# School of Computer Science and Informatics

# Coursework Submission Cover Sheet

***Please use Adobe Reader to complete this form. Other applications may cause incompatibility issues.***

---

| | |
|---|---|
| Student Number | C1531326 |
| Module Code | CMT303 |
| Submission Date | 04/04/2019 |
| Hours spent on this exercise | 60 |
| Special Provision | |

(Please place an x in the box above if you have provided appropriate evidence of need to the Disability & Dyslexia Service and have requested this adjustment).

---

**Group Submission**

For group submissions, *each member of the group must submit a copy of the coversheet.* Please include the student number of the group member tasked with submitting the assignment.

| | |
|---|---|
| Student number of submitting group member | C1865452 |

***By submitting this cover sheet you are confirming that the submission has been checked, and that the submitted files are final and complete.***

---

**Declaration**

***By submitting this cover sheet you are accepting the terms of the following declaration.***

I hereby declare that the attached submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings.

# CMT303 – Software Engineering
# Team Portfolio B
# Group: 4
# Group Members:

*Armando Castany*
*Hannah Rosser*
*Qifa Cai*
*Sicheng Du*
*Wang Chaoran*
*Zak Kyriakou*

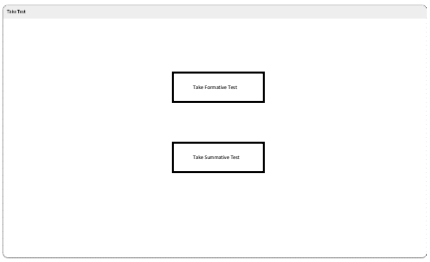
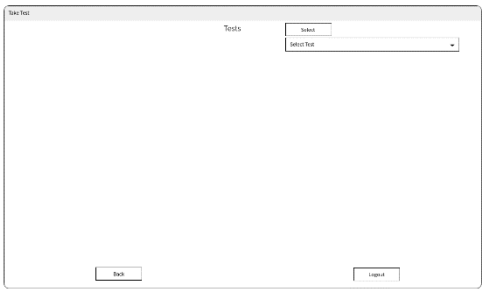
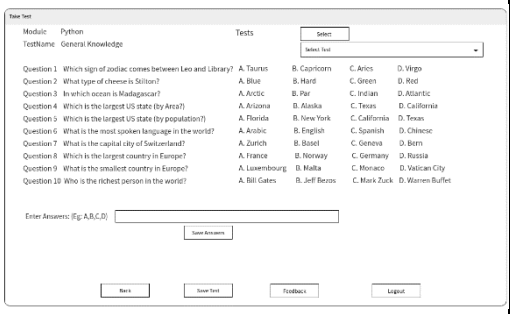
## Contents



**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

# Test Cases:

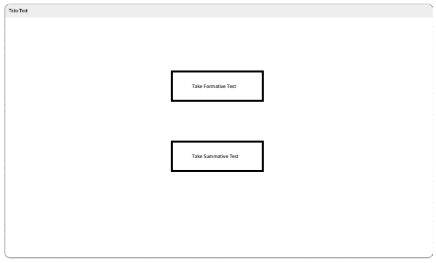
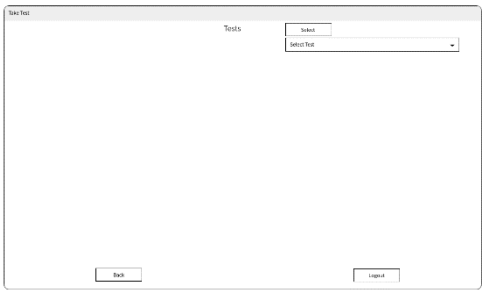
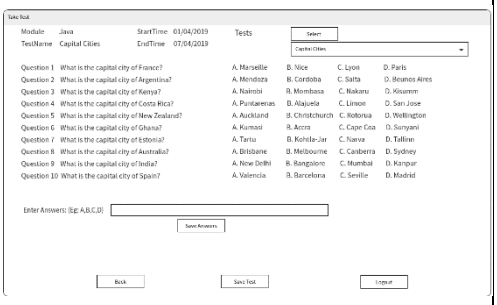
## 1 – Take Formative Test

| **Test Case Id: 1** | **Test Purpose: Test if 'Take Formative Tests' works as intended** | | |
|---|---|---|---|
| **Environment: written in Python v3.7 running under Windows10** | | | |
| **Preconditions: Student logged in and clicks 'take tests'** | | | |
| **Test Case Steps: Basic Flow** | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click Take Formative Test Button<br>  <br> | The test type selection is loaded.<br>  <br> | |
| 2 | Click the Normal Take Button | Screen changes to 'take formative test' screen (shown below)<br>  <br> | |
| 3 | Select the 'General Knowledge' test from the drop-down box in the top right hand corner and click the 'select' button above | The test questions for the General Knowledge test are loaded to the screen and should look like the interface sketch below<br>  <br> | |
| 4 | Type the answer "A,B,C,D,A,B,C,D,A,B" in the | No visual response at this point. | |

**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

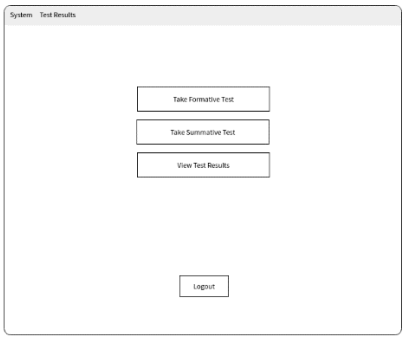
| | | | |
|---|---|---|---|
| | answer box below the questions and select “Save Answers”. | | |
| 5 | Select ‘Save Test’ button | Confirmation box pops up which displays ‘Test has been submitted’. | |
| 6 | Select ‘Logout’ button | The interface closes. | |
| **Comments:** | | | |
| **Related Tests:** Take summative tests (2), View Test Results (3) | | | |
| **Author: Chaoran Wang** | | **Checker:** | |

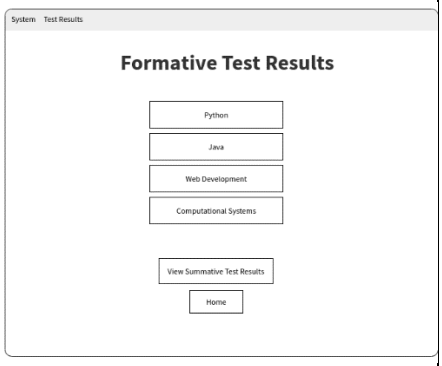
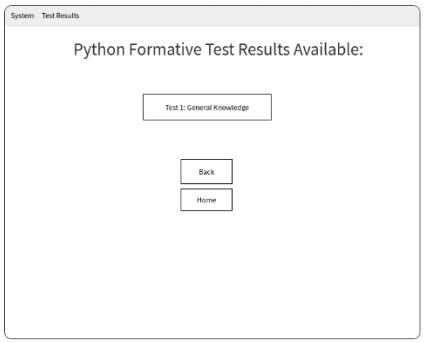
## 2 – Take Summative Test

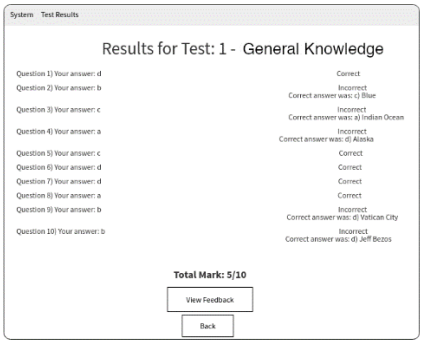
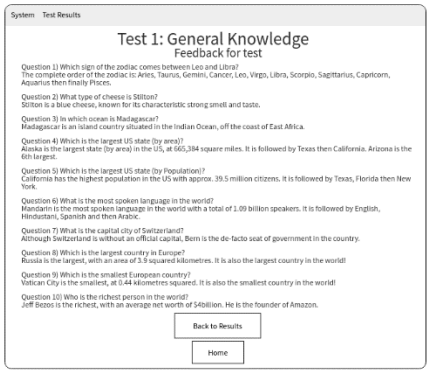
| **Test Case Id: 2** | **Test Purpose: Test if ‘Take Summative Tests’ works as intended** | | |
|---|---|---|---|
| **Environment: written in Python v3.7 running under Windows10** | | | |
| **Preconditions: Student logged in and clicks ‘take tests’** | | | |
| **Test Case Steps: Basic Flow** | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click Take Summative Test Button  | Screen changes to ‘take summative test’ screen (shown below)  | |
| 2 | Select the ‘Capital Cities’ test from the drop-down box in the top right-hand corner and click the select button above | The test questions for the Capital Cities test are loaded to the screen and should look like the interface sketch below  | |
| 3 | Type the answer “A,B,C,D,A,B,C,D,A,B” in the | No visual response at this point. | |

| | | | |
|---|---|---|---|
| | answer box below the questions and select “Save Answers”. | | |
| 4 | Select ‘Save Test’ button | Confirmation box pops up which displays ‘Test has been submitted’. | |
| 5 | Select ‘Logout’ button | The interface closes. | |
| **Comments:** | | | |
| **Related Tests:** Take formative tests (1), View Test Results (3) | | | |
| **Author: Zak Kyriakou** | | **Checker:** | |

