Final Handover Document

for

Group Testing Environment

v1.0

25/05/2022

Prepared by

Filipe Lagrenade,

Sanjay Williams,

Jia Son Pow, and

Isaac Ellis

Capstone Group I3

**Table of Contents:**

[**Introduction  
Contributor(s) : Jia Son Pow**](#_uo2pradetltg) **4**

[**Dependency Matrix**](#_2a89wg42p1yp) **5**

[**Dummy Code/Prototypes**](#_2sy5ry0tmxe) **6**

[**Student Client**](#_ybgtvhg96r62) **6**

[Login Page](#_kvq57ymg38aw) 6

[Dashboard](#_x16y9ldfemo2) 7

[Pre-Assessment Information](#_4udiic2ie2td) 8

[Assessment](#_cxppkwdfx8c7) 9

[Appsmith Link](#_nuf8mtp8bxh6) 9

[**UC Client**](#_ovcptpumqob8) **10**

[**Login Page**](#_m7qcvka5leo) **10**

[Dashboard](#_q7aql0uhuqj6) 11

[Create Assessment](#_6panmxpa9ee) 12

[Assessment](#_a7llslkyceex) 13

[Student Information](#_z3g74n8h8j8u) 14

[Appsmith Link](#_fak43rf3pcr3) 14

[**Tickets for Part 2  
Contributor(s) : Isaac Ellis**](#_g4dbmewlv1bx) **15**

[**Handover Plan**](#_jknuksgvzpya) **28**

[**Contributor(s) : Isaac Ellis, Jia Son Pow**](#_sgvqe2og2hmy) **28**

[**Appendix A: Glossary**](#_oyrr3ayu6okf) **29**

[**Appendix B: Analysis Models**](#_a0mv15ixx60n) **30**

[Top Level Flow Chart](#_binm7z9qna1s) 30

[**Appendix C: Issues List**](#_h9yxpjb6motg) **31**

[**Appendix D: Attributions**](#_2l1k61mrv21w) **32**

[**Appendix E: Agile Report**](#_2gj956ys4y3a) **33**

[Reflection  
Author : Jia Son Pow](#_aqbivgh9yeo3) 33

[Group Toggl Report](#_w1kvzcf7tlmm) 34

[Meeting Minutes  
Author - Jia Son Pow](#_37bkau2294e3) 35

# Introduction Contributor(s) : Jia Son Pow

For the final submission of the first half of the capstone project, our main focus will be to showcase some dummy prototypes of our Group Testing Environment clients, one for students and one for invigilators. All of them were designed and implemented on Appsmith, with clickable buttons and mock-up functionalities. These dummy implementations will be discussed in greater detail later on in the final presentation for this semester.

We have also included a list of task tickets that was carefully planned for the second part of this year-long project. The tickets were properly sized by us, having gone through a discussion for each story and ranking the tasks’ level of complexity using an online Scrum planning poker (PlanITpoker).

Last but not least, we have also updated the Software Requirements Specifications for the 3rd and final time in order to tie up any loose ends that we have missed for the last sprint. The main concerns expressed by our supervisor, Arlen Brower, were the functional requirements and use cases. These have since been properly resolved in this document revision.

# 

# 

# 

# 

# Dependency Matrix

**Contributor(s) : Isaac Ellis**

|  | Backend Setup | UC Client Setup | Student Client Setup | Security Implementation | File Syncing | UC Client Implementation | Student Client Implementation | Performance / Health Monitoring |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Backend Setup |  |  |  |  |  |  |  |  |
| UC Client Setup |  |  |  |  |  |  |  |  |
| Student Client Setup |  |  |  |  |  |  |  |  |
| Security Implementation |  |  |  |  |  |  |  |  |
| File Syncing | X |  |  | X |  |  |  |  |
| UC Client Implementation | X | X |  |  |  |  |  |  |
| Student Client Implementation | X |  | X |  | X |  |  |  |
| Performance / Health Monitoring | X |  |  |  |  |  |  |  |

# 

# Dummy Code/Prototypes

**Contributor(s) : Filipe Lagrenade, Isaac Ellis, Sanjay Williams, Jia Son Pow**

# Student Client

## Login Page

The above screen is the login page that all students will see upon launching the application. When the submit button is clicked it will take the student to the dashboard. The reset password link will be sent to their student email.

## Dashboard

The above screen is the dashboard which students will see once they successfully log in. It contains their student information, the assessments they have coming up in the very near future, a list of their future assessments and a list of their past assessments which they have already completed. The side bars also contains buttons which perform the following actions (these side bars will be consistent throughout all screens of the student client):

* Home button - Takes the student back to the dashboard from any screen
* Current assessment button - If the student is currently sitting an assessment it will take them to the assessment screen
* Notification button - The student can click this to see any notifications which have been sent out. It will appear with a red numbered icon next to it if there are new notifications for the student to read
* Question button - This will allow the students to ask questions to the UC/Invigilators during the assessment
* Settings button - This will allow the student to perform 3 functions.
  + Toggle on or off reminder emails that will remind them of an upcoming assessment
  + Update password
  + Log out
* When the join button is clicked it will take the student to the pre-assessment screen

## Pre-Assessment Information

The above screen is the screen the students will see before they start the assessment. It provides them with key information such as the duration, location, date and time, banned materials and notes that the UC wants them to be aware of. The start button will be inaccessible until the assessment has begun. Once the start button is available, clicking it will bring the student to the assessment page.

