details/Flow | * The student selects the assessment from the student client dashboard. * The student is shown a page detailing banned materials which prompt them to agree to these conditions. * The student is shown a page detailing rules about plagiarism which prompts them to agree to these conditions * If the assessment has not already begun a count down timer is displayed to the student * Once the assessment has begun the student is provided with their test papers and a link to a directory to work from. |
| Goal/Post-Condition | For the student to successfully start an assessment |
| Exception | * Student does not accept any required conditions * The student has no assessments * The student's assessment has already finished * The student has already opted to end this assessment early |
| Comments |  |

### Ask a question to UC/Invigilators

| Description | Students are able to ask questions to UC/Invigilators during a test. The questions will be displayed on the UC/Invigilators client |
| --- | --- |
| Stakeholder/Actors | * Student * UC/Invigilators |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * Student selects to ask a question from the student client * The student submits a question. * The question is processed by the backend server that is sent to UC/Invigilator client |
| Goal/Post-Condition | For the question to be sent to UC/Invigilator |
| Comments |  |

### Assessment timer

| Description | The student should be able to see how much time they have left during the assessment |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * The student client displays a countdown timer and clock that show how long is left during the assessment |
| Goal/Post-Condition | * To display how long is left during an assessment |
| Comments |  |

### Ability to end assessment early

| Description | Students can opt to end their assessments early |
| --- | --- |
| Stakeholder/Actors | * Student |
| Includes | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * The student selects to end their assessment * The student is prompted to confirm that they would like to end the assessment * The students work folder is synced for a final time * The student is disallowed from accessing the assessment and work folder * The student client leaves the assessment. |
| Goal/Post-Condition | For the student to end the assessment early, syncing their work and barring them from accessing the assessment. |
| Comments |  |

### Assessment runs out of time

| Description | Once an assessments timer runs out, the student is unable to make any further changes to their documents |
| --- | --- |
| Stakeholder/Actors | * Student |
| Includes | * Student is logged in on the student client * Student has started an assessment * The assessment has run out of time |
| Details/Flow | * The student is notified that they are out of time * The students work folder is synced for a final time * The student is disallowed from accessing the assessment and work folder |
| Goal/Post-Condition | For the student to end the assessment early, syncing their work and barring them from accessing the assessment. |
| Comments |  |

### Altered on 15 & 5 minutes left

| Description | The student client should notify the student when there are 15 minutes left and when there are 5 minutes left. |
| --- | --- |
| Stakeholder/Actors | * Student |
| Includes | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * When there are 15 or 5 minutes left during the assessment; the student client alerts the user to the amount of time left. |
| Goal/Post-Condition | To notify the student when the assessment is close to ending |
| Comments |  |

### Alerted to any UC/Invigilator notification

| Description | When the UC/Invigilators send notifications, the student client should notify the student of their contents |
| --- | --- |
| Stakeholder/Actors | * Student * UC/Invigilator |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment * The UC/Invigilator has sent a notification to the student via the UC client |
| Details/Flow | * The student client receives a UC/Invigilator notification from the backend server * The student client displays an alert containing the message of the notification |
| Goal/Post-Condition | To alert the student to notifications sent by the UC invigilator |
| Comments |  |

### The ability to change computers during an assessment

| Description | There are likely circumstances during assessments when a student needs to change computers. It is important that if this occurs, the student client can load the student's work documents. |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment * Student was previously doing the assessment on another computer |
| Details/Flow | * The student client initialises the assessments work folder by downloading the student's previously synced work folder from the backend server * The student client displays the progress of this download |
| Goal/Post-Condition | To allow students to change computer during an assessment without losing their work |
| Constraints | There may be differences between the downloaded work folder and the work that student had previously done if the syncing/saving is not up to date. |
| Comments |  |

### Student Password reset

| Description | Students can reset the passwords for their accounts |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student has an account |
| Details/Flow | * Student clicks the reset password link on either the student client or on the Student client. * The student submits their email address to the backend. * The backend sends a password reset token to the students email. * The student receives the password token and submits it to the student client. * The student submits their new password to the student client |
| Goal/Post-Condition | To allow students to reset their passwords |
| Exception | * The token for the reset link expires. * The email address provided by the user does not belong to an account |
| Comments |  |

### Assessment file syncing

| Description | During the assessment a students work folder should be synced to the backend server |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * When a student creates, edits, or deletes a file in the work folder it is synced to the backend |
| Goal/Post-Condition | To keep an up to date version of a students work folder on the backend server |
| Exception | * File permissions locking * File corruption |
| Comments |  |

### Ability to force a save of work folder

| Description | A student is able to save their work folder by pressing a save button. |
| --- | --- |
| Stakeholder/Actors | * Student |
| Pre-Conditions | * Student is logged in on the student client * Student has started an assessment |
| Details/Flow | * The student clicks the save button in the student client. * All files in the work folder are synced to the backend. * The student client displays a message indicating that the save completed |
| Goal/Post-Condition | The students work folder is successfully synced to the backend. |
| Comments |  |

### The ability to view files after a test

| Description | The work folder disappears once a test completes. However, a student's work folder should reappear once they have been released by the UC. |
| --- | --- |
| Stakeholder/Actors | * Student * UC |
| Pre-Conditions | * Student is logged in on the student client * The UC has selected to release the work folders back to students |
| Details/Flow | * The student selects to download the assessment * The work folder is downloaded as zip file |
| Goal/Post-Condition | To allow students to view their work after the assessment and once the UC has marked the assessment as ready for release. |
| Exception | * Work folder can't be downloaded as it has not been released by the UC |
| Comments |  |

## UC/Invigilator Use Cases

### UC Signup

| Description | UCs are required to set up an account before accessing the UC client. Signup emails will be sent to a UC’s Curtin email address when they are first registered on the system by an admin. |
| --- | --- |
| Stakeholder/Actors | * Admin * UC |
| Pre-Conditions | * The UC must not have an account * An admin must create a profile that includes the UC's Curtin staff ID. |
| Details/Flow | * UC sends a request for a UC account to the admin via email. * Admin approves the request and sends an email reply with an account signup link. * UC receives the signup email and clicks the signup link in the email * UC fills out the signup page on UC portal * The backend system processes the request creating an account for the UC. * The backend sends the UC an email confirming their account creation |
| Goal/Post-Condition | * The UC now has an account registered with the backend system |
| Exception | * Invalid passwords (rules to be specified) * Duplicate accounts * Expired verification token |
| Comments |  |

### Test Time Alteration

| Description | A UC is able to alter the timer on a student’s client |
| --- | --- |
| Stakeholder/Actors | * Student * UC/Invigilator |
| Pre-Conditions | * Both the UC and student must have logged in to their respective client accounts. |
| Details/Flow | * UC navigates to the student computer monitoring page on the client. * UC alters the countdown timer on a particular student’s computer. |
| Goal/Post-Condition | * The UC now has given the student the right amount of time required to complete an assessment |
| Exception | * The student’s allocated time has run out in a fair manner. * The student has finished their test ahead of time and ended their session. |
| Comments |  |