## 3a – Student Formative Results

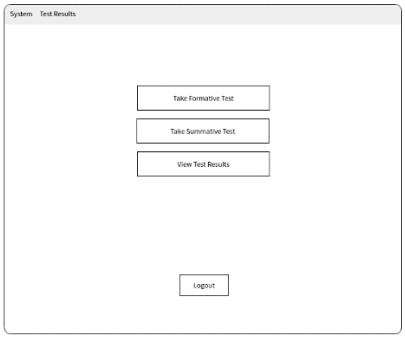
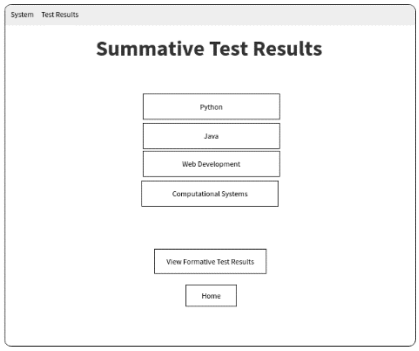
| | | | |
|---|---|---|---|
| **Test Case Id:** 3a | **Test Purpose:** Student can see their Formative Test Results & Feedback | | |
| **Environment:** MacBook Pro (2016) using Python 3.7 | | | |
| **Preconditions:** Student has taken the Formative Test and has logged into the system | | | |
| **Test Case Steps:** | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Home Page (starting place) | Displays the system home page with the menu and buttons.   | |
| 2 | Select to “View test results” | Displays 2 options for viewing test results:<br>“Formative” and “Summative” | |

| | | | |
|---|---|---|---|
| | |  | |
| 3 | Select “Formative” | Displays the student’s modules<br>  | |
| 4 | Select “Python” | Displays the tests taken, in the selected subject, that are available to view the results for<br>  | |
| 5 | Select “Test 1: General Knowledge” | Displays the student’s results for that test. Displays which questions they got correct/incorrect and their total mark. There is a button that links to the Feedback. | |

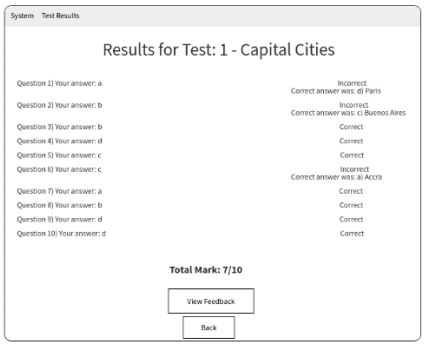
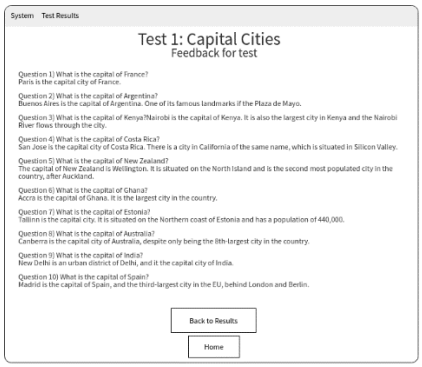
| | | | |
|---|---|---|---|
| | |  | |
| 6 | Select “View Feedback” | Displays the test questions and general feedback for the reasoning behind each correct answer. You can then click ‘Home’ to return to the starting page.<br>  | |
| **Comments:** | | | |
| **Related Tests:** | | | |
| **Author: Hannah Rosser** | | **Checker:** | |

## 3b – Student Summative Results

| | |
|---|---|
| **Test Case Id:** 3b | **Test Purpose:** Student can see their Summative Test Results & Feedback, and the average mark of the class |
| **Environment:** MacBook Pro (2016) using Python 3.7 | |
| **Preconditions:**<br>Student has taken the Summative Test<br>The deadline for the Summative Test has passed<br>The marks and feedback for the Summative Test have been released<br>The student is logged into the system | |

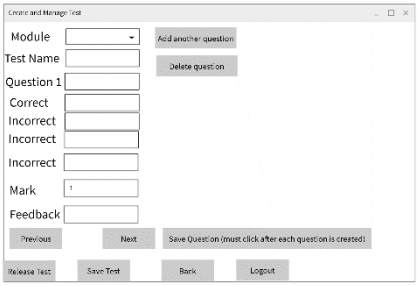
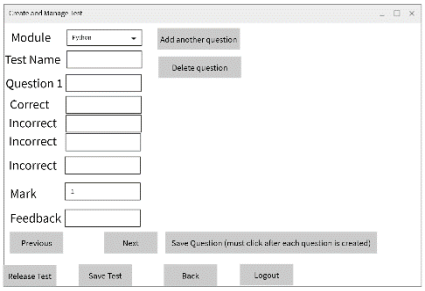
| **Test Case Steps:** | | | |
|---|---|---|---|
| Step No | Procedure | Response | Pass/Fail |
| 1 | Home Page (starting place) | Displays the system home page with the menu and buttons.   | |
| 2 | Select to “View test results” | Displays 2 options: “Formative” and “Summative”   | |
| 3 | Select “Summative” | Displays the student’s modules   | |

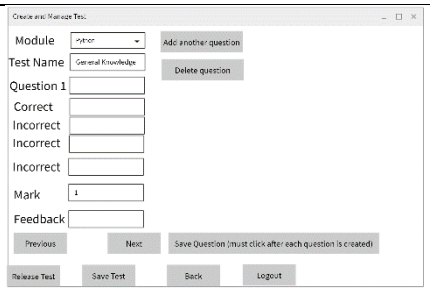
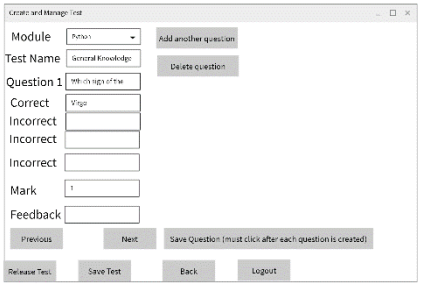
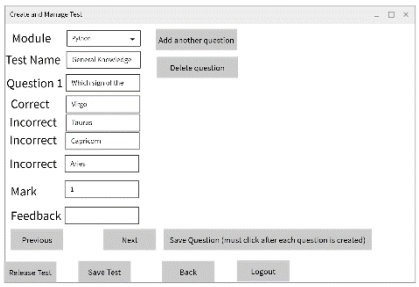
**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