## Assessment

The above screen is the assessment page which contains the following features:

* An on-screen countdown timer which allows the student to keep track of time
* A save button which will allow the student to force-save their work at any time
* A file button which will take the student to the assessment documents
* A submit button which will end the students session, locking them out of the assessment. When the button is clicked a pop-up will appear asking for the students confirmation and letting them know that they will no longer be able to access the assessment. Once they have provided confirmation they will be taken back to the dashboard.

## Appsmith Link

The below link will give access to a prototype client with limited functionality which was created on Appsmith. <http://20.248.180.233/app/student-client/assessment-628c5a0bd174b4217a3dab75>

# 

# UC Client

## Login Page

The above screen is the login page that all Unit Coordinators will see upon launching the application. When the submit button is clicked it will take the unit coordinator to the dashboard. It will also allow the unit coordinator to reset their password if they have forgotten it. The reset password link will be sent to their staff email address.

## Dashboard



The above screen is the dashboard which the unit coordinator will see upon successful login. It provides the following functionality:

* Allows the unit coordinator to see any upcoming assessments which they are managing
* Allows them to see any future/past assessments
* Contains their staff information and the units they are managing
* Allows them to create new assessments
* The side bars contain the same functionality to the student client.

# 

## Create Assessment

The above screen is the screen where the unit coordinators can set up new assessments. They have to add in the details of the assessment, add in a CSV file of the invigilators and students that will be attending the assessment, and any files that the student will need during the test. This information will be able to be edited prior to the assessment starting.

## Assessment

The above screen is the assessment screen which the unit coordinator will be able to see during an assessment. It allows them to pause the session in case of emergency, as well as view all of the computers which are in use. The green means the student is currently logged in, the red means the student has finished the assessment and white means the computer is inactive. Each button is clickable which will show any information about the student sitting at that computer.

## Student Information

The above screen is the student information screen which will show details about the student sitting at a given computer. It will show their most recent saved file, when they joined the assessment, any messages they have sent and the ability to lock the assessment for that student.

## Appsmith Link

The below link will give access to a prototype client with limited functionality which was created on Appsmith.

<http://20.248.180.233/app/uc-client/login-628ef33cd174b4217a3dab90>

# 

# 

# Tickets for Part 2 Contributor(s) : Isaac Ellis

All estimates are in story points, currently a story point roughly equals an hour of time. The total estimated story points amount to 357.

Our group has 4 members each committing at least 10 hour pers week over a 14 week period; an estimated 560 hours of work can be calculated.

The project is currently in scope for a timely completion with 560 hours of work to complete an estimated 357 hours worth of tickets.

| **Spring boot setup** | |  | **Flyway Setup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-1 | **Tags**:Setup, Backend |  | **Ticket**: I3-2 | **Tags**:Setup, Backend, Database, security |
| **Estimate**:2 | **Depends On**: |  | **Estimate**:5 | **Depends On**: I3-1 |
| Create a basic spring boot project in the monorepo  see <https://start.spring.io/>  create a basic folder struct to fit the the TIS document  this should include the folders   * data * dao * service * controller   the project should also have a meaning full package name <https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html> | |  | add flyway to the springboot project.<https://flywaydb.org/>  <https://www.baeldung.com/database-migrations-with-flyway>  use the pgdump txt file tsplit each table definition into its own flyway migration file  Flyway should run migrations as part of the backends build process | |

| **Setup ORM in spring backend** | |  | **UC client project setup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-3 | **Tags**:Setup, Backend, Database, UC Client |  | **Ticket**: I3-4 | **Tags**:Setup, UC Client |
| **Estimate**:13 | **Depends On**: I3-1 |  | **Estimate**:5 | **Depends On**: |
| add an orm to the spring boot project. A library like hibernate would be good.  annotate the data classes.  <https://www.bezkoder.com/spring-boot-postgresql-example/>  <https://medium.com/analytics-vidhya/spring-boot-accessing-data-with-jpa-hibernate-and-postgresql-af68386363a4>  no need to create the daos, but an example dao might be helpful | |  | -use a tool such as yarn/npm for dependency management  it may be easy to base it off a starter repo such as  -<https://github.com/egoist/vuepack>  -<https://github.com/nuxt-community/electron-template>  -<https://github.com/atom-archive/electron-starter>  create project folder structure, maybe:  /pages  /components  /utils  /services  /css  /routes | |

| **IAM Keycloak setup** | |  | **Springboot logging setup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-5 | **Tags**:Setup, Backend, security |  | **Ticket**: I3-6 | **Tags**:Setup, Backend |
| **Estimate**:8 | **Depends On**: I3-1 |  | **Estimate**:8 | **Depends On**: I3-1 |
| Use spring security and keycloak to set up basic security features in the backend webapp  <https://medium.com/teamarimac/how-to-secure-spring-boot-application-with-keycloak-3ffed7e500a4>  <https://dzone.com/articles/secure-spring-boot-application-with-keycloak>  <https://www.baeldung.com/spring-boot-keycloak>  <https://www.tutorialsbuddy.com/keycloak-secure-spring-boot-rest-api>  then create or document login/logout api | |  | Setup a logging library in the springboot backend.  Something that can be configured by a properties file  [https://docs.spring.io/spring-boot/docs/2.1.18.RELEASE/reference/html/boot-features-logging.html#:~:text=Spring Boot uses Commons Logging,optional file output also available](https://docs.spring.io/spring-boot/docs/2.1.18.RELEASE/reference/html/boot-features-logging.html#:~:text=Spring%20Boot%20uses%20Commons%20Logging,optional%20file%20output%20also%20available).  <https://www.baeldung.com/spring-boot-logging>  the default might be good enough. | |

