
Project Validation

Team: L02_04

Alley Dismay, Ka-Kit Jason Cheung, Pak Sun (Harrison) Fok, Zenan (Will) Ji, Zongyang (Eddie) Li

November 19, 2017



Table of Contents

Product Backlog	3 - 9
Changes in the Product Backlog	10
Sprint 3 Backlog	
User Stories/Tasks/Story Points.....	11 - 16
Sprint Plan	17 - 18
Sprint Report	19 - 20
Burndown Chart	21
Snapshots of Taskboard	22 - 25
Sprint 4 Backlog	
User Stories/Tasks/Story Points.....	26 - 27
Sprint Plan	28 - 30
Sprint Report	31 - 32
Burndown Chart	33
Snapshots of Taskboard	34 - 43
Verification and Validation	
Verification, Validation, Code Review Strategy	44
Code Review Summary	45 - 46
Code Review Debriefing	46

Product Backlog (Version 2)

Personas

Student Personas:

Sara Bai

- 20 year-old student, female, Canadian-born-Chinese, in 2nd year at UTSC.
- Speaks English fluently.
- Specialist in Sociology with a 2.5 CGPA.
- Loves to read long papers and write essays, but hates math.
- Low technological competence, only uses her computer to watch videos on YouTube, basic usage of MS Word to write essays and uses default web browser on computer to check UTSC Portal.
- Uses a very basic Nokia phone to make phone calls and text.
- She always prints her homework off the computer because she hates reading off a computer screen.
- Takes STAB22 as an elective, since she was told it was an easy A.
- Doesn't want to spend too much time learning software for an elective course, just to complete assignments.
- Wants to just use her uToronto credentials as opposed to making a new account as it would be too much work.
- Likes to divide her assignments and homework into several parts to complete them in smaller chunks, over time.
- Would rather do hardcopy assignments because she believes using software will complicate things.
- Likes to be prepared and caught up for every lecture and before doing every assignment.

Lucas Liu

- 20 year-old student, male, from China, in 3rd year at UTSC.
- Speaks English and Chinese fluently.
- Double major in Statistics and Physical Sciences
- Introvert and autistic, never wants to communicate with people and enjoys staying alone in a room.
- Good computer skills: uses applications like the Microsoft Suite, Dropbox, Skype, computer games
- Typically too lazy to do homework because it's too boring and not engaging enough, would rather play video games like Dota 2
- Very interested in exploring new software and reading about new technologies.
- Works 25 hours a week at a retail store, trying hard to balance work and school.
- Likes to do a lot of practice and do as many problem sets as he can to prepare for tests and understand material
- Hates when professor give limited examples to work on
- Wants to be able to redo assignments to get the best possible mark.
- Appreciates how software looks and loves to use aesthetically pleasing software.
- Doesn't always have internet connection though, so prefers to have a desktop app rather than web app

Alice Smith

- 22 year-old student, female, single Irish-Canadian.
- Lives on residence, with constant access to the school's wifi.
- Speaks English and Irish Gaelic.
- Specialist in Statistics, learns new things quickly and has strong multitasking abilities.
- Passionate about academics, outgoing and spends her free time getting her work done as soon as possible, tries to learn the fastest way of doing everything.
- Loves technology and has access to the cutting-edge of technology.
- Proficient in MS Suite, Photoshop, Final Cut Pro, and numerous other softwares.
- Only likes to do assignments once and disregards them afterwards, not looking at them again.
- Gets frustrated easily when it takes multiple steps to do something simple.
- Is a visual learner, so likes to look at graphs and charts which will help her learn the material better.
- Likes to work on mobile devices like her phone or tablet because its more convenient for her to carry around, rather than a desktop or laptop.
- Likes to keep track of her performance in assignments and other course work she has done.

Professor Personas:

Professor Obi Wan-Too-Thrie

- 65 year-old professor, female
- Speaks English and Hindi
- Works as a professor in the Statistics Department where she mainly teaches introductory statistics courses.
- Prefers to use simple software that is straight forward and uncomplicated.
- She can only perform tasks when someone teaches her and follows the exact same steps
- Little exposure to technology, no cell phone and uses very basic functions in MS Word and Excel to enter grades of her students.
- Teaching style: lets students practice on their own to figure out their own method of approaching problems.
- Doesn't use software that is not engaging or interesting aesthetically
- Currently has students print out exercises to do and hand in at tutorial, but those take really long to be marked (2-3 weeks)
- Teaches classes as large as 1000 students and is usually unable to go over exercises in lecture
- Wants to be able to generate many different questions and types for students through the software, so that students can work them out at home.
- Wants software to be intuitive and familiar to the current software she uses, like MS Word and Excel.
- Believes that gamification(virtual rewards) of exercises/ assignments will increase student's interest in learning and course content.