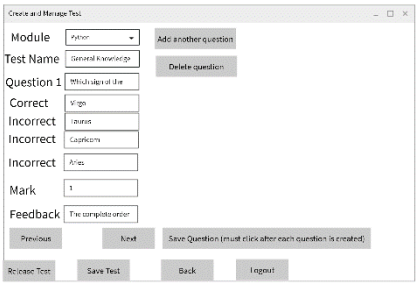
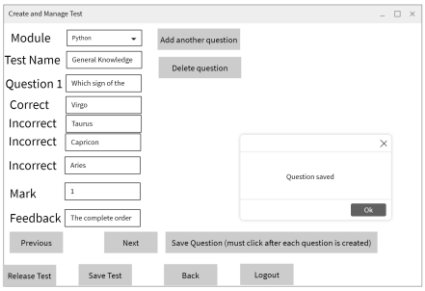
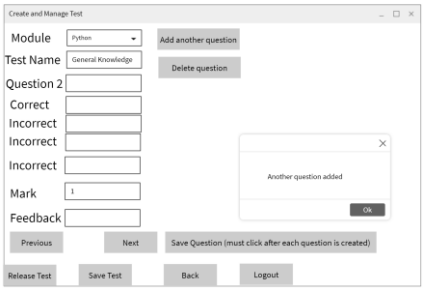
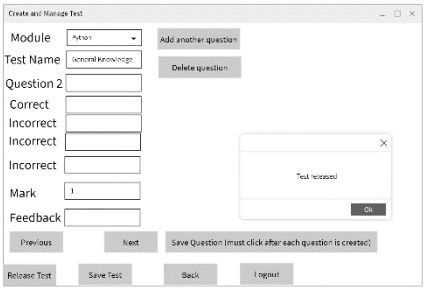
| | | | |
|---|---|---|---|
| 4 | Select ‘Java’ | Displays the tests taken in the selected subject, that are available to view the results for  | |
| 5 | Select ‘Test 1: Capital Cities’ | Displays the student’s results for that test. Displays which questions they got correct/incorrect and their total mark. There is a button that links to the Feedback.  | |
| 6 | Select “View Feedback” | Displays the test questions and general feedback for the reasoning behind each correct answer. You can then click ‘Home’ to return to the starting page.  | |

| | |
|---|---|
| **Comments:** | |
| **Related Tests:** | |
| **Author: Hannah Rosser** | **Checker:** |

## 4a – Create Formative Test

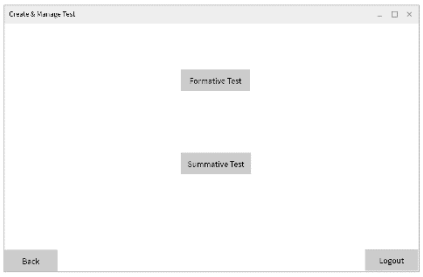
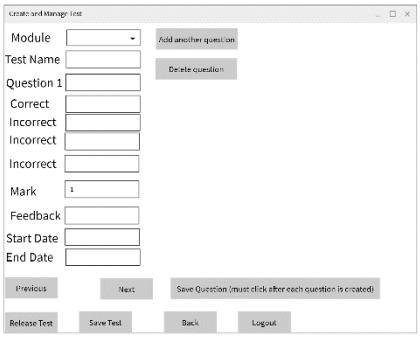
| | | | |
|---|---|---|---|
| **Test Case Id:** 4a | **Test Purpose:** To be able to create a formative test. | | |
| **Environment:** Dell XPS 15 running windows 10 written in Python 3.7 in Sublime text editor using Tkinter | | | |
| **Preconditions:** Lecture is logged into the system | | | |
| **Test Case Steps:** Basic flow | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click on create test. | Displays choice of either summative or formative test.   | |
| 2 | Click on formative test. | Displays a page with a form.   | |
| 3 | Choose Python in module combo box. | Python is selected.   | |
| 4 | Input “General Knowledge” into the entry box for test name. | Test name is created. | |

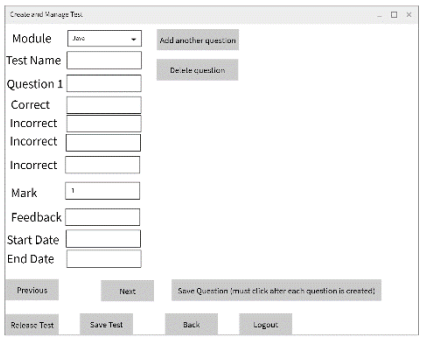
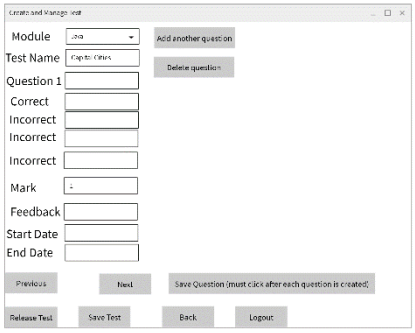
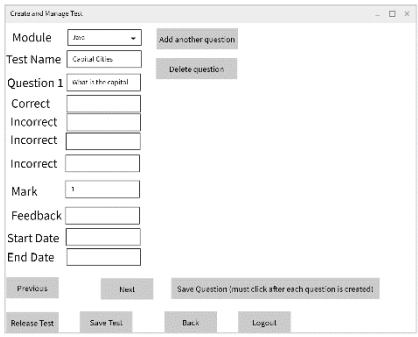
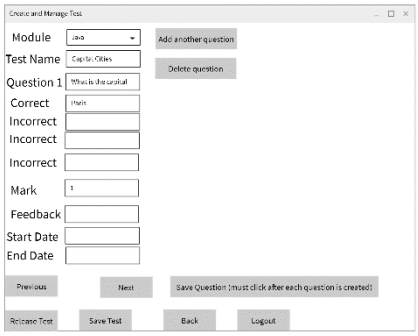
| | | | |
|---|---|---|---|
| | |  | |
| 5 | Add the question:” Which sign of the zodiac comes between Leo and Libra?” in the question 1 input box. | Question name is created.  | |
| 6 | Enter “Virgo” into the correct input box. | Correct answer inputted in the correct answer input box.  | |
| 7 | Enter “Taurus”, “Capricorn”, “Aries” for the three incorrect answers into the three incorrect input boxes. | Incorrect answers inputted into the boxes.  | |

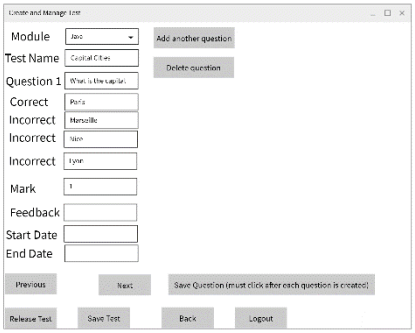
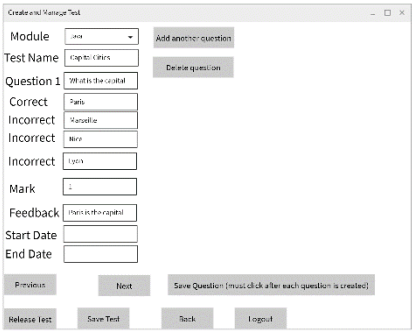
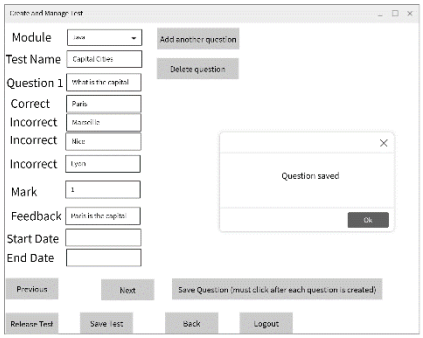
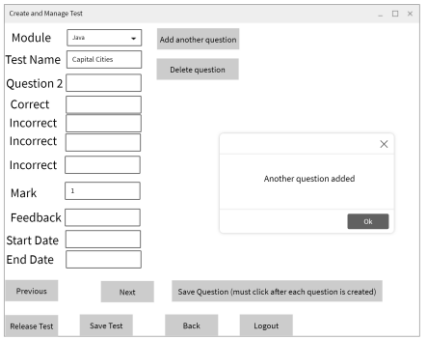
| | | | |
|---|---|---|---|
| 8 | Add “The complete order of the zodiac is: Aries, Taurus, Gemini, Cancer, Leo, Virgo, Libra, Scorpio, Sagittarius, Capricorn, Aquarius then finally Pisces.” in feedback entry box. | Feedback inputted in feedback input box.   | |
| `9 | Click “Save Question” button to save question. | Message box appears.   | |
| 10 | Continue creating test by clicking add another question. | Another question added.   | |
| 11 | Repeat steps 5 to 10 but for a new question until the desired amount of questions. | Output will be different for a new question. | |
| 12 | Click release test. | Test released message box will appear and text saved in a json file with module and test name as the file name in “formative/Released" directory.   | |