| **UC Client Basic page routing** | |  | **Add selenium to backend** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-7 | **Tags**:Setup, UC Client |  | **Ticket**: I3-8 | **Tags**:Setup, Backend, UC Client, Student Client |
| **Estimate**:5 | **Depends On**: I3-4 |  | **Estimate**:5 | **Depends On**: I3-1, I3-4 |
| On the uc client we need to setup vue based routing to connect URLs with pages/components  <https://vuejs.org/guide/scaling-up/routing.html>  <https://router.vuejs.org/>  As part of this ticket setup the create header and side bars to be parent components of other pages. So that they are visible regardless of the route | |  | Add selenium to the backend project. This will be used for integration tests for the UC client. Create an example login/logout selenium test  <https://www.baeldung.com/java-selenium-with-junit-and-testng>  <https://www.javacodegeeks.com/2015/03/spring-boot-integration-testing-with-selenium.html>  <https://www.swtestacademy.com/selenium-spring-boot-cucumber-junit5-project/> | |

| **Add exsiting data package to backend** | |  | **Student Client project initialization** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-9 | **Tags**:Setup, Backend |  | **Ticket**: I3-10 | **Tags**:Setup, Student Client |
| **Estimate**:1 | **Depends On**: |  | **Estimate**:5 | **Depends On**: |
| I created basic data package to generate the UML used in the TIS. This should be copied into the backend as a starting basis for our java data model | |  | -use a tool such as yarn/npm for dependency management  -use a build tool such as webpack for packaging the css/js  -setup electron  it may be easy to base it off a starter repo such as  -<https://github.com/microsoft/TypeScript-Vue-Starter>  -<https://github.com/search?q=vue+starter>  create project folder structure, maybe:  /pages  /components  /utils  /css  /routes | |

| **Student Client routing** | |  | **Spring boot email client setup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-11 | **Tags**:Setup, Student Client, Email |  | **Ticket**: I3-12 | **Tags**:Setup, Backend |
| **Estimate**:3 | **Depends On**: I3-10 |  | **Estimate**:8 | **Depends On**: I3-1 |
| On the student client we need to setup vue based routing to connect URLs with pages/components  <https://vuejs.org/guide/scaling-up/routing.html>  <https://router.vuejs.org/> | |  | we need to be able to send emails from the spring backend  <https://www.baeldung.com/spring-email>  for dev purposes, you may need to use a fake email server to send mail such as smtp4dev<https://github.com/rnwood/smtp4dev>  we should also be able to send emails using templates to construct HTML emails  <https://codingnconcepts.com/spring-boot/send-email-with-thymeleaf-template/>  <https://www.baeldung.com/spring-email-templates> | |

| **Spring security role based security** | |  | **Setup integration testing for Student client** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-13 | **Tags**:Setup, Backend, security |  | **Ticket**: I3-14 | **Tags**:Setup, Student Client |
| **Estimate**:5 | **Depends On**: I3-5 |  | **Estimate**:5 | **Depends On**: I3-1, I3-10 |
| Using keycloak and spring security we need to create secure APIs by the user roles  see<https://www.baeldung.com/role-and-privilege-for-spring-security-registration>  <https://www.appsdeveloperblog.com/role-based-access-control-to-rest-api-with-keycloak/>  <https://stackoverflow.com/questions/56214991/enable-role-authentication-with-spring-boot-security-and-keycloak>  <https://wstutorial.com/rest/spring-security-oauth2-keycloak-roles.html> | |  | The student client needs integration testing to set this up we could use something similar to selenium  <https://www.electronjs.org/docs/latest/tutorial/automated-testing>  we could potentially add these tests into the backend next to the UC client selenium tests. This would mean all the integration tests are written in the same language.  They could also be written in js and kept in the student client directory.  there are also tools such as<https://github.com/electron-userland/spectron> | |

| **Student Signup** | |  | **Integrate vcs with youtrack** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-15 | **Tags**:Backend, UC Client security, Email |  | **Ticket**: I3-16 | **Tags**:Setup, devops |
| **Estimate**:13 | **Depends On**: I3-5, I3-10, I3-12 |  | **Estimate**:2 | **Depends On**: |
| This relates to use case 4.1.1,  This will require the use of keycloak , spring security. An API will need to be created, as well as an email template to be sent to students | |  | <https://www.jetbrains.com/help/youtrack/standalone/GitHub-Integration.html>  this will allow us to track merge/pullrequests and use ticket numbers as branches | |