Professor Bob

- 35 year old stats professor, male with no wife or kids
- works at a the university of Toronto with a degree in both Computer Science and Statistics as a Professor and researcher
- Prof Bob is a full time professor that teaches multiple first and second year courses on Statistics
- Likes to be very organized, likes predictability and things to be done quickly and efficiently
- Speaks English only
- Used to use software for assigning and having students answer questions, but it has become outdated, confusing to use and didn't have sufficient functionality (adding graphs, attachments to questions), which made him frustrated and stopped using it
- good computer skills in the Microsoft Suite, multiple IDEs and terminal
- Has at most 100 students in each class and only a couple of TAs to mark course work since Prof Bob is constantly busy doing research and writing papers
- wants to be able to quickly and easily create and upload questions and distribute them amongst all students
- wants students to be able to provide immediate feedback once students finish an assignment
- Likes to see how the class is doing and what material they are struggling with
- his teaching style is that he prefers students learn by example, rather than memorizing theorems, hoping using the system will allow for abundant question generation to provide many examples to learn and prepare with for exams
- want to give students as much practice / retries on different assignment questions until the deadline of the assignment

User Stories

- As a Professor (Obi-Wan-Too-Thrie), I want to be able to create different types of questions like multiple choice, fill-in-the-blanks, matching, etc.
- As a Professor (Obi-Wan-Too-Thrie), I want each student's version of each question to be different, by having randomized numerical values.
- As a Professor (Bob), I want to assign graphics to questions.
- As a Professor (Bob), I want to be able to add/edit/delete questions from my csv.
- As a Professor (Bob), I want to be able to select questions from the csv and create an assignment (a collection of questions) for the students to answer.
- As a Professor (Bob), I want to view/edit/remove questions from my csv after I've created them.
- As a Professor (Obi-Wan-Too-Thrie), I want to register a professor profile.
- As a student (Sara), I want to be able to make a student profile.
- As a Professor (Bob), I want to toggle the visibility of assignments for students, so that students won't see drafts of the assignments.
- As a Professor (Bob), I want to set a timeframe for when assignments can be completed. (i.e. If it's 1 week of visibility. At the end of 1 week, the assignment's visibility is automatically turned off).
- As a student (Alice), I want to be able to see a list of assignments that I have completed and ones that I have not yet completed.
- As a student (Sara), I want to know the deadline of each posted homework.
- As a student (Sara), I want to view which homework is graded.
- As a student (Lucas), I want to be able to answer the questions using this system and submit the assignment.
- As a student (Sara), I want the online homework system to allow me to save my current answer and quit an unfinished homework.
- As a Professor (Obi-Wan-Too-Thrie), I want a set of questions belonging to an assignment to be automatically marked as soon as a student completes it, so that they get immediate feedback.
- As a professor (Bob), I want to be able to store the student's highest, most recent grade on an assignment in the csv.
- As a student (Sara), I want to know the mark of all the tests and the solution of the last test done.
- As a Professor (Bob), I want to see how many students get each question right or wrong, so that I can gauge the class performance.

-
- As a Professor (Bob), I want assignment feedback to consist a list of correctly and incorrectly answered questions, the number of questions answered correctly in percentage and also the correct answers for questions that were answered incorrectly.
 - As a student (Lucas), I want to be able to redo assignments with new questions until the deadline so that I can get the most practice on a topic.
 - As a Professor (Obi-Wan-Too-Thrie), I want one occurrence of each selected question in a single assignment.
 - As a Professor (Obi-Wan-Too-Thrie), I want an assignment to re-generate random values each time a student attempts an assignment.
 - As a Professor (Bob), I want to be able to create multiple lectures for each class I teach, if I need to.
 - As a Professor (Obi-Wan-Too-Thrie), I want to be able to add students to a class, so that they can view and complete any assignment questions in the course.
 - As a student (Sara), I want to login to the online homework system using my UTORid and password.
 - As a Professor (Obi-Wan-Too-Thrie), I want to create badges and rewards for students depending on their performance on assignments, so that the gamification aspect increases interest.
 - As a student (Alice), I want to receive a notification/email indicating when a new assignment has been uploaded, so that I can finish it as soon as possible.
 - As a student (Sara), I want to know the related chapter of each posted homework.
 - As a student (Sara), I want to be able to save the assignment as a PDF, and other formats and print it so that I can work outside the app.
 - As a student (Lucas), I would like to return to where i stopped when reopen the app.
 - As a student (Alice), I want to track and see my progress on an assignment or set of questions with a progress bar.
 - As a student (Sara), I want the online homework system to have a tutorial to teach me how to use this system when I first use it.

The change in the Product Backlog

First Change

At the beginning of sprint 4, we were picking up user stories and decomposing them into tasks to do for the sprint. When we saw the user stories

As a student (Alice), I want to be able to see a list of assignments that I have completed and ones that I have not yet completed.

And

As a student (Sara), I want to know the deadline of each posted homework.

These 2 user stories in the product backlog are highlighted because there were changes for both for them. We realized that the deadline of the assignment could be displayed simultaneously along with each assignment. As a result, we decided to merge these 2 user stories into 1 to form the new user story

As a student (Sara), I want to be able to see a list of assignments that are both completed and incomplete along with relevant information about them such as deadline and assignment name,

Second Change

At the beginning of sprint 3, although we finished implementing many features from sprint 1 and 2 the features were not connected together which didn't allow users to access all these features easily. As a result, we added the following user story

As a professor (Bob) or student (Alice), I want to be able to log in and have a menu to display possible features I can do based on my user type. (ie. Students can view and complete assignments while professors create new assignments and view the ones they've made.)

These 2 new user stories are labelled 13 and 14 respectively and are implemented in sprint 4.