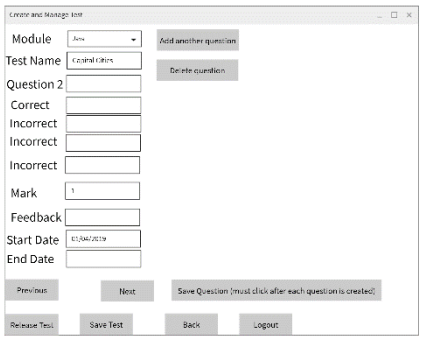
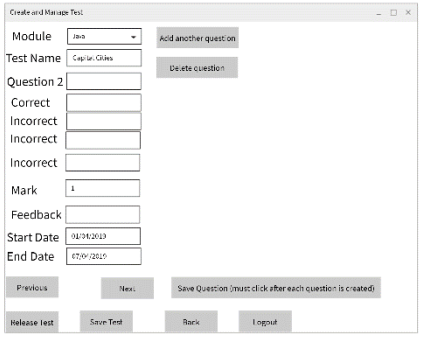
| | | | |
|---|---|---|---|
| 13 | Click logout. | Interface quits. | |
| **Comments:** | | | |
| **Related Tests:** Manage Formative test. | | | |
| **Author:** Spencer Du | | **Checker:** | |

## 4b – Create Summative Test

| | | | |
|---|---|---|---|
| **Test Case Id:** 4b | **Test Purpose:** To be able to create a summative test. | | |
| **Environment:** Dell XPS 15 running windows 10 written in Python 3.7 in Sublime text editor using Tkinter | | | |
| **Preconditions:** Lecturer is logged into the system | | | |
| **Test Case Steps:** Basic flow | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click on create test. | Displays choice of either summative or formative test.   | |
| 2 | Click on summative test. | Displays a page with a form.   | |

| | | | |
|---|---|---|---|
| 3 | Choose Java in module combo box. | Java is selected.<br>  <br> | |
| 4 | Input “Capital Cities” into the entry box for test name. | Test name is created.<br>  <br> | |
| 5 | Add the question:” What is the capital city of France?” in the question 1 input box. | Question name is created.<br>  <br> | |
| 6 | Enter “Paris” into the correct input box. | Correct answer inputted in the correct answer input box.<br>  <br> | |

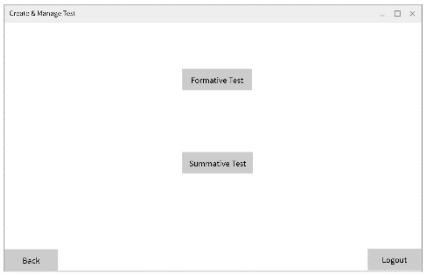
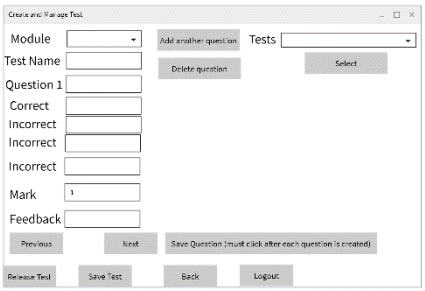
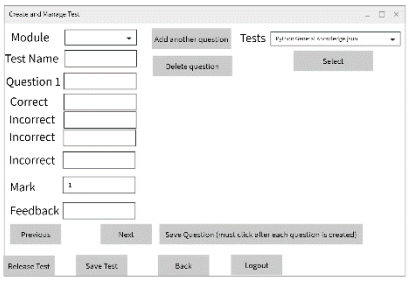
| | | | |
|---|---|---|---|
| 7 | Enter “Marseille”, “Nice”, “Lyon” for the three incorrect answers into the three incorrect input boxes. | Incorrect answers inputted into the boxes.   | |
| 8 | Add “Paris is the capital city of France.” in feedback entry box. | Feedback inputted in feedback input box.   | |
| 9 | Click “Save Question” button to save question. | Message box appears.   | |
| 10 | Continue creating test by clicking add another question. | Another question added.   | |

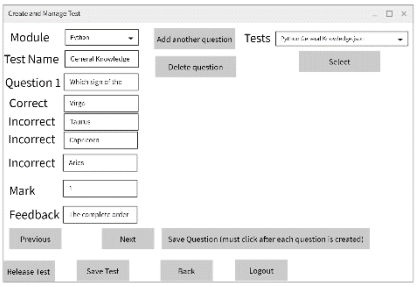
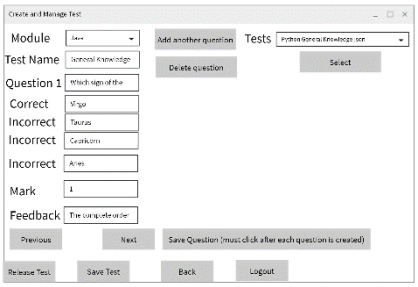
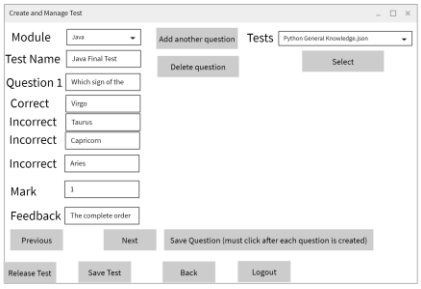
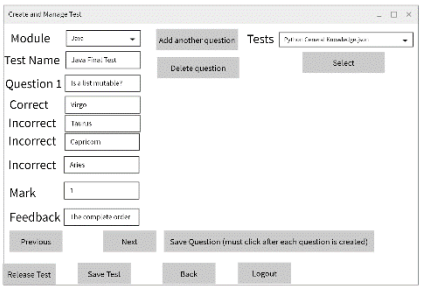
| | | | |
|---|---|---|---|
| 11 | Repeat steps 5 to 10 but for a new question until the desired amount of questions. | Output will be different for a new question. | |
| 12 | Add start date to start date input box. | Start date added in start date input box.<br>  <br> | |
| 13 | Add end date to end date input box. | End date added in end date input box.<br>  <br> | |
| 14 | Click release test. | Test released message box will appear and text saved in a json file with module and test name as the file name in “summative/Released" directory.<br>  <br> | |
| 15 | Click logout | Interface quits | |
| **Comments:** | | | |

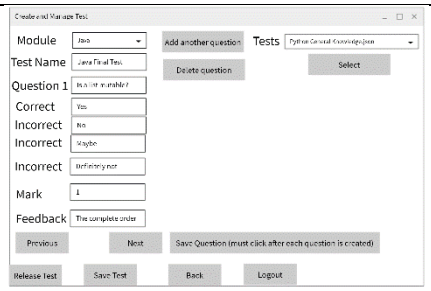
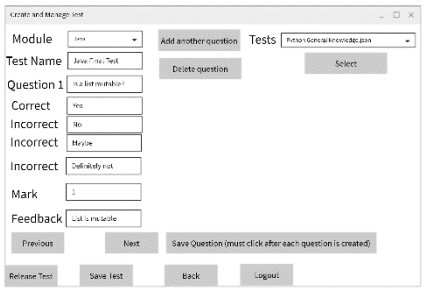
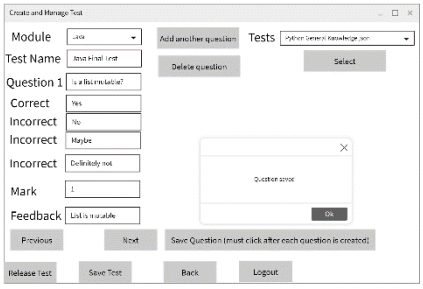
**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

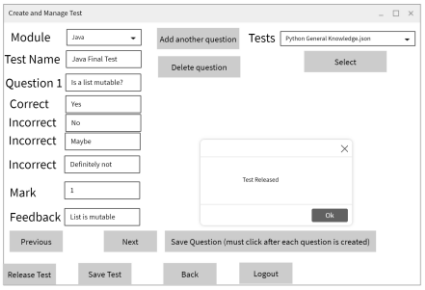
| | |
|---|---|
| **Related Tests:** Manage summative test | |
| **Author:** Spencer Du | **Checker:** |