### Password reset UC/Invigilator

| Description | UC/Invigilators are able to reset the passwords for their accounts |
| --- | --- |
| Stakeholder/Actors | * UC/Invigilators |
| Pre-Conditions | * User has an account |
| Details/Flow | * User clicks the reset password link on the UC client * The user submits their email address to the backend * The backend sends a password reset email to the user * The user follows the password reset link from the email * The user submits their new password |
| Goal/Post-Condition | To allow users to reset their passwords |
| Exception | * The token for the reset link expires * The email address provided by the user does not belong to an account |
| Comments |  |

### Assessment Creation

| Description | UC/Invigilators are able to create/upload a new assessment |
| --- | --- |
| Stakeholder/Actors | * UC/Invigilators |
| Pre-Conditions | * User has an account on the UC client. |
| Details/Flow | * User logins to their account on the client. * The user fills out the form required to upload a folder. This includes assessment start time, test duration, name, a CSV of student IDs for the assessment and test documents. * The user uploads the folder. * The backend process detects the folder and sends it to all student computers. |
| Goal/Post-Condition | The UCs now have assessments available for students to view/complete. |
| Comments |  |

### Sending Alerts

| Description | UC/Invigilators can send global alerts to all students partaking in an assessment. |
| --- | --- |
| Stakeholder/Actors | * UC/Invigilators * Students |
| Pre-Conditions | * UCs/Invigilators have an account logged in to the UC client. * Students have logged in to the Student client. |
| Details/Flow | * UC clicks the Send Alert button. * A small pop-up browser will appear and there will be a textbox where the UC will be able to type the alert message in. This does not interfere with any important screen visibility * UC types the message and clicks the Send button. * The backend process sends this alert message to all student computers in the form of a small pop-up banner. * Student clicks on the pop-up banner to read the alert message. * Student can close the pop-up once they have read the message. |
| Goal/Post-Condition | The students will be made aware of the important announcements made by the UCs/Invigilators. |
| Exception | * Students who have finished the assessment before an alert will not be held responsible for any changes announced and will be marked accordingly |
| Comments |  |

### View Student Messages

| Description | Invigilators are able to view messages that students send in regarding the assessment. |
| --- | --- |
| Stakeholder/Actors | * Invigilator (may include UC) * Student |
| Pre-Conditions | * Invigilator has an account that's logged in which as access to the related assessment * Student is logged in to the software |
| Details/Flow | * Student Clicks send question button * Student types message and sends to Invigilator * A small pop-up/banner will appear on the Invigilator screen which does not interfere with any important screen visibility * Invigilators to action the question as necessary * Invigilators to tick off once they have actioned a question |
| Goal/Post-Condition | * The student has received an answer to their question |
| Comments |  |

### Download Test Files (One Student at a time / All at once)

| Description | Test Markers are able to download student files from a database. |
| --- | --- |
| Stakeholder/Actors | * Test Markers (May include UC) |
| Pre-Conditions | * The test must have been completed by the students whose assessments need downloading * Markers must have access to an account which has access to the relevant test |
| Details/Flow | * Marker logs into the database where the tests are being kept * Marker selects the test(s) that they want from a list (potentially with a checkbox system), or selects them all * Marker downloads selected files |
| Goal/Post-Condition | * The markers have downloaded the tests they would like to view |
| Comments |  |

### Create notes during the test

| Description | The Invigilators are able to create notes during the test of potential cheating/collusion as well as any incidents that may have occurred |
| --- | --- |
| Stakeholder/Actors | * Invigilators (May include UC) |
| Pre-Conditions | * Invigilators have an account that is logged in that has access to the relevant test * Test has begun |
| Details/Flow | * Invigilator clicks on a button named something similar to “notes” * Invigilator logs which computers or student numbers are related to the incident * Invigilator logs what events occurred * Invigilator logs what time the events occurred * Invigilator may have an option to tick if it is just a note or if it is something that may be of high concern * Invigilator saves note * Note gets saved in the database by system |
| Goal/Post-Condition | * The Invigilator has successfully logged a note which is saved in the database |
| Comments |  |

### View/Edit upcoming assessment

| Description | UC’s are able to view and edit any upcoming assessments that they have access to |
| --- | --- |
| Stakeholder/Actors | * UC |
| Pre-Conditions | * The UC must be logged in to an account that has access to the relevant assessments * The test must not have already started |
| Details/Flow | * UC selects assessment which they would like to view and/or edit * System shows screen to UC with all of the relevant information for the assessment including date, time, which students are sitting the test and questions being asked * (Assuming they want to edit) UC updates any of the fields and then clicks save * System is notified of the changes * If new test questions were put in, system updates database with new test paper |
| Goal/Post-Condition | * The UC has viewed the assessment they would like to and has made any changes that they wanted to |
| Exception | * Invalid dates/times selected for the test * Invalid file type uploaded for the test * Invalid student numbers added |
| Comments |  |

## Admin Use Cases

### Health Monitoring

| Description | Admins are able to view the performance of the backend server and student clients |
| --- | --- |
| Stakeholder/Actors | * Admins |
| Pre-Conditions | * The admin must have high-level access to the server. |
| Details/Flow | * Server sends an automated help message to the admin via email * Admin logs in to the server. * Admin assesses the cause of the problem. * Once the problem is solved, the admin will then reset the emergency alert mechanism on the server. |
| Goal/Post-Condition | * The admin has viewed and addressed the problems on the server. |
| Comments |  |

### Performance Monitoring

| Description | Admins can view the performance throughput of the server to meet the demands of students/UCs. |
| --- | --- |
| Stakeholder/Actors | * Admins * UCs/Invigilators * Students |
| Pre-Conditions | * The admin must have high-level access to the server. * The students/UCs must have a communication channel to relay information to the admins. |
| Details/Flow | * Student clients and backend server provide observable metrics * Admin tunes the throughput of the server’s power, balancing between the end users’ demands and the server’s health. |
| Goal/Post-Condition | * The students/UCs will now have a server which has the optimal performance required. |
| Comments |  |

### Admin Sign In

| Description | Administration will be able to log into the native application for further access and control over the system. |
| --- | --- |
| Stakeholder/Actors | * Admin |
| Pre-Conditions | * The admin has been given an account * The administrator has access to the native client |
| Details/Flow | * The admin opens the native client, displaying the normal login screen. * The admin logs in using their given username and password * If the details are correct, the system logs them in and after a possible loading screen, displays the admin dashboard |
| Goal/Post-Condition | * The admin gains access to the native client * The admin can then control certain areas of the system blocked by other login types |
| Exception | * Invalid email address * Invalid password   In either case, the user will be shown a warning message and will be prompted to try again |
| Comments |  |