| **master build that builds all modules** | |  | **Run tests as part of pull request** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-17 | **Tags**:devops |  | **Ticket**: I3-18 | **Tags**:devops |
| **Estimate**:8 | **Depends On**: I3-1, I3-4, I3-10 |  | **Estimate**:2 | **Depends On**: I3-1, I3-4, I3-10 |
| there should be a top level maven/gradle build script that correctly build and runs tests for all submodules   * uc client * student client * backend service | |  | The github pull requests should run all tests as part of the pull requests.  This is optional we might not be able to do this for free on github. Otherwise, we can just run the tests ourselves | |

| **create docker file for required services** | |  | **Student sign in** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-19 | **Tags**:devops |  | **Ticket**: I3-20 | **Tags**:Backend, security, Student Client |
| **Estimate**:5 | **Depends On**: |  | **Estimate**:3 | **Depends On**: I3-5, I3-10 |
| Since we have a lot of third party services a docker compose file will be useful.  If we had a build server we could avoid this by hosting services there.  The service we'll likely run   * postgres * pgadmin(for postgres visual access) * email server * grafana * prometheus(for grafana) | |  | Relates to use case 4.1.1  A login and logout API will need to be created with spring security and keycloak | |

| **Student dashboard page** | |  | **student start assessment and assessment page** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-21 | **Tags**:Backend, Student Client |  | **Ticket**: I3-22 | **Tags**:Backend, Student Client |
| **Estimate**:8 | **Depends On**: I3-3, I3-10 |  | **Estimate**:5 | **Depends On**: I3-3, I3-10 |
| Create the dashboard for the student client as per the UI design.  This will require the creation of an API to retrieve a students assessments | |  | This relates to the Usecase 4.1.3  The flow from the dashboard to the assessment page will need to be created. This will require APIs for starting an assessment, and getting assessment documents. It will also need to show banned materials.  A basic assessment page will need to be created in the student client, much of this functionality will be created in a future tickets | |

| **Student ask question to uc invigilator** | |  | **Student Assessment timer** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-23 | **Tags**:Backend, Student Client |  | **Ticket**: I3-24 | **Tags**:Student Client |
| **Estimate**:8 | **Depends On**: I3-22 |  | **Estimate**:3 | **Depends On**: I3-22 |
| this relates to use case 4.1.4  An API will need to be created to post questions to the backend server  A UI to view and send messages will need to be created on the student client.  Note that the UC client functionality to view and receive messages will be created in a future ticket | |  | This ticket relates to use case 4.1.4  This ticket relates to creating the timer UI and making sure it is accurate. | |

| **Student ability to end assessment early** | |  | **Student assessment runs out of time** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-25 | **Tags**:Backend, Student Client |  | **Ticket**: I3-26 | **Tags**:Backend, Student Client |
| **Estimate**:5 | **Depends On**: I3-22, I3-32 |  | **Estimate**:5 | **Depends On**: I3-22 |
| this relates to usecase 4.1.6  This will require an API to end the assessment early. Once a student has ended their assessment early they should not be able to access other APIs related to the ended assessment.  The save functionality will be created in another ticket | |  | This ticket relates to usecase 4.1.7  When a student runs out of time they should not be able to access apis related to their assessment.  The UI should also notify the student and redirect them to the dashboard page of the student client | |

| **Student alerted on 15 & 5 minutes left** | |  | **Student alerted to uc/invigilator notification** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-27 | **Tags**:Backend, Student Client |  | **Ticket**: I3-28 | **Tags**:Backend, Student Client |
| **Estimate**:5 | **Depends On**: I3-22 |  | **Estimate**:5 | **Depends On**: I3-22 |
| this relates to usecase 4.1.8  Students should be notified when 15 and 5 minutes are left on their assessment. This notification should be on the client and via the OS notification system.  15 & 5 minutes left could be detected by the client or initiated by the backend server(via push notification/webhooks) | |  | This relates to usecase 4.1.9.  The student UI needs to display UC/invigilator notification. The client can either poll for updates or use WebSockets/push notifications to retrieve updates from the backend. | |

| **Student the ability to change computers during an assessment** | |  | **Student password reset** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-29 | **Tags**:Backend, Student Client |  | **Ticket**: I3-30 | **Tags**:Backend, security, Student Client |
| **Estimate**:13 | **Depends On**: I3-31 |  | **Estimate**:5 | **Depends On**: I3-12, I3-20 |
| This relates to usecase 4.1.10  Once the project has working rsync/save/assessment sitting capabilities we need to adequately test and refactor code(if necessary) so that this requirement is met | |  | this relates to usecase 4.1.11  This will require an API and a UI to be built.  The password reset email that is sent should be presentable making use of the email template engine.  The email should send a random token, (created by keycloak maybe or we can create it)  This token will then be entered on the client to verify the user before they submit their new password via the student client UI | |

| **Student assessment file syncing** | |  | **student ability to force save of a work folder** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-31 | **Tags**:Backend, security, Student Client, Rsync |  | **Ticket**: I3-32 | **Tags**:Backend, Student Client, Rsync |
| **Estimate**:34 | **Depends On**: I3-22 |  | **Estimate**:8 | **Depends On**: I3-31 |
| This relates to usecase 4.1.12  This will require the backend and the front end to handle rsync process.  This functionality should be extensively tested for security/performance/scalability | |  | this relates to usecase 4.1.13  This functionality should ideally be separate from the rsync process.  The first step could be to compare a checksum between the student work folder and the backend rsync work folder. If there's a difference a force save should be used.  This process must not interrupt or break the rsync process.  We should extensively test this functionality | |