## 5a – Manage Formative Test

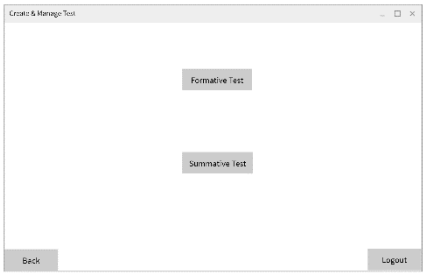
| | | | |
|---|---|---|---|
| **Test Case Id:** 5a | **Test Purpose:** To be able to manage formative tests | | |
| **Environment:** Microsoft Surface laptop running windows 10 written in Python 3.7 in Sublime text editor using Tkinter | | | |
| **Preconditions:** Lecturer log into the system | | | |
| **Test Case Steps:** Basic flow | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click on “Manage tests” button | Displays choice of either summative or formative test.   | |
| 2 | Click on “formative test” button. | Displays a page with a form.   | |
| 3 | Choose “Python General Knowledge.json” in tests combo box | “Python General Knowledge.json” is selected.   | |

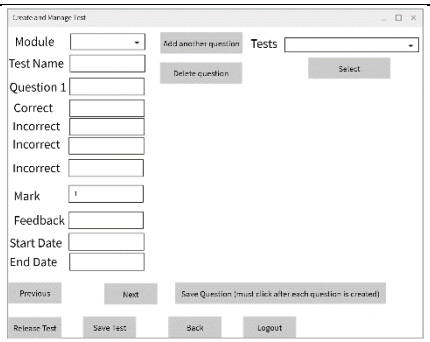
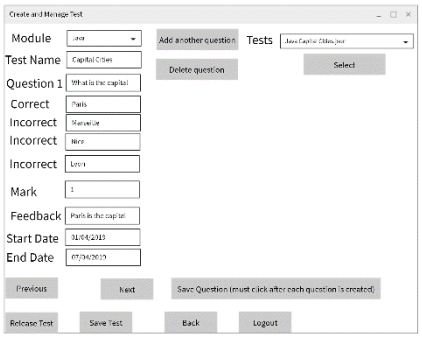
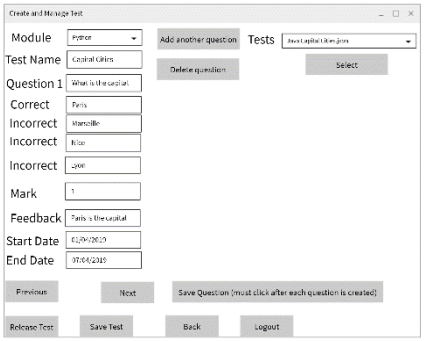
| | | | |
|---|---|---|---|
| 4. | Click on “select” button | The information of “Python General Knowledge” test is loaded.  | |
| 5 | Amend the module of the test to Java. | Displays the change of the module to Java  | |
| 6 | Amend the name of the test to “Java Final Test” | Displays the change of the name to “Java Final Test”.  | |
| 7 | Amend question of the test to “Is a list mutable?” | Displays the change of question to “Is a list mutable?”  | |
| 8 | Amend answers of the question to yes, no, maybe, definitely not | Displays the change of answers to yes, no, maybe, definitely not. | |

| | | | |
|---|---|---|---|
| | |  | |
| 9 | Amend the feedback of the formative test to “List is mutable because it can be changed”. | Displays the change of the feedback to “List is mutable because it can be changed”.  | |
| 10 | Click on “Save Question” button | Message box appears with “Question saved”.  | |
| 11 | Repeat steps 5 to 10 but for editing a new question until the desired amount of questions. | Output will be different for a new question. | |

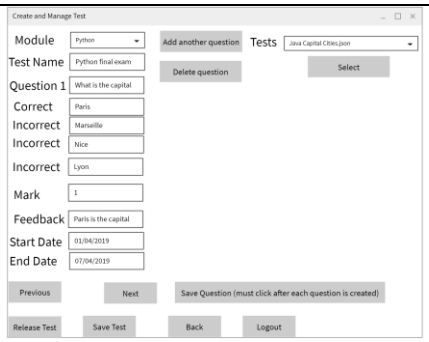
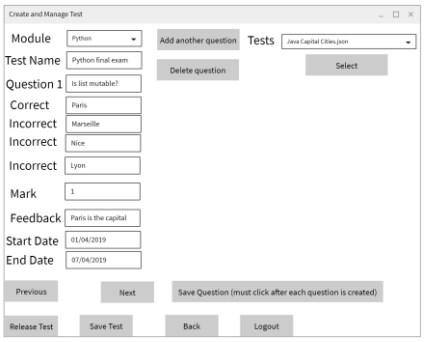
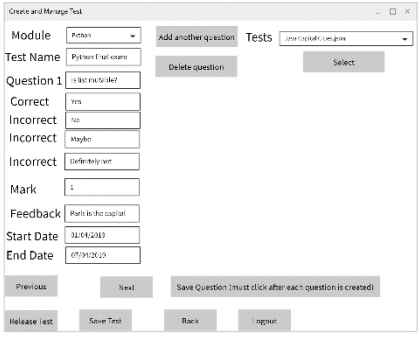
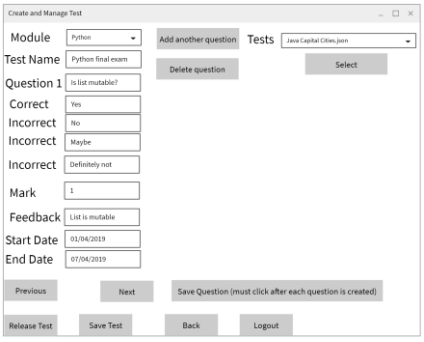
| | | | |
|---|---|---|---|
| 12 | Click on “Release test” button | Test released message box will appear and text saved in a json file with module and test name as the file name in “formative/Released" directory.   | |
| 13 | Click logout | Interface quits | |
| **Comments:** | | | |
| **Related Tests:** Create formative and summative tests | | | |
| **Author:** Qifa Cai | | **Checker:** | |

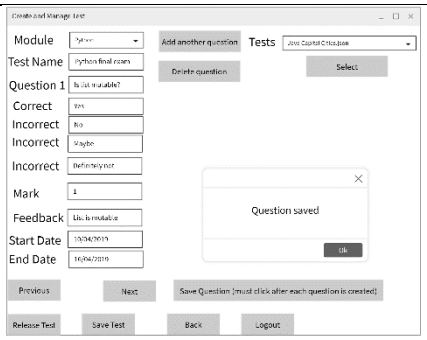
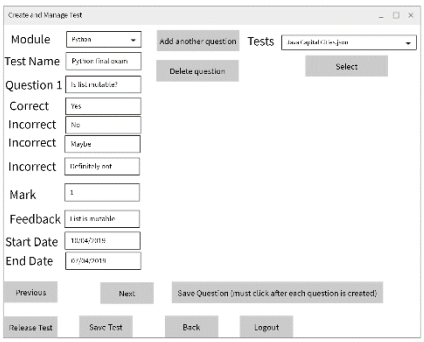
## 5b – Manage Summative Test

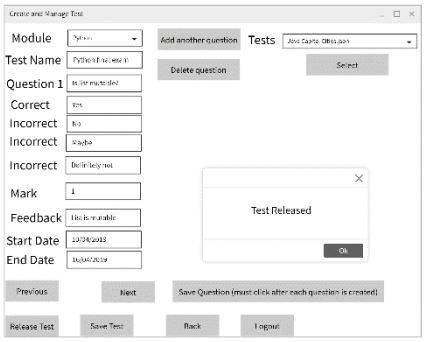
| | | | |
|---|---|---|---|
| **Test Case Id:** 5b | **Test Purpose:** To be able to manage summative tests | | |
| **Environment:** Microsoft Surface laptop running windows 10 written in Python 3.7 in Sublime text editor using Tkinter | | | |
| **Preconditions:** Lecturer log into the system | | | |
| **Test Case Steps:** Basic flow | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Click on “Manage tests” button | Displays choice of either summative or formative test.   | |
| 2 | Click on “summative test” button. | Displays a page with a form. | |

| | | | |
|---|---|---|---|
| | |  | |
| 3 | Choose "Java Capital Cities.json" in tests combo box | "Java Capital Cities.json" is selected<br>  | |
| 4 | Click on "select" button | The information of "Java Capital Cities.json" test is loaded<br>  | |
| 5 | Amend the module of the test to Python | Displays the change of the module to Python<br>  | |
| 6 | Amend the name of the test to "Python final exam" | Displays the change of the name to "Python final exam". | |