Sprint 3

Decomposed user stories for Sprint 3

User Story 6: As a Professor (Bob), I want to add/view/edit/remove questions from my text file after I've created them. (6 story points)

Tasks:

- a) Repurpose the Question Creation UI to allow the professor to view, edit and remove questions. (2 story points)
 - i. Likely going to be selecting questions from a file that will give a list of questions, so make a window with the questions stacked vertically on top of each other.
 - ii. Then use built-in list methods to extract each question in the list and output it in a formatted way.
- b) Implement methods to add, edit and remove questions (2 story points)
- c) Connect the repurposed front-end UI to the newly implemented back-end method. (2 story points)

Depends on a, b

User Story 7: As a Professor (Bob), I want to be the only one able to add/edit/delete questions from a csv file. (2 story points)

Tasks:

- a) Implement a new class OptionsMenu.py that will check whether a user is a professor or student. (2 story points)
 - i) this class will display different buttons according to what type of user this is. (either student or professor)
 - ii) if the user is a professor, then provide a button/option to let the professor add/edit/delete questions from the csv.

User Story 8: As a Student (Lucas), I want to be able to answer questions for an assignment using the system and submit my assignment. (6 story points)

Tasks:

- a) Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer. (2 story points)
 - i) also have a submit button that saves all the answers for a student in a .csv file
 - ii) after submitting, show the correct answer beside the question if the student got it wrong.
- b) implement the back end methods to pull data from the csv file that stores the questions of an assignment and compares the student's answer with the one in the csv when they hit the submit button. (2 story points)
 - Dependent on a**
 - i) store the result of the student's answers from this assignment in a new .csv file called AssignmentAttempts.csv
 - ii) connect the back end with the front end.
- c) AssignmentAttempts.csv will hold all the results of an assignment, overwriting the respective row with the most recent attempt. (2 story points)
 - ie) the .csv file will have the 1st row correspond to the 1st assignment, 2nd row for 2nd assignment and so on...
 - Dependent on a, b**
 - i) if the student retries the 2nd assignment, the 2nd row will be overwritten.

User Story 9: As a Professor (Bob), I want to toggle the visibility of assignments for students, so that students won't see drafts of the assignments. (7 story points)

Tasks:

- a) create a button in the AssignmentDashboard GUI to be displayed beside an assignment name that will take you to the 'visibility options menu', with radio buttons for visibility on and visibility off. (2 story points)
 - i) create a class for an Assignment that has a boolean attribute for visibility, along with other necessary attributes
 - ii) Create a GUI for the student that will display all assignments that have its visibility attribute set to True.
 - iii) using tkinter to implement the UI
- b) implement the back end methods to pull an assignment from the .csv that stores all the assignments and change it's attribute for visibility depending on what the professor chose. (2 story points)
 - i) want to pull the assignment out of the .csv, update it's value for visibility and then insert it back into the .csv that stores all the assignments
 - ii) based on this value, the assignments list UI should display the appropriate assignments
- c) connect the front and back end together (3 story points)

User Story 10: As a Professor (Bob), I want to set a timeframe for when assignments can be completed. (ie. if the timer is 1 week long, at the end of the week the assignment's visibility will automatically be turned off.) (7 story points)

a) Add a button that takes the AssignmentDashboard GUI menu for the professor beside each assignment. (2 story points)

i) this button will take them to a new SetTimer GUI where they can set the visibility of that assignment for x number of hours/up to a certain date

b) Implement back-end methods so that when the time length is set in the SetTimer GUI, and the timer is up, the visibility will be automatically turned off for an assignment. (3 story points)

i) the next time the program is run and the student opens their dashboard to see the list of assignments, if the time is up then a method will pull the assignment from Assignments.csv, change the attribute for visibility to be false, and the student dashboard will reflect this.

c) Connect the front end to the back end (2 story points).

depends on a,b

User Story 11: As a Professor (Bob), I want to check if the student answers are correct for the assignment once they submit it. (7 story points)

- a) Implement methods to iterate through the .csv file to find the particular assignment the student is working on. (2 story points)
 - i) iterate through each question from the assignment and using the provided formula for the answer, compute the answer.
 - ii) form a dictionary with the key being the question id and the value being the calculate answer based on the formula.
- b) Implement methods to extract the student's answers from AssignmentAttempts.csv, and compare these answers with the calculated answer in part a) (3 story points)
Dependent on a
 - i) form a dictionary with the key being the question id and the value being the students answer
 - ii) iterate through both dictionaries from part a and b and compare the values given the same key.
- c) Design and implement a UI using Tkinter that will display the results of a students answers when compared to the actual answer. (1 story point)
- d) Connect the front-end and the back-end (1 story point)
Dependent on a, b, c

Sprint Plan

Team of 5 developers: Andy, Eddie, Harrison, Jason, Will:

- Andy is able to complete 10 story points a week
- Eddie - 6 story points a week
- Harrison - 8 story points a week
- Jason - 6 story points a week
- Will - 6 story points a week

Developers will work for 7 days a week and the length of the sprint is 1 week

Team Velocity

The team velocity will be 36 story points a week.

Day 1: November 6th, 2017

Day 7: November 13th, 2017

Tas ks	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
6a	Eddie: 2						
6b		Eddie:2					
6c				Eddie: 2			
7a					Andy: 2		
8a		Jason:2					
8b			Jason:2				
8c				Jason: 2			
9a					Andy: 2		

9b					Andy: 1	Andy: 1	
9c						Andy: 1	
9d						Andy: 2	
10a		will:2					
10b			will:2				
10c					will:2		
11a	H: 2	H: 2					
11b			H: 1				
11c			H: 1				
11d				H: 1			

Based on this provisional schedule, 6 user stories will be completed by the end of the sprint, with 35 user stories burned.

Sprint Report

This sprint report shows the actual hours spent during the sprint.