| **student ability to view files after a test** | |  | **UC signup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-33 | **Tags**:Backend, Student Client |  | **Ticket**: I3-34 | **Tags**:Backend, UC Client, security, Email |
| **Estimate**:5 | **Depends On**: I3-21 |  | **Estimate**:13 | **Depends On**: I3-4, I3-5, I3-12 |
| This ticket relates to use case 4.1.14  This usecase might have changed our client notes and should be consulted before editing | |  | This relates to usecase 4.2.1  Signup should be for invigilators as well  This will likely involve creating forms on the UC clients, backend APIs, keycloak integration and making a presentable email template(using the template engine) to be sent to the UC for signup purposes | |

| **UC client dashboard** | |  | **UC active assessment page** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-35 | **Tags**:Backend, UC Client |  | **Ticket**: I3-36 | **Tags**:Backend, UC Client |
| **Estimate**:13 | **Depends On**: I3-3. I3-4 |  | **Estimate**:8 | **Depends On**: I3-3, I3-4 |
| This ticket will require a dashboard UI to be created based on the UI diagrams.  To contain a list of assessments and a create assessment button | |  | A UI for the active assessment needs to be created for an active assessment. Some APIs will likely need to be created.  Much of the functionality of the page will be created in future tickets, such as adding time, sending notifications, receiving messages | |

| **UC Test Time Alteration** | |  | **Password reset UC/Invigilator** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-37 | **Tags**:Backend, UC Client, Student Client |  | **Ticket**: I3-38 | **Tags**:Backend, UC Client |
| **Estimate**:5 | **Depends On**: I3-36 |  | **Estimate**:3 | **Depends On**: I3-12, I3-45 |
| relates to usecase 4.2.2  An interface to send/view alterations needs to be created, APIs to support this.  New alterations should update on the student client. (push notifications/webhooks) might be a good fit for this. | |  | This ticket relates to 4.2.3  Similar to 13-30  An interface for the UC client will need to be created. | |

| **UC Assessment Creation** | |  | **UC Sending Alerts** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-39 | **Tags**:Backend, UC Client |  | **Ticket**: I3-40 | **Tags**:Backend, UC Client |
| **Estimate**:13 | **Depends On**: I3-56 |  | **Estimate**:8 | **Depends On**: I3-36 |
| this relates to usecase 4.2.4  This will need a UI and supporting apis | |  | relates to usecase 4.2.5  This requires a UI/API to view and send alerts. | |

| **UC View Student Messages** | |  | **UC Download Test Files (One Student at a time / All at once)** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-41 | **Tags**:Backend, UC Client |  | **Ticket**: I3-42 | **Tags**:Backend, UC Client |
| **Estimate**:8 | **Depends On**: I3-23, I3-36 |  | **Estimate**:8 | **Depends On**: I3-35 |
| This relates to usecase 4.2.6  A UI/API will need to be created to support this.  Functionality to mark student messages as resolved should be created.  Other UC clients should be notified so that they can see that message is resolved. | |  | this relates to 4.2.7  this will requires changes to the UI and the creation of APIs | |

| **UC Create notes during the test** | |  | **UC/Admin to View/Edit upcoming assessment** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-43 | **Tags**:Backend, UC Client |  | **Ticket**: I3-44 | **Tags**:Backend, UC Client |
| **Estimate**:8 | **Depends On**: I3-35, I3-36 |  | **Estimate**:8 | **Depends On**: I3-35 |
| This relates to usecase 4.2.8  This will require a UI and APIs to be built | |  | This relates to usecase 4.2.9, 4.3.4  This will require a UI and APIs to support it | |

| **UC/ADMIN login** | |  | **Grafana setup** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-45 | **Tags**:Backend, UC Client, security |  | **Ticket**: I3-46 | **Tags**:Setup, Backend, Performance |
| **Estimate**:8 | **Depends On**: I3-4, I3-5 |  | **Estimate**:8 | **Depends On**: I3-1 |
| This relates to usecase 4.3.2  UC, invigilators and Admins need to be able to login into the client  This will require keycloak integration, APIs, and a UI | |  | This relates to 4.3.1  See<https://ordina-jworks.github.io/monitoring/2020/11/16/monitoring-spring-prometheus-grafana.html>  <https://medium.com/javarevisited/springboot-app-monitoring-with-grafana-prometheus-7c723f0dec15>  <https://www.techgeeknext.com/spring-boot/spring-boot-actuator-prometheus-grafana>  An existing grafana/spring board could be used  <https://grafana.com/grafana/dashboards/6756>  <https://grafana.com/grafana/dashboards/12464>  there are many other boards | |

| **Rsync Grafana Monitoring** | |  | **User Observability for Grafana** | |
| --- | --- | --- | --- | --- |
| **Ticket**: I3-47 | **Tags**:Backend, Rsync, Performance |  | **Ticket**: I3-48 | **Tags**:Backend, Performance |
| **Estimate**:13 | **Depends On**: I3-31, I3-46 |  | **Estimate**:13 | **Depends On**: I3-5, I3-46 |
| The backends use of rsync should be made monitorable using micrometer. This can be then be used for monitoring by the grafana dashboard  see<https://autsoft.net/defining-custom-metrics-in-a-spring-boot-application-using-micrometer/>  <https://www.tutorialworks.com/spring-boot-prometheus-micrometer/> | |  | Similar to I3-47  Observability using micrometer should be added to the spring boot app. This should then be used to create observable metrics in grafan  such as number of user logged in  number of UCs logged in | |