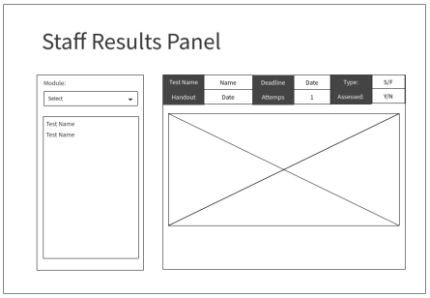
**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

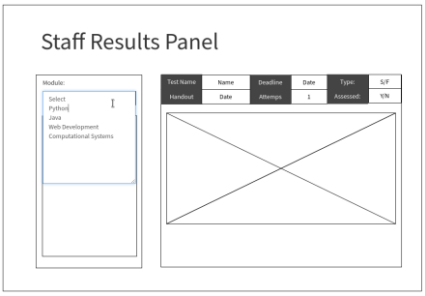
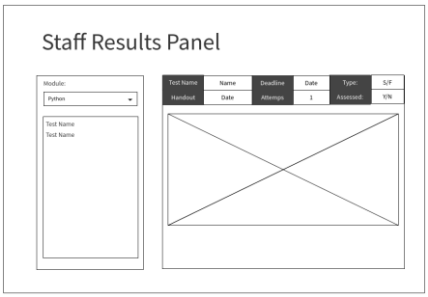
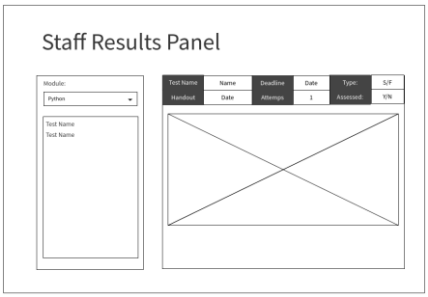
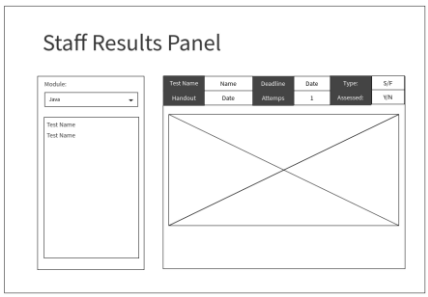
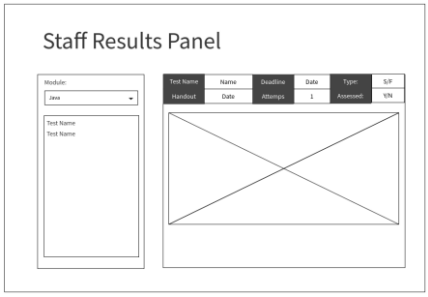
| | | | |
|---|---|---|---|
| | |  | |
| 7 | Amend question 1 of test to “Is list mutable?” | Question 1 changed to “Is list mutable”.<br>  | |
| 8 | Amend answers of question 1 to yes, no, maybe, definitely not | Displays the change of answers to yes, no, maybe, definitely not<br>  | |
| 9 | Amend the feedback of question 1 to “List is mutable because it can be changed”. | Displays the change of the feedback to “List is mutable because it can be changed”.<br>  | |
| 10 | Click on “Save Question” button | Message box appears. | |

| | | | |
|---|---|---|---|
| | |  | |
| 11 | Repeat steps 5 to 10 but for editing a new question until the desired amount of questions. | Output will be different for a new question. | |
| 12 | Amend the start date time of the summative test to “10/04/2019 | Displays the change of the start date time to “10/04/2019”  | |
| 13 | Amend the end date time of the summative test to “16/04/2019” | Displays the change of the end date time to “16/04/2019”.  | |

| | | | |
|---|---|---|---|
| 14 | Click on “Release test” button | Test released message box will appear and text saved in a json file with module and test name as the file name in “summative/Released" directory.   | |
| 15 | Click logout. | Interface quits | |
| **Comments:** | | | |
| **Related Tests:** Create summative test | | | |
| **Author:** Qifa Cai | | **Checker:** | |

## 6 – Staff Results Panel

| | | | |
|---|---|---|---|
| **Test Case Id:** See test results (staff) | | **Test Purpose:** See test results from the staff panel | |
| **Environment:** Fedora 29 and python 3.7 with tkinter | | | |
| **Preconditions:** Deadline for the test have passed. | | | |
| **Test Case Steps:** Basic flow | | | |
| Step No | Procedure | Response | Pass/Fail |
| 1 | Open the Results Panel |   | |

| | | | |
|---|---|---|---|
| 2 | Click on the module drop down menu |  | |
| 3 | Select a module, and the tests for that module will appear on the white box bellow. |  | |
| 4 | Select the test to see the results, results will appear on the Results Panel |  | |
| 5 | Select Another module, test results available will appear on the white box bellow the menu. |  | |
| 6 | Select the test to see it's results: |  | |

| | |
|---|---|
| **Comments:** | |
| **Related Tests:** | |
| **Author:** Armando Castany | **Checker:** |

**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

# Development Techniques and Methodologies

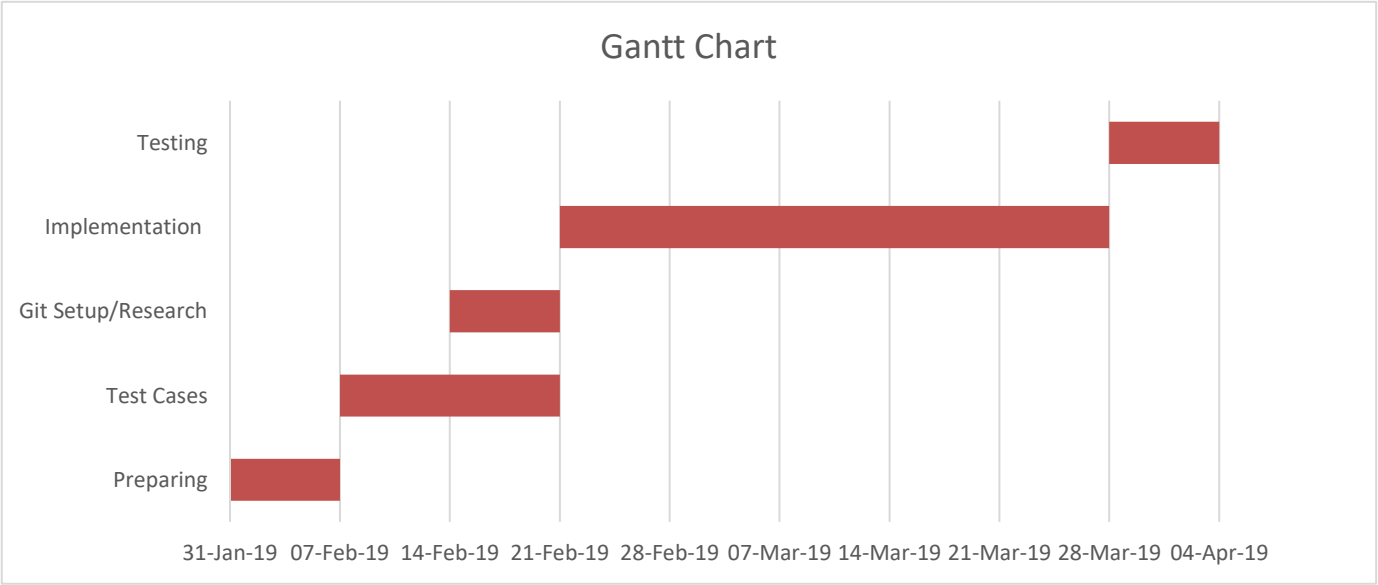
The Gantt chart below was used to illustrate our project schedule, along side using scrum methodologies to develop our project where we used fortnightly sprints, which will be explained below the Gantt chart.

Firstly, we allowed the first week (week starting Thursday 31st) of the semester to be a preparation week where all team members made sure we have the same version of Python and TKinter running on our systems (and allowed other members time to finish other assignment commitments).