### 

### View/edit all assessments

| Description | Admin will be able to view and edit all assessments |
| --- | --- |
| Stakeholder/Actors | * Admin |
| Pre-Conditions | * Admin has an account on the native client |
| Details/Flow | * Admin logs into the native client using their given username and password * The dashboard shown to the admin after logging in displays the options available to the admin, including viewing and editing assessments * If viewing the assessments, once clicking on the view/edit assessments option, the admin will have access to all assessments possibly sorted in a hierarchical system. * If editing assessments, the admin will be able to select the unit or assessment being edited and will be taken to an editing page for that unit/assessment. * On this page, the admin will be able to change assessment details such as times, dates, time length, and the assessment files themselves * Once successfully edited, the admin will click on the ‘save’ button on the page and, if successful, the system will display a success message to the admin |
| Goal/Post-Condition | The Admin has now successfully viewed and/or edited an assessment |
| Exceptions | * The admin selects an invalid date or time |
| Comments |  |

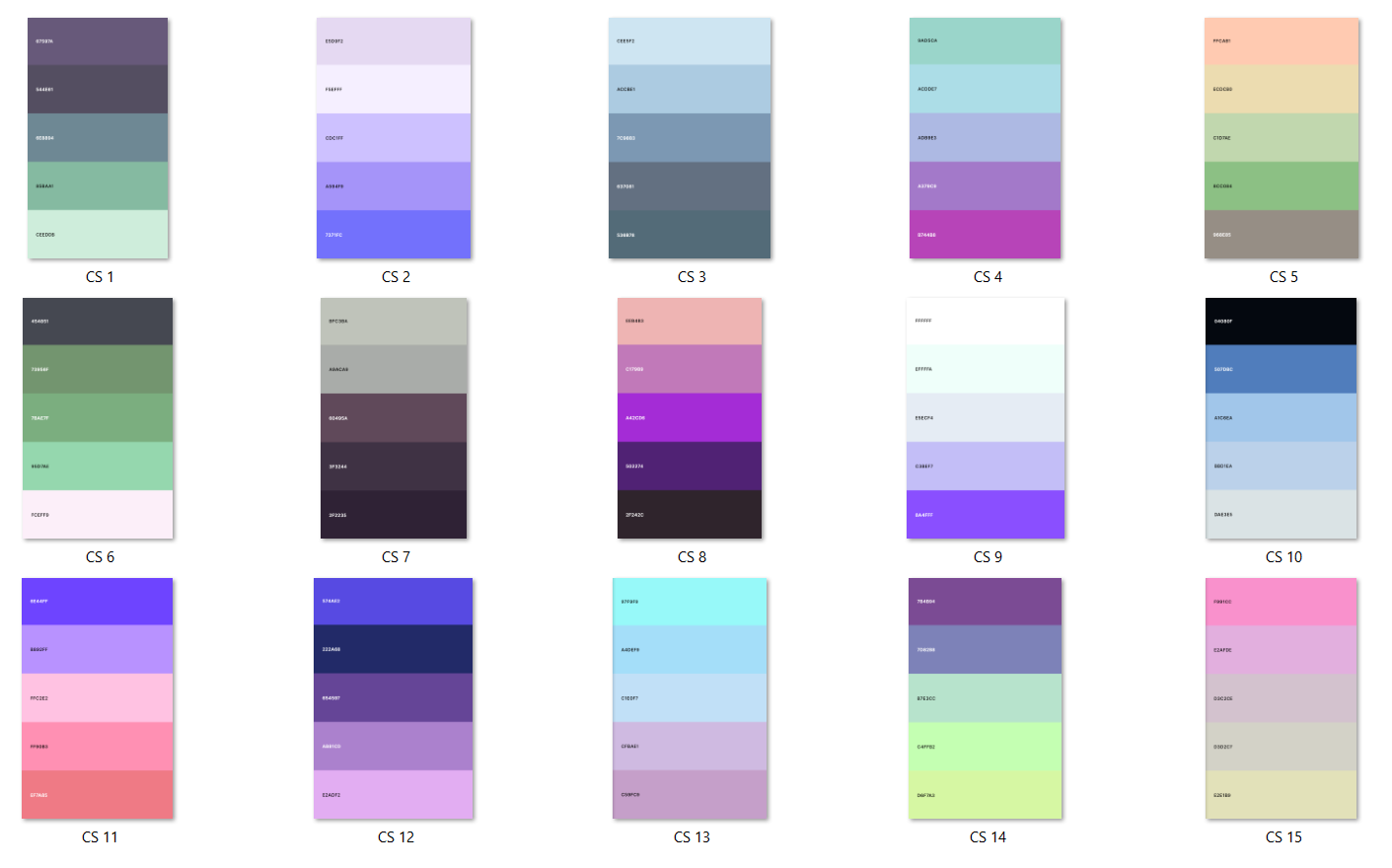
# **External Interface Requirements**

## User Interfaces

This is a list of possible UI designs and themes for some of the pages of the application. These are subject to change as the project develops.

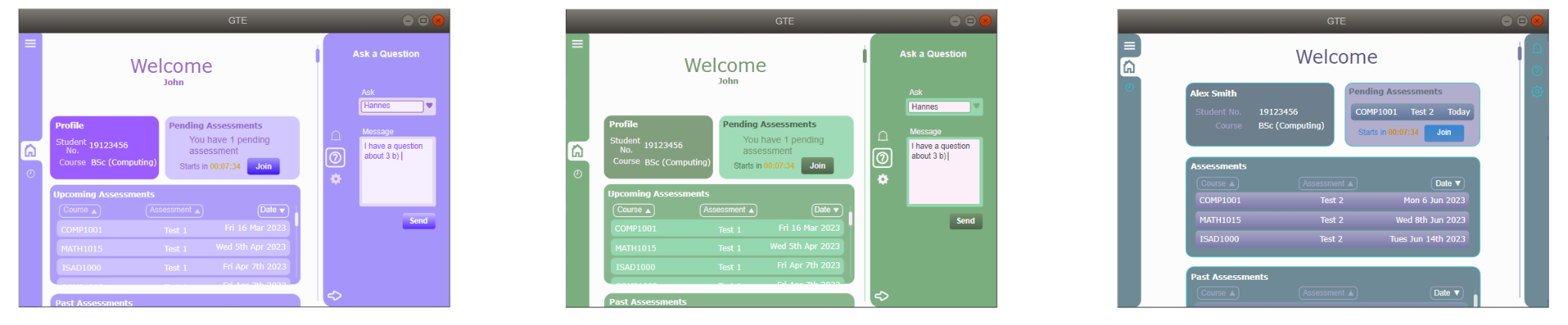
### Colour Schemes

A few different colour schemes were looked at for the prototypes to give the app a more polished feel.

The schemes were first generated using [*https://coolors.co/*](https://coolors.co/).