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Handover Plan

# Contributor(s) : Isaac Ellis, Jia Son Pow

* The UC client will be the first application that we will focus on installing as it is the first vital juncture of the invigilation system. The application will first be installed on the school’s backend server with admin permissions.
* The student client will be the first point of contact to the system for the students. This will be installed on every single computer that is used in the SuperLab which is scheduled for completion of construction by the end of the year.
* We will also be providing installation instructions for both clients should there be any potential security concerns so that the admin/staff members can do it themselves. A guide on how to reboot/restart the system will also be provided alongside.
* There will be a list of service documents (pgAdmin, Keycloak, Grafana) provided in order to update these services. The admin credentials to access these services will be provided as well.
* The git repository that we have set up for this project will also be available publicly for ease of access on our documentation for the project. All tickets planned on YouTrack will also be provided.

# 

# **Appendix A: Glossary**

N/A

# 

# **Appendix B: Analysis Models**

## Top Level Flow Chart

# 

# **Appendix C: Issues List**

N/A

# Appendix D: Attributions

| **Sections** | **Author** | **Contributors** |
| --- | --- | --- |
| Introduction | Jia Son Pow |  |
| Dependency Matrix | Isaac Ellis |  |
| UI/Dummy Prototypes | Filipe Lagrenade | Isaac Ellis, Sanjay Williams |
| Tickets for Part 2 | Isaac Ellis |  |
| Handover Plan | Jia Son Pow | Isaac Ellis |
| SRS revision | Sanjay Williams | Jia Son Pow |

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Appendix E: Agile Report

## Reflection Author : Jia Son Pow

For this final sprint that spanned for almost 3 weeks long, the team was able to organise themselves very quickly which ensured that the final version of the SRS was able to be completed by the end of the final teaching week. This allowed the team to discuss any issues with the client beforehand, get feedback from the supervisor on any changes which should get made and implement said changes on time.

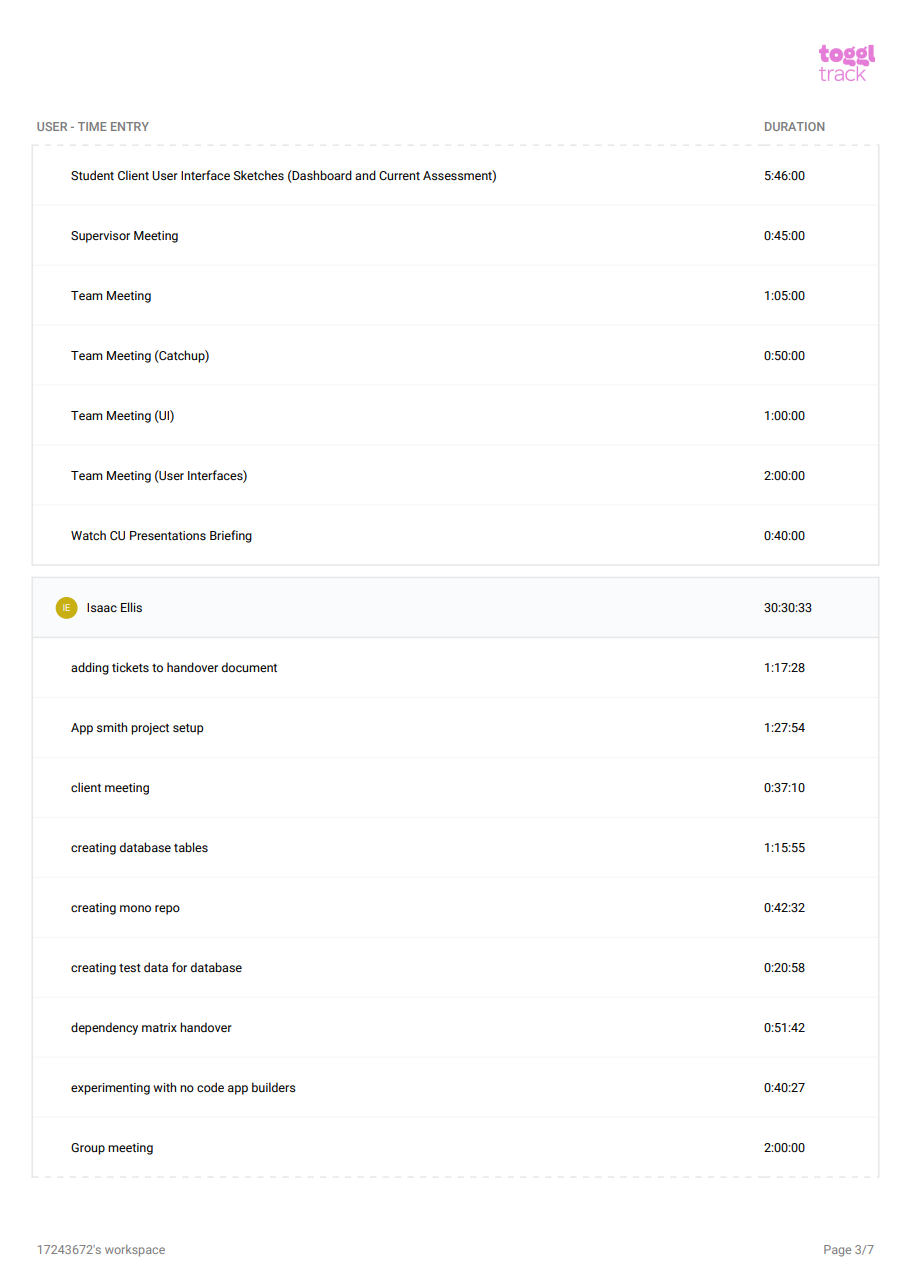
This meant that within the time frame available to this sprint, the team was able to adjust and work accordingly to produce a strong final submission document which will be a strong foundation to the project when we move to the second phase of the project next semester.