Weeks 2 and 3 of the semester were aimed at completing draft versions of our test cases (with interface sketches) ready to be sent of for review. During the third week we also used the time to familiarise ourselves with using Git in a team environment as most members had little to no experience with Git (this was crucial as it allowed for version control of our software, which when designing a large project in a team is extremely important so everything can be kept track of).

Weeks 4 to 8 were left to develop the project, with every team member working on their individual modules (pairing up as some team members modules were similar – this is further explained later). We dedicated the most amount of time to this part of the project as making sure all the code works and runs together and smoothly is the most important part of the project.

The final week of the project was dedicated to testing the software, ensuring it was compatible on different devices and ready for the demonstration.




As a team we decided early on to use Scrum methodologies for agile development, implementing fortnightly sprints. The first sprint commenced on the 7th February with the final sprint starting 21st March, with the last scrum meeting being the week before the demonstration. The sprints reports are shown below, along with the weekly meeting notes and screenshots of the Trello board we used to keep track of tasks.

# Sprint One

## Sprint Planning (7th February)

**<u>Our sprint goal is to</u>;** develop the test cases for the main requirements of the system (including developing interface sketches); practice using Git and setting up of the repository and branches; practice using TKinter by following the workbook on Learning Central.

**<u>Our sprint backlog contains the following stories (titles only):</u>** None

## Weekly Scrum (7th Feb – Meeting 1)

**<u>Scrum / Trello Board</u>**

Trello Board Setup with tasks for each member to complete the test cases and interface sketches.

**<u>Narrative</u>**

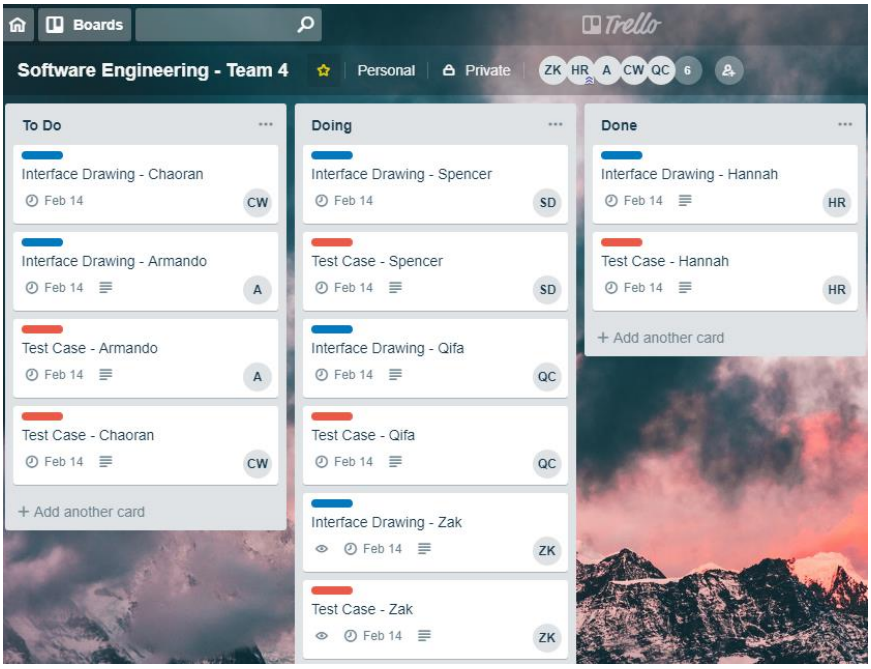
Sprint One goals set. Week 1 targets set with deadlines and Trello board setup.

There are no impediments

## Weekly Scrum (14th Feb – Meeting 2)

**<u>Scrum / Trello Board</u>**

Screenshot:




**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

**Narrative**

This week the team used the 3-hour meeting/work time to familiarize themselves with Git. Armando took the lead and explained to everyone how to manage git branches within a repository (with a helpful website visualizing the Git repository.

In the previous week the team had started work on their interface sketches and draft versions of the test cases.

This week the team will continue with their previous tasks, whilst also doing the TKinter workbook as advised by Zak, to familiarize the team with using TKinter within Python.

There are no impediments

**Sprint Review (21st February)**

Every team member had completed the draft versions of the Test cases and had emailed them off for review. The team is now comfortable with using Git and TKinter.

## Sprint Two

**Sprint Planning (21st February)**

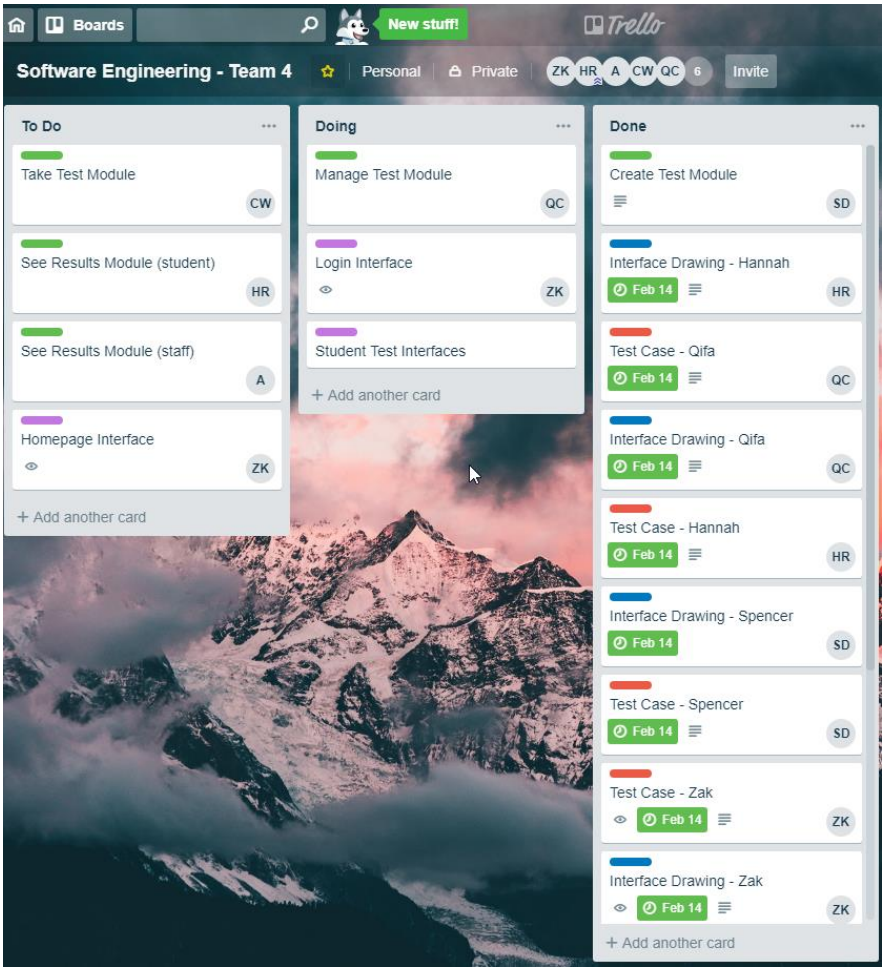
**Our sprint goal is to;** Start development of the classes from the class diagram (main class being the Test class); make some test dummy date and decide on the format for everyone to use within their modules (JSON, csv, etc.); develop some TKinter standards so that all the interfaces when developed are consistent with one another.

**Our sprint backlog contains the following stories (titles only):** Test Cases (complete), Interface Sketches (complete), Git Setup (complete), TKinter practice (complete).

## Weekly Scrum (21st Feb – Meeting 3)

**Scrum / Trello Board**

Screenshot:




**Narrative**

Sprint Two goals set. Week 1 targets complete. Trello board updated with new tasks.

This week the team made dummy tests and uploaded them and decided to use JSON format for the data structures. Also, a TKinter standard compliance was made (see 'Strategies and Methods to Ensure Good Quality Software for the standards).

The team will also continue working on developing their individual modules for the system.