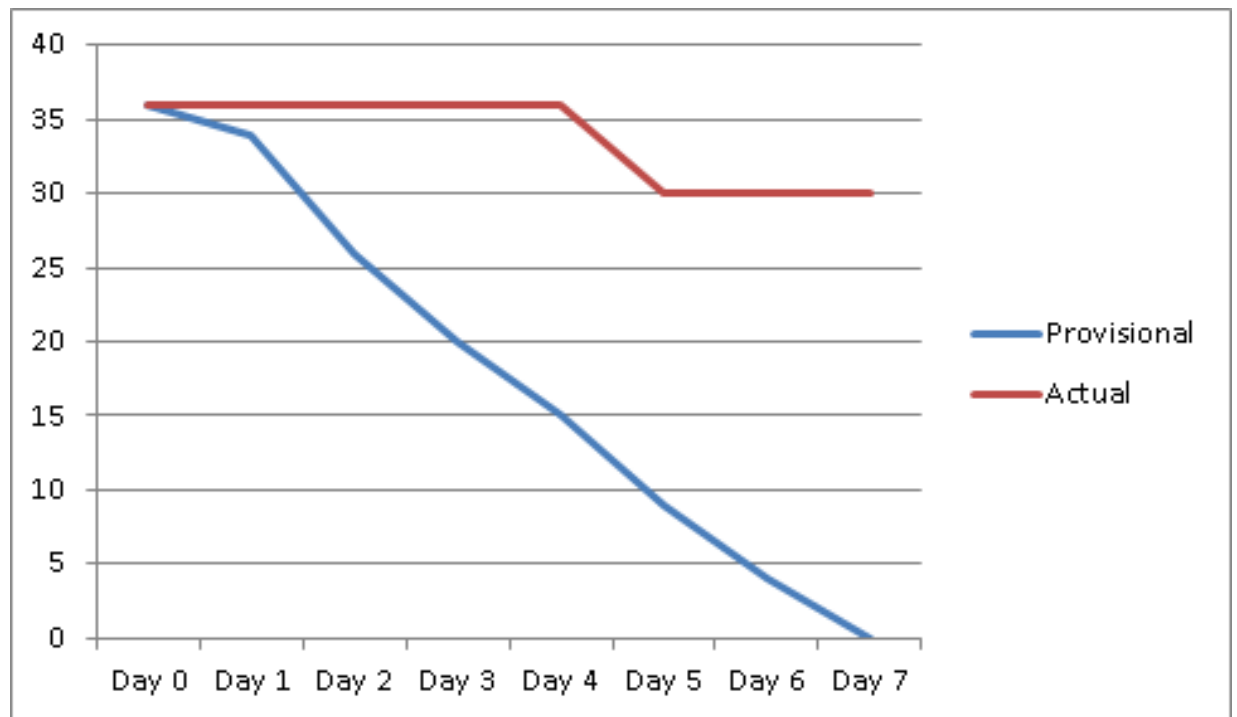
Tas ks	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
6a							
6b							
6c							
7a	Andy: 1						
8a							
8b							
8c							
9a							
9b							
9c							
9d							
10a			Will:2				
10b					Will:2		
10c					Will:2		
11a	Harris on: 3	Harrison :2	Harrison:2				
11b				Harriso n:1			

11c				Harrison:1			
11d				Harrison:2	Harrison:2		

At the end of this sprint only user story 10 was completed. This is due to the fact that many of us were occupied with midterms and other tasks at hand, however most of the members still devoted whatever possible time to implementing their features. From this, the unfinished tasks will be completed in sprint 4.

Burndown Chart

Story Points



Days

Snapshots of Taskboard

Since the Trello board was filled with too many tasks, 2 screenshots were not sufficient to capture the entire taskboard.

Start of Sprint

Each list has the task number, number of story points and the person responsible for the task. For example, the first one is task 6b, which requires 2 story point and is allocated to Eddie. Y = Andy, J = Jason, ZL = Eddie, W = William, HF = Harrison

CS001 CS001 Team ☆ Team Visible

Sprint Backlog	Current Work	Completed Work
6b) [2 story points] Implement methods to add, edit and remove questions ZL	6a) [2 story points] Repurpose the Question Creation UI to allow the professor to view, edit and remove questions. ZL	1a) [1 story point] Design a form that has an email, password field and a login button to allow the professor to log into the system. Y
6c) [2 story points] Connect the repurposed front-end UI to the newly implemented back-end method. Depends on a, b ZL	8a) [2 story points] Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer. J	1b) [2 Story points] Implement Professor class to generate professor objects that store the information of a single professor, along with its associated methods. (ie. getters and setters for the Professor). Y
8b) [2 story points] Implement the back end methods to pull data from the csv file that stores the questions of an assignment and compares the student's answer with the one in the csv when they hit the submit button. J	9a) [2 story points] create a button in the AssignmentDashboard GUI to be displayed beside an assignment name that will take you to the 'visibility options menu', with radio buttons for visibility on and visibility off. Y	1c) [1 story point] Create an event handler and methods so that when the submit button is clicked, information is taken from the email, password fields and a new professor is written to the appropriate txt file for storage. Depends on a and b Y
8c) [2 story points] AssignmentAttempts.csv will hold all the results of an assignment, overwriting the respective row with the most recent attempt. J	10a) [2 story points] Add a button that takes the AssignmentDashboard GUI menu for the professor beside each assignment. W	2b) [1 story point] Create a UI using Tkinter to allow a Student to register their account using their name, email, student number and password. J
9b) [2 story points] implement the back end methods to pull an assignment from the .csv that stores all the assignments and change it's attribute for visibility depending on what the professor chose Add a card...	11a) [2 story points] Implement methods to iterate through the .csv Add a card...	3a) [1 story point] Design and Add a card...