They were then narrowed down and the tints and shades of each colour were found using[*https://colors.dopely.top/color-toner/*](https://colors.dopely.top/color-toner/).

These were experimented with on one page to see how each scheme may look throughout the rest of the application.



This was the final scheme chosen for the prototypes, and the rest of the pages were then created using it.

However, the scheme is subject to change and was partly chosen to flesh out the mockups.

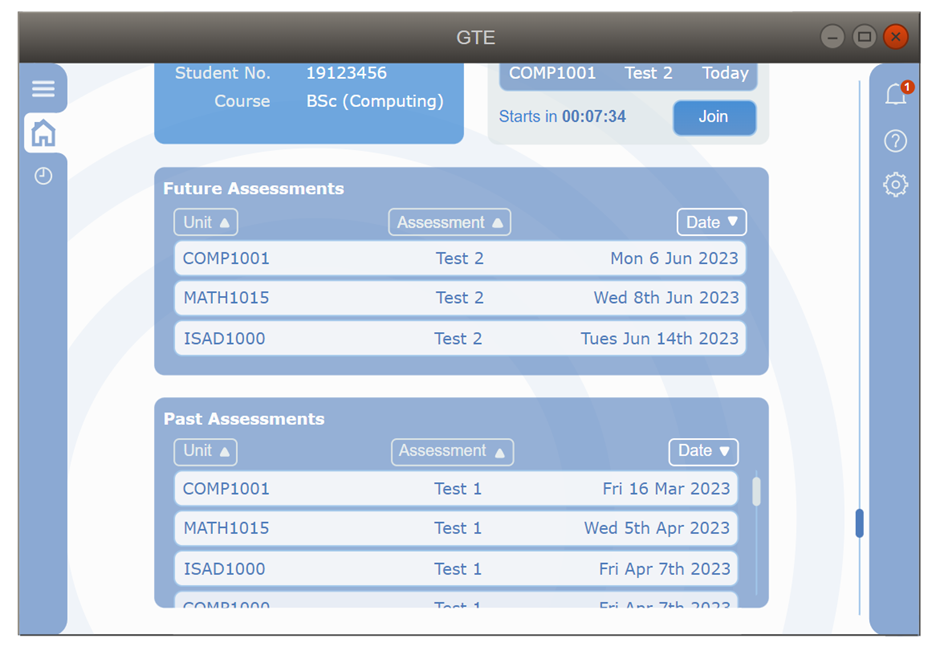
### Student Client

* + - 1. **Dashboard**

The dashboard is the central page from which the students will be able to access most parts of the application and it is from this page that a student will join an assessment

Here, they will also find their account details as well as their assessments; upcoming, past, and future.



The dashboard will be scrollable, showing the rest of the page that lists the student’s future and past assessments.

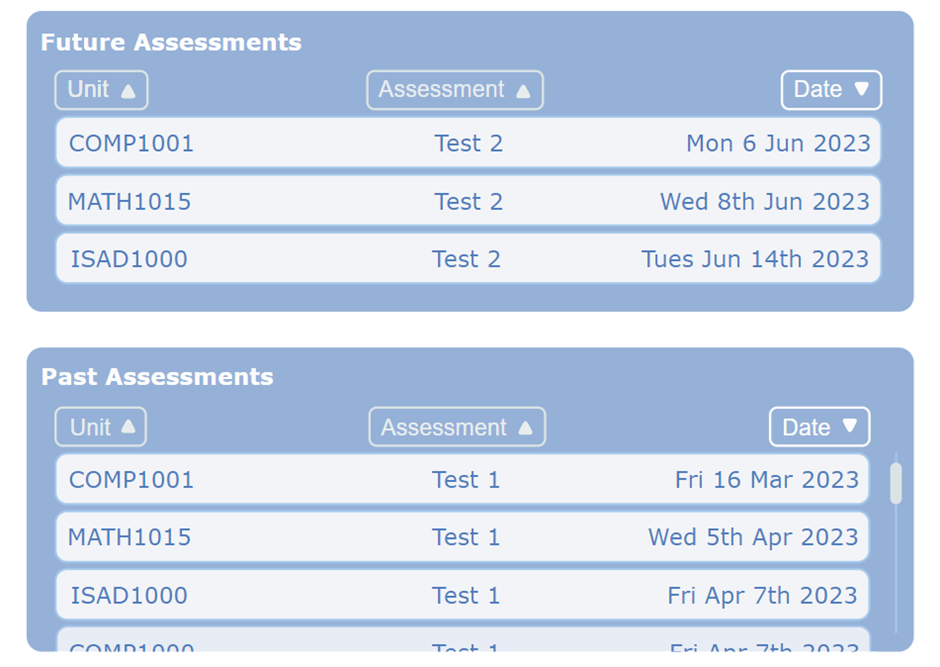
* + - * 1. **Upcoming Assessments**

The ‘Upcoming Assessment’ bubble will change depending on whether the student has an upcoming assessment. If there is an upcoming assessment, the user will be shown the assessment name, when it starts, and a button to quickly join the session.

The element that contains the assessment name will highlight when hovered over and be clickable, taking the user to an information page about that assessment. 

* + - * 1. **Future and Past Assessments**

The ‘Future Assessments’ and ‘Past Assessments’ bubbles will alter according to how many items are in their list, including scroll bars when necessary, and will be sortable by Unit, Assessment Name, or Date.

Each element in the bubbles will highlight when hovered over and be clickable, taking the user to another page with information about that assessment.

* + - 1. **Display Assessment**

Any assessment available to the student will be able to have its information displayed on this page after clicking on the relevant assessment on the dashboard.

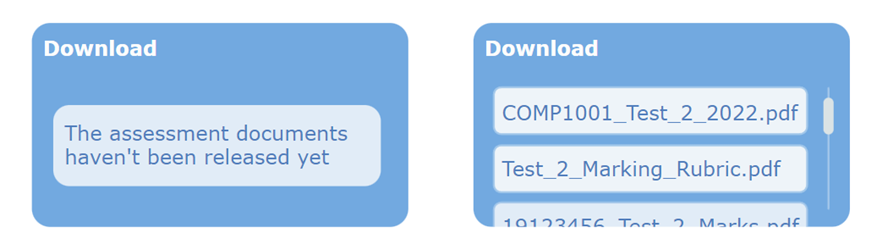
This page will display the date and time of the assessment as well as its duration and location along with any other relevant information.



* + - * 1. **Download Past Assessments**

When available, the download links will be displayed in this bubble.

This may include the assessment paper itself, any relevant marking rubric, and the student’s answer documents.