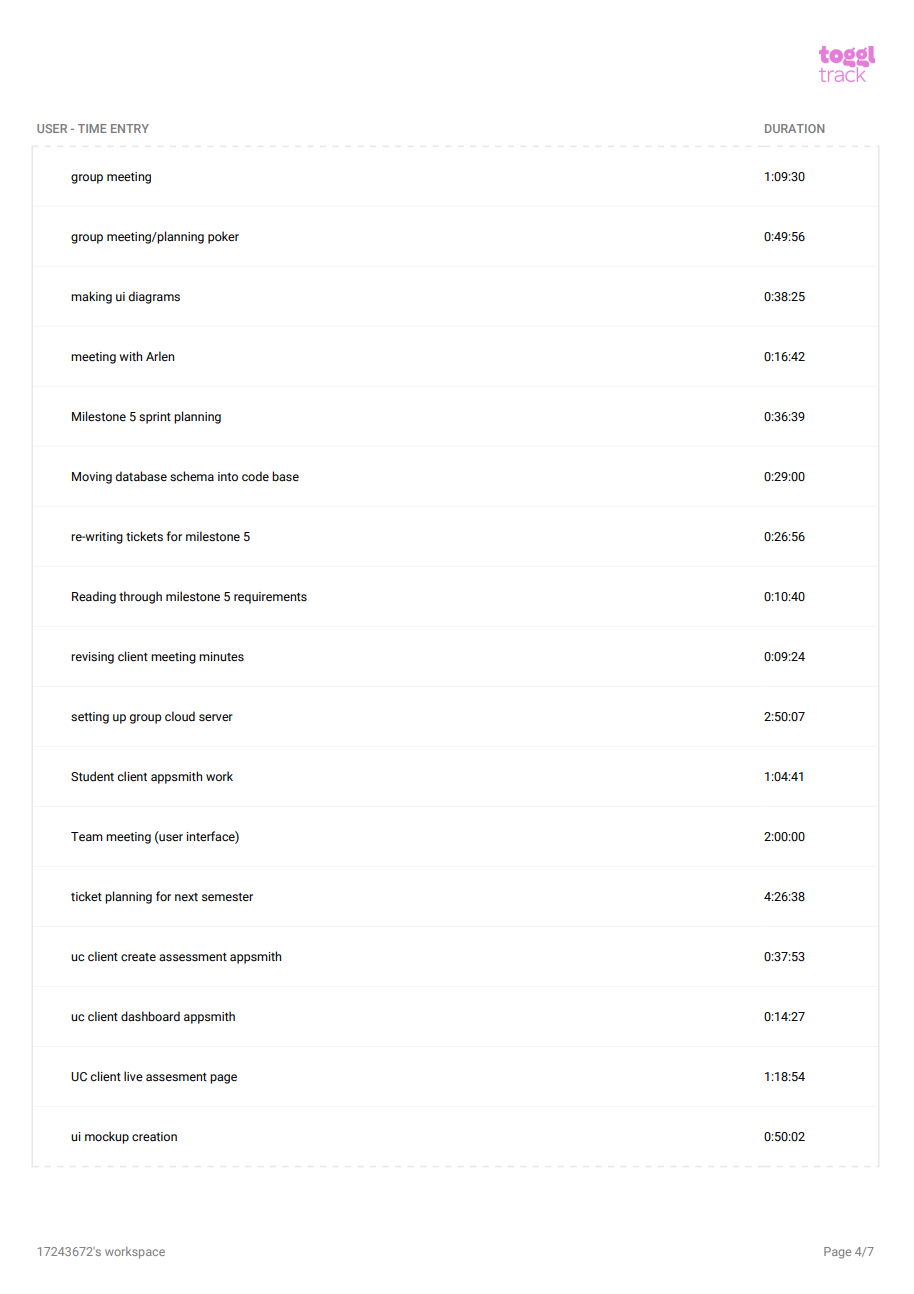
## 

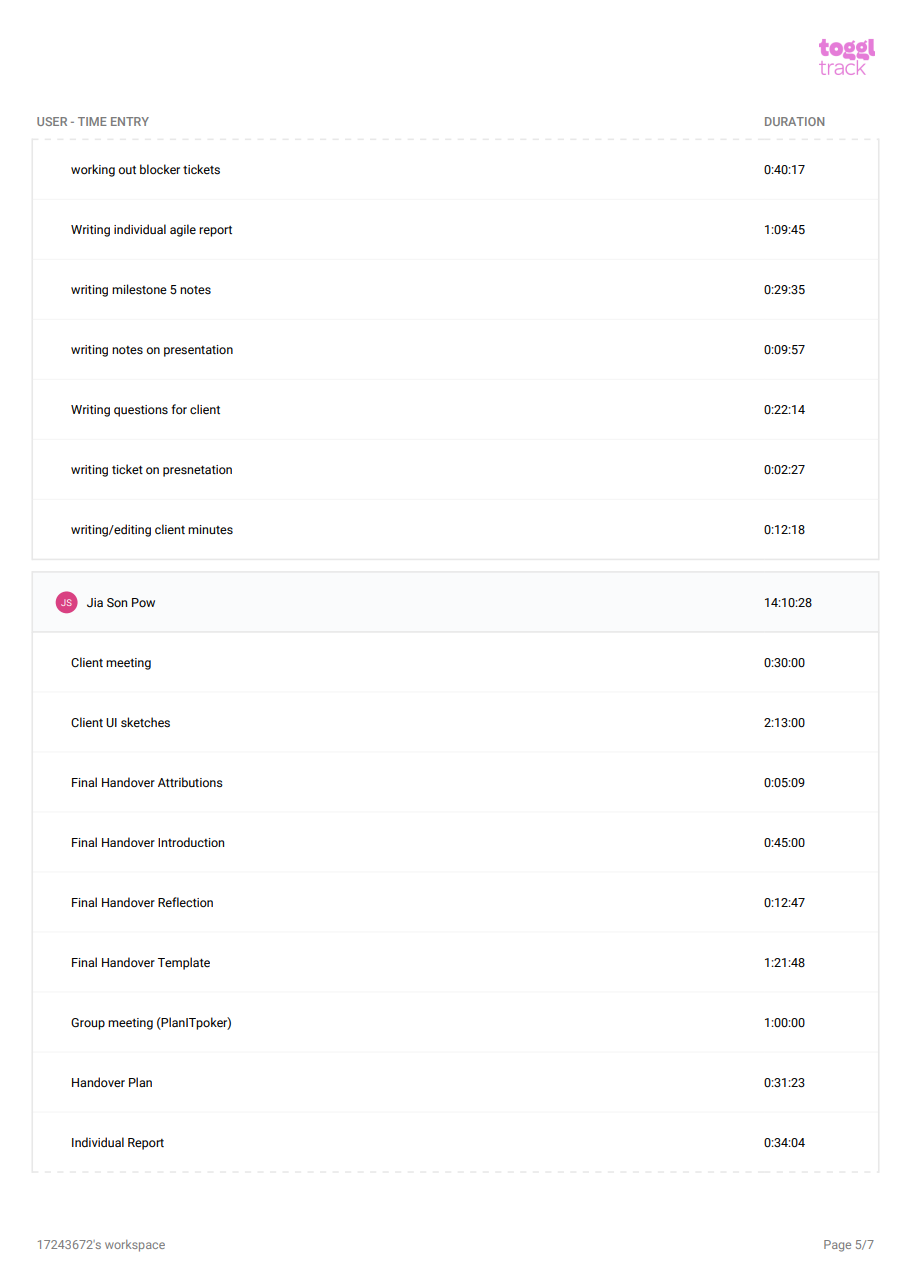
## Group Toggl Report

## 

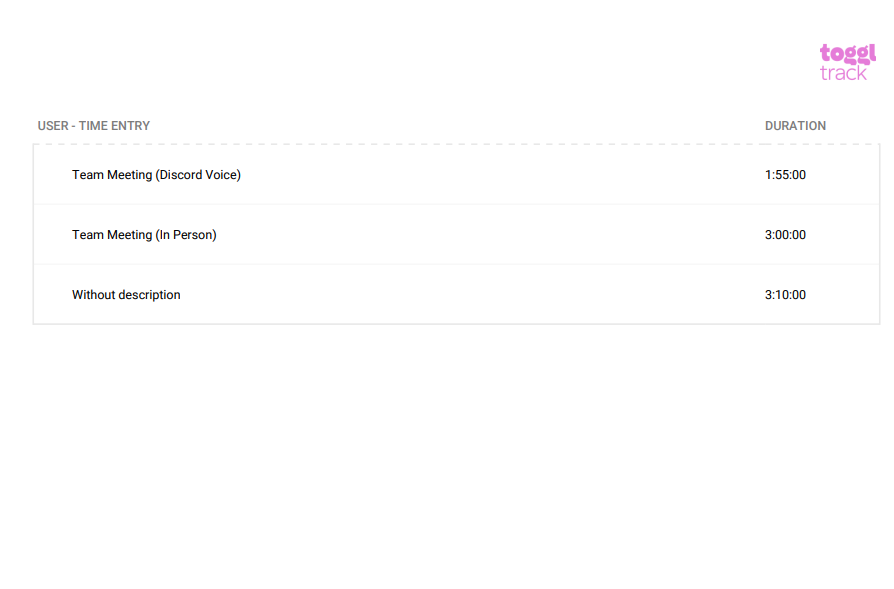
## 







## 



## Burndown Chart

## Contributor: Sanjay Williams

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## Meeting Minutes Author - Jia Son Pow