Sprint Backlog

9b) [2 story points] implement the back end methods to pull an assignment from the .csv that stores all the assignments and change it's attribute for visibility depending on what the professor chose.

Y

9c) [3 story points] connect the front and back end together. Dependent on a, b

Y

10b) [3 story points] Implement back-end methods so that when the time length is set in the SetTimer GUI, and the timer is up, the visibility will be automatically turned off for an assignment.

W

10c) [2 story points] Connect the front end to the back end. Dependent on a, b

W

11b) [3 story points] Implement methods to extract the student's answers from AssignmentAttempts.csv, and compare these answers with the calculated answer in part a)

Add a card...

Current Work

ZL

8a) [2 story points] Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer.

👁

J

9a) [2 story points] create a button in the AssignmentDashboard GUI to be displayed beside an assignment name that will take you to the 'visibility options menu', with radio buttons for visibility on and visibility off.

Y

10a) [2 story points] Add a button that takes the AssignmentDashboard GUI menu for the professor beside each assignment.

W

11a) [2 story points] Implement methods to iterate through the .csv file to find the particular assignment the student is working on.

HF

Add a card...

Completed Work

3a) [1 story point] Design and implement the UI for allowing a professor to choose a type of question to create.

ZL

3b) [1 story point] Create a general Question class that includes attributes about a question such as the question_id, type of question, question body, formula and its associated methods to add a question to a file for storage.

ZL

3c) [1 story point] Add subclasses for the Question class to encompass all the types of questions that can be made. Depends on a

ZL

3d) [3 story points] Design and implement forms / windows for the different types of questions (Multiple choice, fill in the blanks)

ZL

3e) [2 story points] Connect the front end UI elements to the back end methods. Depends on a, b, c

ZL

Add a card...

Completed Work

4c) [1 story point] Implement method to randomize the given variables of a question each time it's retrieved from the .txt file and inserted into another. Depends on a, b

W

5a) [3 story points] Implement methods to retrieve questions from a .txt file and insert them into another .txt file that will contain the questions for a particular assignment.

HF

5c) [1 story point] Keep track of how many questions are chosen from the .txt file.

HF

5d) [1 story point] Design and implement a UI that shows all the questions that have been chosen for an assignment.

HF

5e) [1 story point] Connect the front end and back end together.
Dependency: a, b, c, d

HF

Add a card...

Completed Work

5e) [1 story point] Connect the front end and back end together.
Dependency: a, b, c, d

HF

1d) [1 story point] Create a form that will display all the information about that professor to the screen.

Y

2a) [1 story point] Implement Student class to generate student objects that store the information of a single student, along with its associated methods. (ie. Getters/setters, txt helper methods)



J

4a) [3 story points] Implement methods to parse a formula provided by a professor that is associated with a particular question.

W

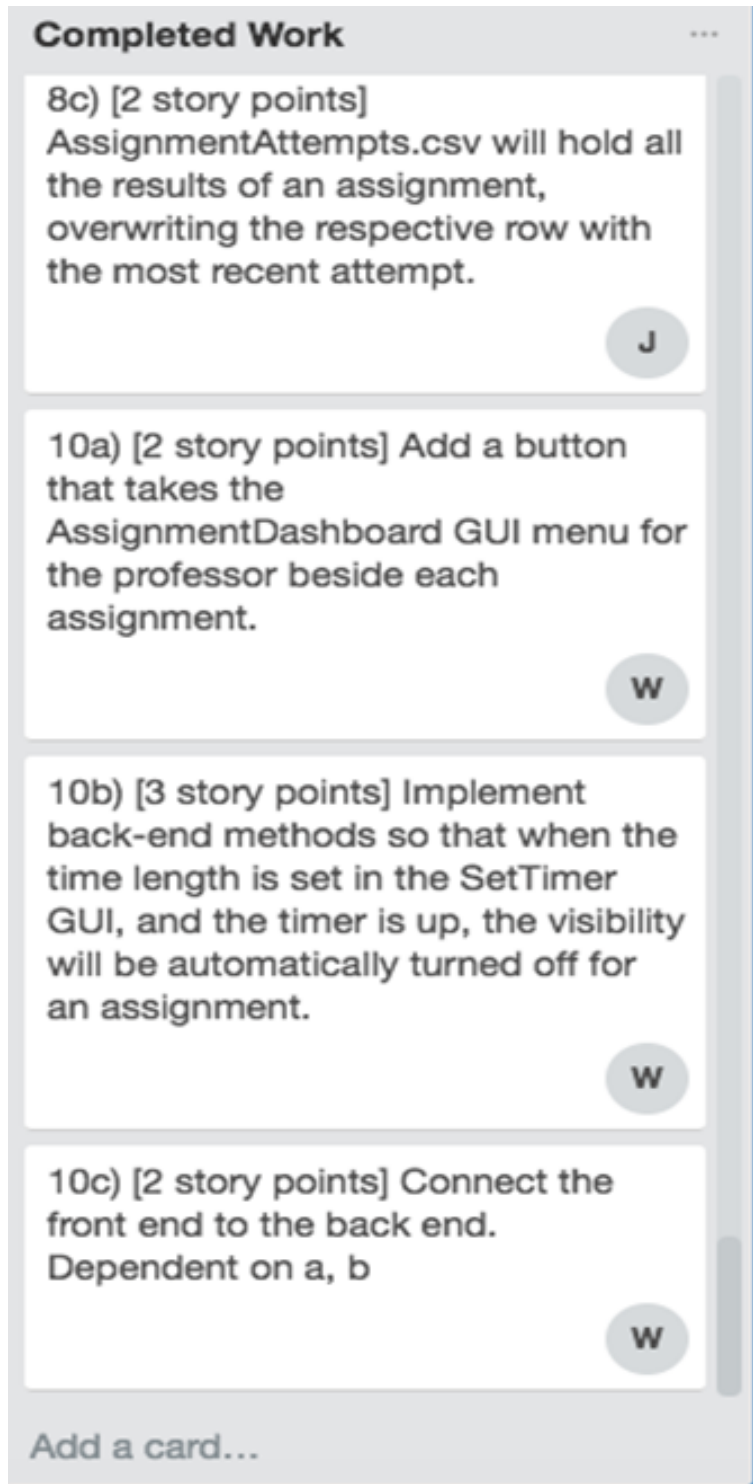
7a) [2 story points] Implement a new class OptionsMenu.py that will check whether a user is a professor or student.