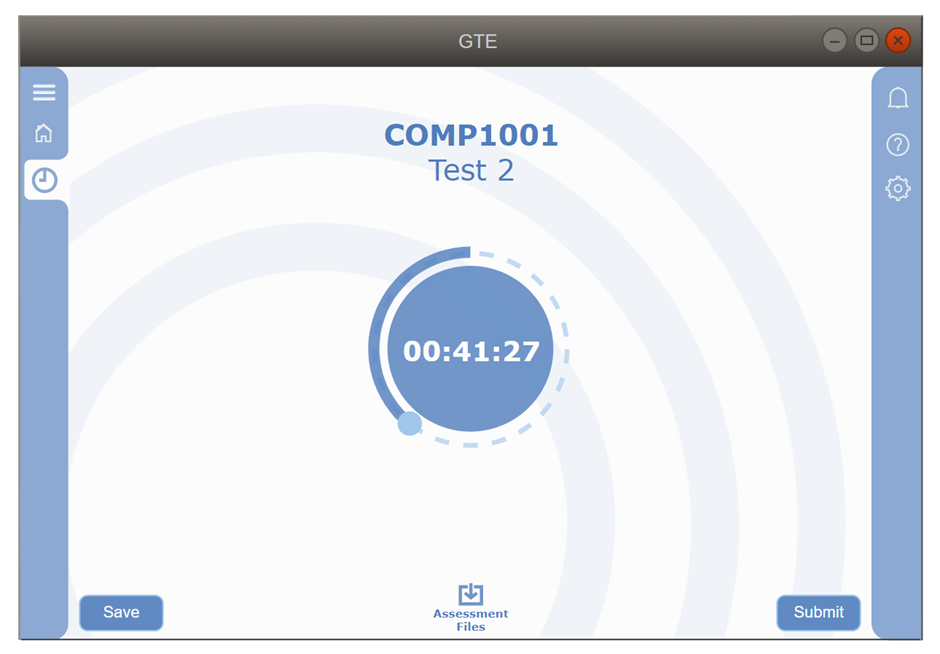
This will be available after a specified amount of time after the assessment is complete or after marking is complete. 

* + - 1. **Current Assessment**

The Current Assessment tab is where the students will spend the duration of an assessment.

It will display the unit’s and assessment’s names, and the remaining time.

The user will also be able to use the save button to manually save their progress or the submit button to do a final save and submission of their working folder.

The assessment files will be able to be downloaded using the link in the centre bottom of the page.

* + - 1. **Tab Bars**
         1. **Navigation Tab Bar**

The left tab bar will be able to be extended and will cause the displayed page to contract to the new window size as shown.

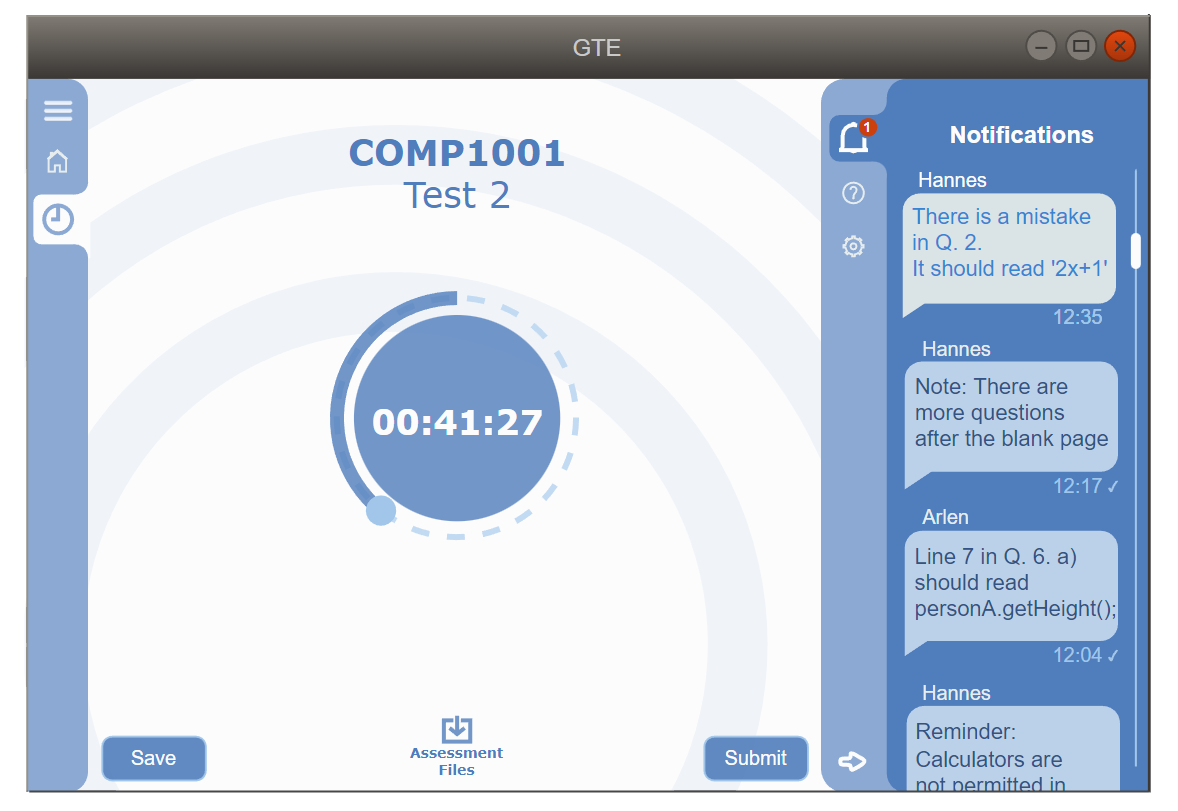
This will possibly be for any additional information, settings, or links that will be displayed here in the future. 

* + - * 1. **Notifications Tab**

The notifications tab will also be accessible from any page during an assessment and will display all incoming messages from the Unit Coordinator or invigilators, including timestamps and read receipts.

When opened, any elements of the currently displayed page will contract to fit into the new view size.

If there is a new notification, before opening the right tab bar, the bell icon on it will display an icon next to it, indicating a new, unchecked notification.



* + - * 1. **Ask a Question Tab**

The ‘Ask a Question’ tab will also be accessible from any page during an assessment and will allow the user to make a query about the current assessment or the assessment environment.

The user will be able to choose to whom they’re asking the question, fill in the message box, and press ‘Send’ to send the message.

* + - * 1. **Settings Tab**

The Settings tab will be accessible from any page at any point and will allow the user to make persistent changes to the application settings.

The user will have access to all user settings including logging out of their account.