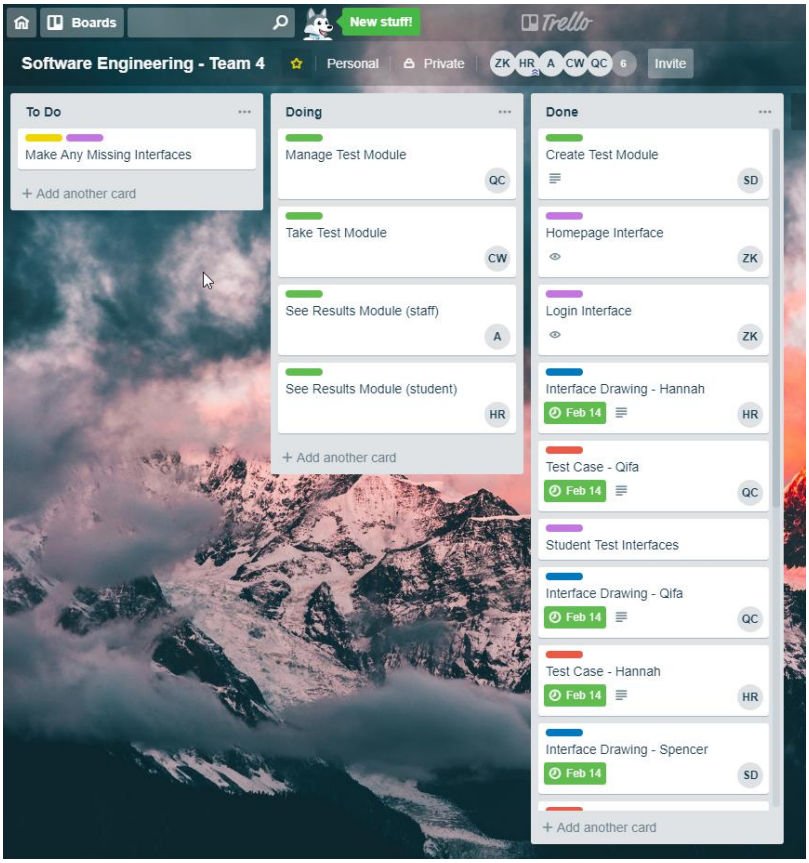
There are no impediments

**Team Members:** c1868921, c1772122, c1810275, c1840336, c1865452, c1531326.

## Weekly Scrum (28th Feb – Meeting 4)

**Scrum / Trello Board**

Screenshot:




**Narrative**

This week the team decided to continue working on their modules in pairs (as the work for both members in the pair was similar. With Qifa and Spencer working on the Create/Manage Tests, Hannah and Armando working on the student/lecture results page and Chaoran and Zak working on the Take Formative/Summative Tests.

Zak and Armando were absent in this meeting (due to extend deadlines on another assignment due to lack of lecture time in the previous semester).

**Sprint Review (7th March)**

Standards compliance and dummy data made. Good progress on the rest of the systems. On track to complete in time.

# Sprint Three

## Sprint Planning (7th March)

**Our sprint goal is to;** Continue developing the system, making all the necessary interfaces to connect the module together (i.e. home page screen, staff panel homepage). Bug fixing the modules which are almost finished.

**Our sprint backlog contains the following stories (titles only):** Test Cases (complete), Interface Sketches (complete), Git Setup (complete), TKinter practice (complete), Standard Compliance TKinter (complete), Dummy Data (complete), Modules Developed (incomplete).

## Weekly Scrum (7th March – Meeting 5)

**Scrum / Trello Board**

From this point on we have decided to stop using the Trello board as the main tasks are the same as before, to continue working on the modules together in pairs (adding all the necessary features).

**Narrative**

Sprint Three goals set. Week 2 targets complete (except for continuation of current tasks)

This week the team pairs continued working on their modules together, helping with bug fixes and layout issues discovered with the question's layout on the student interface.

The team will also continue working on developing their individual modules for the system.

There are no impediments

## Weekly Scrum (14th March – Meeting 6)

**Narrative**

Meeting cancelled due to illness and other commitments within the team. Progress still being made on the system between team pairs.

**Sprint Review (21st March)**

All the necessary interfaces are made for connecting all the modules. The modules are all very almost finished (just bug fixing and tidying up left).

# Sprint Four

## Sprint Planning (21st March)

**Our sprint goal is to;** Connect the modules together to make one system with smooth transition between all the modules. Bug fix and tidy up the 'final demo' version. Test the software and do a practice run of the demonstration.

**Our sprint backlog contains the following stories (titles only):** Test Cases (complete), Interface Sketches (complete), Git Setup (complete), TKinter practice (complete), Standard Compliance TKinter (complete), Dummy Data (complete), Modules Developed (Complete).

## Weekly Scrum (21st March – Meeting 7)

**Narrative**

Sprint Four goals set. Week 3 targets complete.

The week the team all meet and discussed the final steps to finish the system. Including making small adjustments such as button sizes, text within the buttons, etc...

## Weekly Scrum (28th March – Meeting 8)

**Narrative**

Final meeting before the demonstration – Done a test run of the system on the lab computers ensuring everyone knows what to say about all the features of the system and ensuring the system works in that environment.

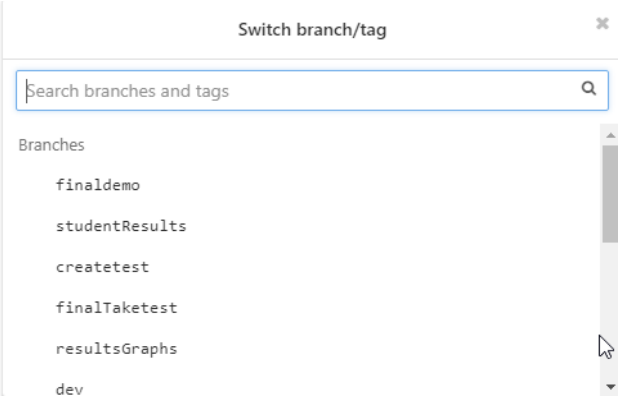
### Sprint Review (4th March)

Final review to be had during a meeting after the demonstration (post deadline of this report).

# Strategies and Methods to Ensure Good Quality Software

## Version Control

As a team we decided very early on to use Git to keep track of our software development and version control. Each team member had a branch they could create each module on, once a module had been complete it was merged into the main branch. Armando oversaw the Git repo as he was very experienced with it.




As shown in the screenshot on the right, each class had its own branch. Which when completed Armando merged to the 'dev' branch. Also, the final amendments were done on the 'finaldemo' branch, which was then merged to the dev branch as the final version of the software.

Git proved to be extremely useful, it helped keep track of versions when there were issues and it helped keep track of all the changes team members were making (see images below).


## McCall's Quality Model

Our team followed McCall's Quality model when developing the code for the system. Following the Quality Factors of Product **Operation**: Correctness; our code ensures to fulfil the full specification of the system. Reliability; our code will not fail, there will be no crashes in the final version and the code will be simple to understand. Efficiency; the code will be as efficient as possible to reduce the amount of computing needed to be done to perform a function. Usability; this is most import for our team, ensuring that users can operate our system without requiring a lot of effort to learn how it works.

We also made sure to follow the Quality Factors of Product **Revision** model including: Maintainability; the code will be split up into modules (classes) so that if a fault needs to be fixed, it can be easily located and changed. Testability; the separate modules can be tested individually to make sure that it is error free and meets the specification.

We also made sure to follow the Quality Factors of Product **Transition** model including: Portability, reusability and interoperability; the software is designed to work on all types of systems and environments without the need to change anything.

## Programming Style

The team has adhered to the Style Guide for Python Code (PEP8). The correct Naming Conventions have been followed, with classes being capitalised on each word and functions with underscored between each word. Imports are all at the top of the file. Frequent use of detailed comments to help the team members understand each other's code without the need to contact them. Also, the code has followed the PEP8 guidelines for whitespace and indentation (all of these can be seen in the screenshot below).

```
class TestCreaterApp(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.title_font = tkfont.Font(family='Helvetica', size=18, weight="bold", slant="italic"
        self.is_save = False

        # The container is where we'll stack a bunch of frames
        # on top of each other, then the one we want visible
        # will be raised above the others.
        container = tk.Frame(self)
        container.pack(side="left", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.quiz = Test(True, 'testID', 'module')

        self.frames = {}
        for F in (LoginPage, StudentLogin, StaffLogin, Register, StartPage, TestPage, ManageTest
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame

            # Put all of the pages in the same location;
            # the one on the top of the stacking order
            # will be the one that is visible.
            frame.grid(row=0, column=0, sticky="nsew")
```