Y

Add a card...

End of Sprint

Only user story 10 was finished. This snapshot is in addition to the previously finished tasks on the completed work list.



Sprint 4

Decomposed user stories for sprint 4

In addition to the user stories from sprint 3 (with the exception of user story 10 because it was finished), these are the user stories that were decomposed to be implemented in sprint 4.

User Story 12: As a professor (Bob) or student (Alice), I want to be able to log in and have a menu to display possible features I can do based on my user type. (ie. Students can view and complete assignments while professors create new assignments and view the ones they've made.) (3 story points)

- a) Draw a “flowchart” of what is supposed to happen in the GUI depending on what button is pressed **(1 story point)**
- b) Implement the flowchart **(2 story points)**
 - i) for each step in the flowchart, if a button is pressed we want the corresponding method to be called in another python file to render the appropriate methods/windows**Depends on a**

User Story 13: As a student (Sara), I want to be able to see a list of assignments that are both completed and incomplete along with relevant information about them such as deadline and assignment name. (6 story points)

- a) Design and implement a list layout using Tkinter to display information about the assignments that is relevant to students. **(2 story points)**
 - i) For each assignment, a new row on the window will display the following information about that assignment: the due date, assignment name.
- b) Implement back end method to format and display this data to the implemented UI. (dependent on a) **(2 story points)**
 - i) Check if the directory has a 'Assignment_studentNum.csv' file that exists for that student.
 - ii) If it exists, parse it to find out how many different assignments there are based on the assignmentID

Depends on a
- c) connect the front end UI and back end methods together **(2 story points)**

Depends on a, b

Sprint Plan

Team of 5 developers: Andy, Eddie, Harrison, Jason, Will:

- Andy is able to complete 10 story points a week
- Eddie - 6 story points a week
- Harrison - 8 story points a week
- Jason - 13 story points a week
- Will - 6 story points a week

Developers will work for 7 days a week and the length of the sprint is 1 week

Team Velocity

The team velocity will be 43 story points a week.

Provisional Schedule

Day 1: November 13th

Day 7: November 20th

Tas ks	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
6a			Eddie: 3				
6b				Eddie: 3			
6c							
7a					Andy: 2		
8a	Jaso n:2		Jason: 2				
8b		Jason:2	Jason: 2				

8c			Jason:2		Jason: 2		
9a					Andy: 2		
9b					Andy: 1	Andy: 1	
9c						Andy: 1	
9d						Andy: 2	
11a	Harri son: 2	Harrison : 2					
11b			Harrison: 1				
11c			Harrison: 1				
11d			Harrison: 1				
13a				Harris on: 1			
13b					Harrison : 2		
14a				Jason: 2			
14b					Jason:2		
14c						Jason:2	

This schedule focuses on completing the unfinished tasks from Sprint 3 first, before starting on the sprint 4 tasks. By the end of this sprint, there will be a

more unified user experience to access all previously implemented features and new ones in the sprint backlog for sprint 4.

Sprint Report

Tas ks	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
6a							
6b							
6c							
7a							
8a			Jason: 2				
8b			Jason: 2				
8c					Jason: 2		
9a							
9b							
9c							
11a	Harri son: 3	Harrison : 3					
11b			Harrison: 1				
11c			Harrison: 1				
11d			Harrison: 1				
13a				Harris on: 1			

13b					Harrison : 3	Harrison: 4	
14a						Jason: 1	
14b						Jason: 2	
14c						Jason:1	