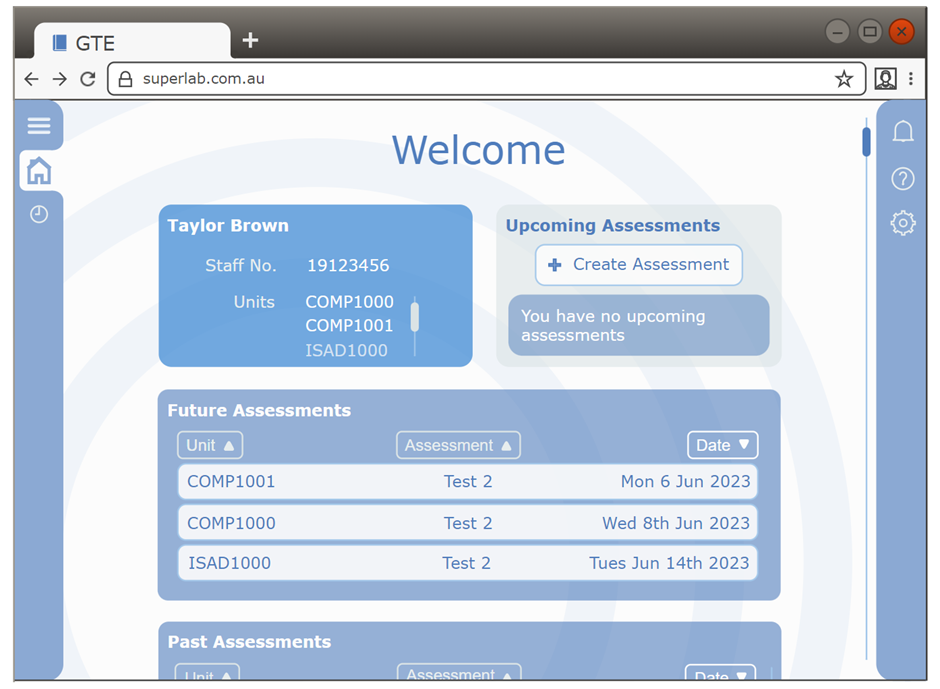
### Invigilator Client

* + - 1. **Dashboard**

The Invigilator Dashboard will be very similar to the student client’s dashboard with a few differences.

The UC login will allow the user to create an assessment using the ‘Create Assessment’ button which will highlight when hovered over and will take the user to a ‘Create Assessment’ page when clicked.

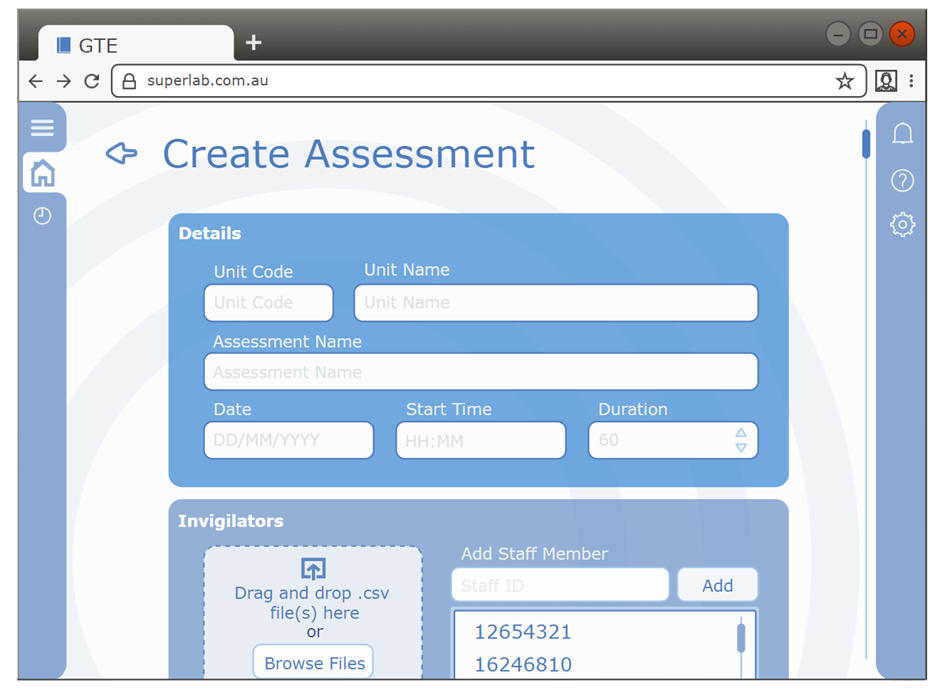
Each element in the ‘Future Assessments’ and ‘Past Assessments’ bubbles will highlight when hovered over and will be clickable, allowing the user to view and/or edit that assessment.



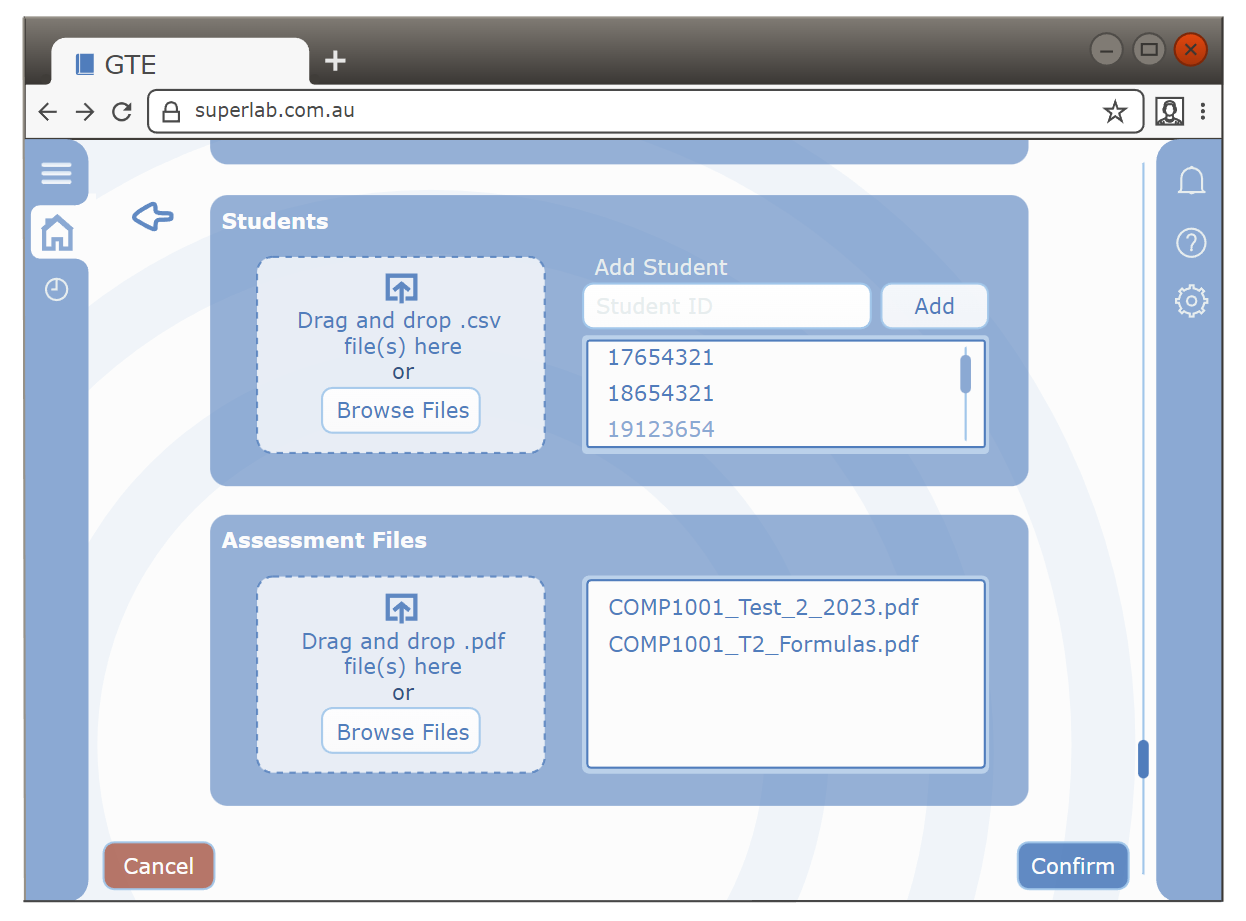
* + - 1. **Create Assessment**

The user will access this page from their dashboard and will use this page to create a new assessment by inserting the relevant details.

The basic information about the assessment will be able to be inserted manually into the respective boxes.



This page will also be scrollable, displaying the rest of the page and the additional information to be input.

The three bubbles for Staff, Students, and Assessment Files will have their information updated using either a drag-and-drop method, a browse button that will allow to user to upload using the native file explorer, and the permission information can be manually added by typing in the student or staff IDs.

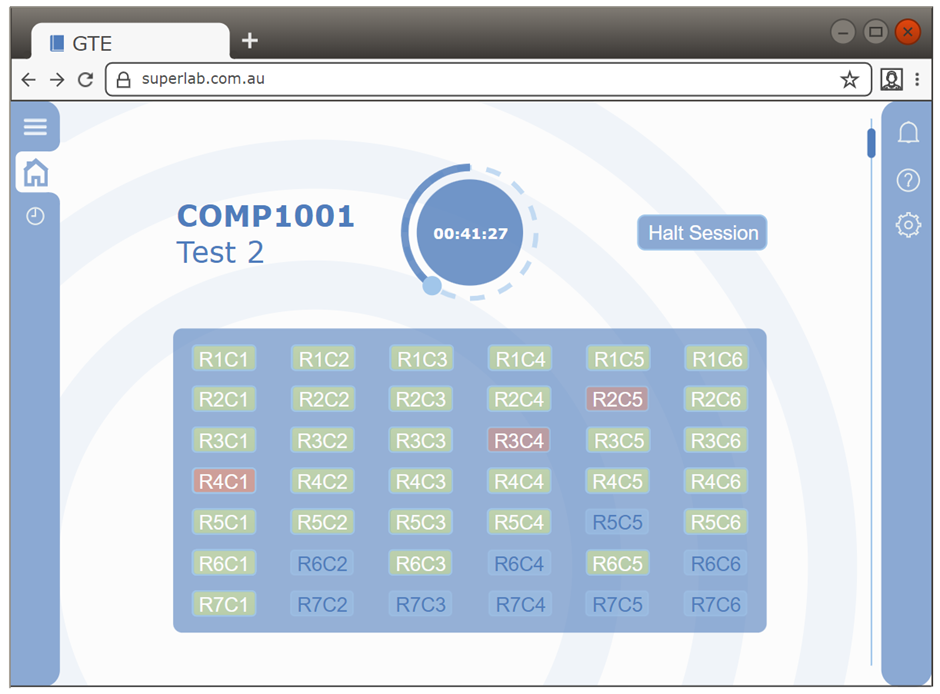
* + - 1. **Current Assessment**

The current assessment page will display the name of the assessment and the time remaining.

It will also display every machine running the same assessment in the native app and its student’s current status.

This will likely be displayed with colours indicating the states of active, asking a question, inactive, or any connection or saving errors.

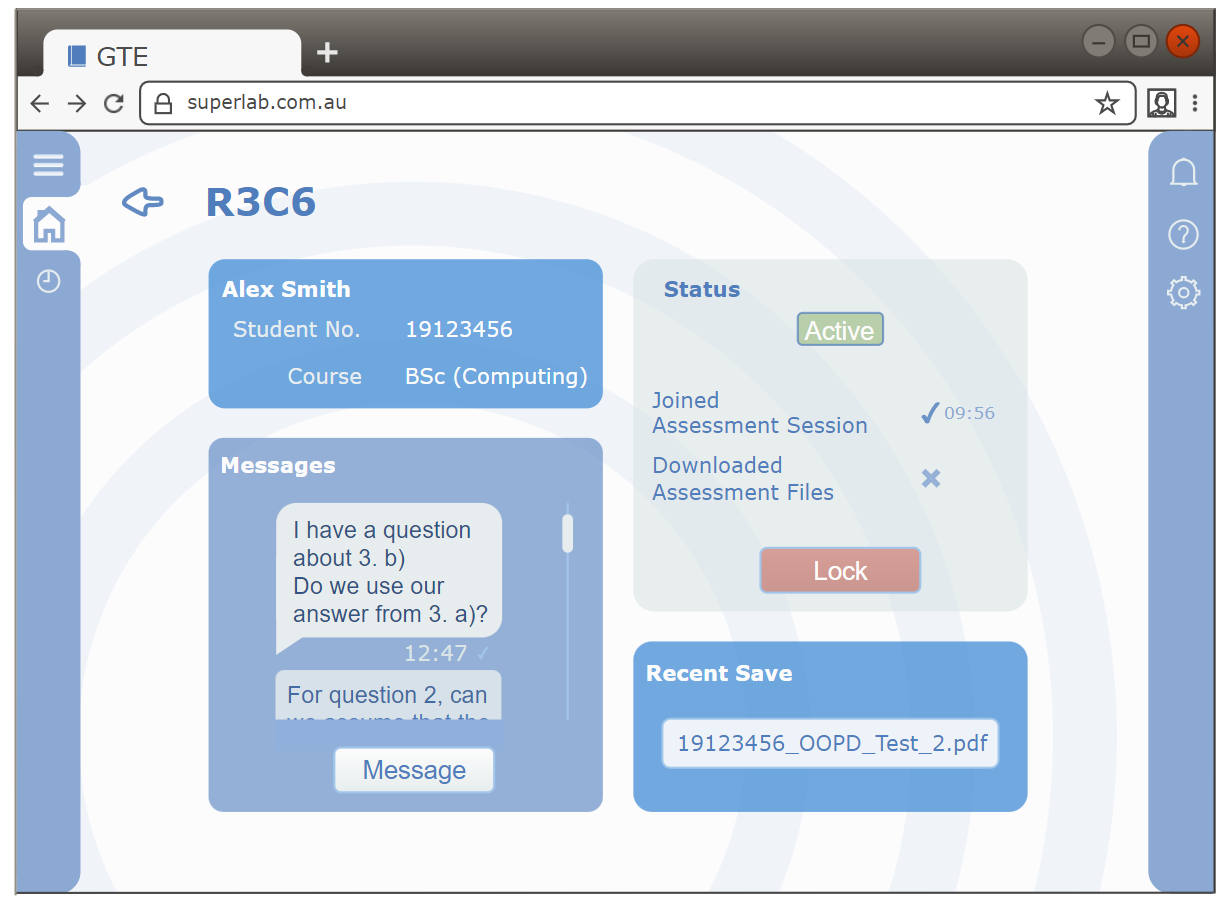
Each element in the student computer list will highlight when hovered over and will be clickable, taking the user to another page with a summary of that student and their current status.