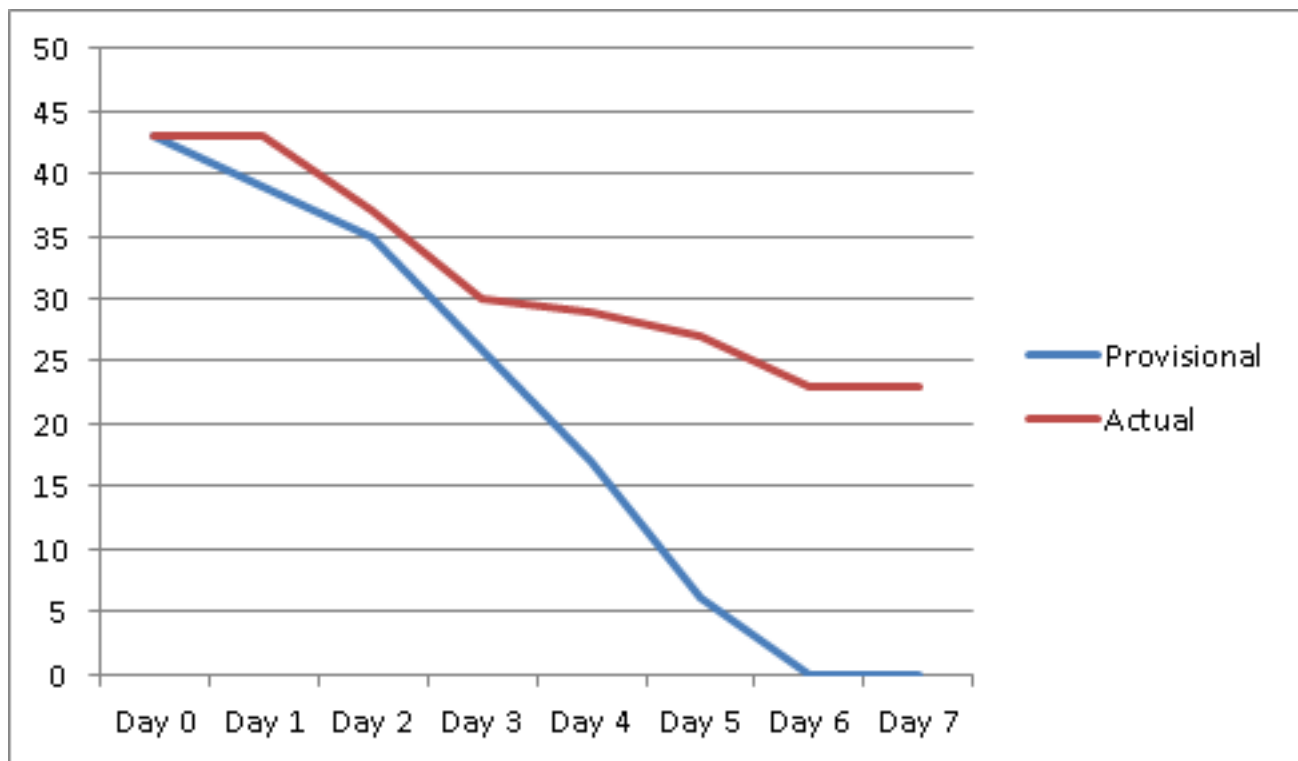
At the end of this sprint, we finished implementing features mentioned in the sprint backlog for sprint 4, all unit tests and validation tests and implementation of a more unified user experience by adding menus for the user to navigate through the software and explore features more easily, on top of other.

Overall, sprint 4 was more productive than sprint 3 because we realized we had to make up for the lack of work done in sprint 3.

Note: The user story 10 row is removed from this table because user story 10 was already completed in sprint 3.

Burndown Chart

Story
Points



Days

Snapshots of Taskboard

Start of Sprint

Each list has the task number, number of story points and the person responsible for the task. For example, the first one is task 6b, which requires 2 story point and is allocated to Eddie. Y = Andy, J = Jason, ZL = Eddie, W = William, HF = Harrison. These are the same as the end of sprint 3.

Sprint Backlog	Current Work
6b) [2 story points] Implement methods to add, edit and remove questions ZL	6a) [2 story points] Repurpose the Question Creation UI to allow the professor to view, edit and remove questions. ZL
6c) [2 story points] Connect the repurposed front-end UI to the newly implemented back-end method. Depends on a, b ZL	8a) [2 story points] Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer. J
8b) [2 story points] Implement the back end methods to pull data from the csv file that stores the questions of an assignment and compares the student's answer with the one in the csv when they hit the submit button. J	9a) [2 story points] create a button in the AssignmentDashboard GUI to be displayed beside an assignment name that will take you to the 'visibility options menu', with radio buttons for visibility on and visibility off. Y
8c) [2 story points] AssignmentAttempts.csv will hold all the results of an assignment, overwriting the respective row with the most recent attempt. J	11a) [2 story points] Implement methods to iterate through the .csv file to find the particular assignment the student is working on. HF
9b) [2 story points] implement the back end methods to pull an assignment from the .csv that stores all the assignments and change it's attribute for visibility depending on what the professor chose. Y	
9c) [3 story points] connect the front and back end together. Dependent on a, b Y	
11b) [3 story points] implement methods to extract the student's answers from AssignmentAttempts.csv, and compare these answers with the calculated answer in part a). Dependent on a) HF	

Completed Work

1a) [1 story point] Design a form that has an email, password field and a login button to allow the professor to log into the system.

Y

1b) [2 Story points] Implement Professor class to generate professor objects that store the information of a single professor, along with its associated methods. (ie. getters and setters for the Professor).

Y

1c) [1 story point] Create an event handler and methods so that when the submit button is clicked, information is taken from the email, password fields and a new professor is written to the appropriate txt file for storage. Depends on a and b

Y

2b) [1 story point] Create a UI using Tkinter to allow a Student to register their account using their name, email, student number and password.



J

3a) [1 story point] Design and implement the UI for allowing a professor to choose a type of question to create.

ZL

Completed Work

3b) [1 story point] Create a general Question class that includes attributes about a question such as the question_id, type of question, question body, formula and its associated methods to add a question to a file for storage.