The user will also have access to a ‘Halt Session’ button with which they can halt the session of all students in that assessment.

* + - 1. **Student Profile**

When a student’s computer is clicked on from the ‘Current Assessment’ page, the user will be taken to this page displaying the student using that computer and any relevant information about them.

It will also display the student’s current status and information on how far along the assessment process they are as well as the ability to download the student’s most recent save of their working folder.

From this page, the user will also be able to lock a particular student, halting their session.

## Hardware Interfaces

The invigilator application must provide hardware interfaces to:

* The computers where the application is installed, as this will be a web-based application, it needs to be compatible with mobile phones, tablets and other commonly used non-desktop devices.

The student application must provide hardware interfaces to:

* The computers in the labs where the application is installed.

## Software Interfaces

Both the invigilator and student applications must provide software interfaces to:

* the operating system used by the computers in the labs (in this case, it will be only the Linux distribution defined by the university)
* The software will provide forms for tests creation and scheduling

## Communications Interfaces

The invigilator application must provide communications interfaces to:

* the web browsers that are available on the computers
* the network server used by the invigilators

The student application must provide communications interfaces to:

* The network server used by the students to store their submissions.
* The query messaging platform used to alert invigilators/UCs.

# **Other Nonfunctional Requirements**

## Performance Requirements

### Persistent Error Messages

Error messages should remain displayed until the Notification/Messages window has been closed.

### Help Alert Response Time

The application should send the alert message to the invigilator/unit coordinator within 10 seconds after the help request is sent by the student.

### Ignore Alert Response Time

The application should send the alert message to the student within 10 seconds after the ignore request is sent by the invigilator/unit coordinator.

### Set up Efficiency

A user (Student) should be able to log into and set up their assessment session within 5 mouse clicks.

### Uploading Assessment Documents

A Unit Coordinator/Assessment Supervisor should be able to upload the necessary assessment documents within 3 mouse clicks for each document.

### Folder/Server Backup

The folder should be synchronised with the server every XXX minutes/seconds.

### Coding Standard

The code will adhere to Curtin University’s Coding Standard.

## Safety Requirements

* Data that is stored in the folders provided to the students should be backed up regularly to the server to minimize data loss in case of outages. The regularity should meet the requirement set in 5.1.6

## Security Requirements

The application should provide the utmost security to its users. The following are the requirements:

* Users must be verified via ID log-in which will be done through the initial screen of the application.
* Students will receive a signup email that will link them to a signup paging allowing them to create an account.
* Student accounts will not be able to view any test material prior to the test beginning.
* A strong password policy will be enforced
* Users will be able to quickly recover their password.
* Multiple failed password attempts will be recorded.
* Only invigilators/supervisors will have access to student work folders once the test has been completed.
* Student accounts will not be able to manually configure the test timer apparatus.
* All codes/work in work folders will be secured during and after the test for security and integrity purposes.
* The application code should be obfuscated
* The database should be secured against SQL injection

## Software Quality Attributes

### Robustness

The software must be highly modular in such a way that additional extensions can be integrated in a smooth manner. The client has expressed concerns about having a randomizer script for exam questions to prevent plagiarism and a tokenizer that compares code for possible collusion.

### Maintainability

The software will be written with clean code that will act as future proof to prevent functional obsolescence. This will be done by having consistency in coding components like naming conventions and variable scopes. There will also be comments throughout the code so the client will be able to make easy modifications in the future.

### Availability

The test paper should be made available to the students once the session time has commenced and not before.

### Usability

The usability of the group testing environment should ensure that a minimal amount of the student’s time is compromised with setting up/using the software

# **Other Requirements**

## Documentation

The project must include adequate documentation for students and administrators. This documentation will include information on installation, student use, UC/Invigilator use and how to build the source code.

# 

# **Appendix A: Glossary**

# 

# **Appendix B: Analysis Models**

## Top Level Flow Chart

# 

# **Appendix C: Issues List**

# Appendix D: Attributions

| **Sections** | **Author** | **Contributors** |
| --- | --- | --- |
| Section 1 - 3 | Isaac Ellis | JIA SON Pow |
| Section 2 - 3 | Filipe Lagrenade | Isaac Ellis |
| Section 3 - 4 | Sanjay Francis Williams | Isaac Ellis  Jia Son Pow  Filipe Lagrenade |
| Section 4 - 6 | Jia Son Pow | Isaac Ellis  Filipe Lagrenade  Sanjay Francis Williams |
| Supervisor meeting minutes 1 | Filipe Lagrenade |  |
| Supervisor meeting minutes 2 | Filipe Lagrenade |  |
| Client meeting minutes 1 | Sanjay Francis Williams | Isaac Ellis  Jia Son Pow |
| Client Meeting minutes 2 | Filipe Lagrenade |  |

# 

# 

# Appendix E: Agile Report

## Reflection

Project Status:

|  |
| --- |

The Project is on track and completed by the deadline. The sprint started slow due to less communication being made during study week. In the second week of the sprint the team got together, performed a session of planning poker and delegated tasks.

One thing to improve on would be more small meetings during the week and more meetings with the supervisor/client.

## 

## Group Toggl Report:

## 

## 

## 

## 

## 

## 

## 

## Burndown Chart:

## Meeting Minutes

### I3 Project Supervisor Meeting

Date of meeting – 29/04/2022

Start time – 4:15PM

End time – 4:40PM

Present: Filipe, Isaac, Arlen

Absentees: Sanjay, Jay

Next meeting: TBD

**1.** **Announcements**

-

**2.** **Discussion**

• Arlen brought up that changes in the functional requirements of the SRS need to be made to Functional Requirements – were written as what the system requires to be functional, not what the system will be functionally doing

• Isaac to send Arlen an email to follow up on the CID information

• Arlen suggested to not write the file syncing module from scratch as it would potentially be too time-consuming, suggested to do one of the other options written in the team’s TIS document

• Arlen mentioned the lack of in-text referencing in the team’s research summaries

• Arlen mentioned that there should have been more options explored for the database systems section of the TIS, although a decent choice has been selected

• Arlen mentioned that the hours were a little low this sprint, the team needs to aim for 30 hours each sprint. On the odd occasion, 25+ hours may be accepted. Not doing this will result in marks lost at the end of semester.