ZL

3c) [1 story point] Add subclasses for the Question class to encompass all the types of questions that can be made. Depends on a

ZL

3d) [3 story points] Design and implement forms / windows for the different types of questions (Multiple choice, fill in the blanks)

ZL

3e) [2 story points] Connect the front end UI elements to the back end methods. Depends on a, b, c

ZL

Completed Work

4c) [1 story point] Implement method to randomize the given variables of a question each time it's retrieved from the .txt file and inserted into another. Depends on a, b

W

5a) [3 story points] Implement methods to retrieve questions from a .txt file and insert them into another .txt file that will contain the questions for a particular assignment.

HF

5c) [1 story point] Keep track of how many questions are chosen from the .txt file.

HF

5d) [1 story point] Design and implement a UI that shows all the questions that have been chosen for an assignment.

HF

5e) [1 story point] Connect the front end and back end together. Depedency: a, b, c, d

HF

Add a card...

Completed Work

5e) [1 story point] Connect the front end and back end together. Depedency: a, b, c, d

HF

1d) [1 story point] Create a form that will display all the information about that professor to the screen.

Y

2a) [1 story point] Implement Student class to generate student objects that store the information of a single student, along with its associated methods. (ie. Getters/setters, txt helper methods)

J

4a) [3 story points] Implement methods to parse a formula provided by a professor that is associated with a particular question.

W

7a) [2 story points] Implement a new class OptionsMenu.py that will check whether a user is a professor or student.

Y

Add a card...

<div> <div>Completed Work</div> <div> <div>8c) [2 story points] AssignmentAttempts.csv will hold all the results of an assignment, overwriting the respective row with the most recent attempt.</div> <div>J</div> </div> <div> <div>10a) [2 story points] Add a button that takes the AssignmentDashboard GUI menu for the professor beside each assignment.</div> <div>W</div> </div> <div> <div>10b) [3 story points] Implement back-end methods so that when the time length is set in the SetTimer GUI, and the timer is up, the visibility will be automatically turned off for an assignment.</div> <div>W</div> </div> <div> <div>10c) [2 story points] Connect the front end to the back end. Dependent on a, b</div> <div>W</div> </div> <div>Add a card...</div> </div>	<div> <div>Completed Work</div> <div> <div>10c) [2 story points] Connect the front end to the back end. Dependent on a, b</div> <div>W</div> </div> <div> <div>14a) [2 story points] Design and implement a list layout using Tkinter to display all the assignments, with it's name, due date, number of questions and assignment ID for a specific student.</div> <div> <div>🔔 2</div> <div>J</div> </div> </div> <div> <div>14b) [2 story points] Implement back end method to retrieve data from the csv file, format and display this data to the implemented UI. (dependent on a)</div> <div>J</div> </div> <div> <div>14c) [2 story points] Connect the front end UI to the back end methods (depends on a and b)</div> <div> <div>🔔 2</div> <div>J</div> </div> <div> <div>Main class for connecting all the current classes</div> <div>HF</div> </div> </div> </div>
--	---

Completed Work

11a) [2 story points] Implement methods to iterate through the .csv file to find the particular assignment the student is working on.

HF

11b) [3 story points] Implement methods to extract the student's answers from AssignmentAttempts.csv, and compare these answers with the calculated answer in part a).
Dependent on a)

HF

11c) [1 story point] Design and implement a UI using Tkinter that will display the results of a students answers when compared to the actual answer.

HF

11d) [1 story point] Connect the front-end and the back-end.
Dependent on a, b, c

HF








8a) [2 story points] Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer.

J

11) Create Assignment.py and Question.py to make the whole program object-oriented

HF

End of Sprint

Sprint Backlog	Current Work	Completed Work
<p>9b) [2 story points] implement the back end methods to pull an assignment from the .csv that stores all the assignments and change it's attribute for visibility depending on what the professor chose.</p> <p> 1 </p> <p>Y</p>	<p>6c) [2 story points] Connect the repurposed front-end UI to the newly implemented back-end method. Depends on a, b</p> <p>ZL</p>	<p>1a) [1 story point] Design a form that has an email, password field and a login button to allow the professor to log into the system.</p> <p>Y</p>
<p>9c) [3 story points] connect the front and back end together. Dependent on a, b</p> <p> 1 </p> <p>Y</p>	<p>6b) [2 story points] Implement methods to add, edit and remove questions</p> <p>ZL</p>	<p>1b) [2 Story points] Implement Professor class to generate professor objects that store the information of a single professor, along with its associated methods. (ie. getters and setters for the Professor).</p> <p>Y</p>
<p>Add a card...</p>	<p>9a) [2 story points] create a button in the AssignmentDashboard GUI to be displayed beside an assignment name that will take you to the 'visibility options menu', with radio buttons for visibility on and visibility off.</p> <p> 2 </p> <p>Y</p>	<p>1c) [1 story point] Create an event handler and methods so that when the submit button is clicked, information is taken from the email, password fields and a new professor is written to the appropriate txt file for storage. Depends on a and b</p> <p>Y</p>
	<p>21 [3 story point]As a student (Sara), I want to know the mark of all the tests and the solution of the last test done. Create class to show all the mark</p> <p>ZL</p>	<p>2b) [1 story point] Create a UI using Tkinter to allow a Student to register their account using their name, email, student number and password.</p> <p></p> <p>J</p>
	<p>Merge the assignment making feature into the master to incorporate the changes made in the format of the questions and assignments</p>	<p>3a) [1 story point] Design and implement the UI for allowing a professor to choose a type of question to create.</p> <p>ZL</p>

Completed Work

3b) [1 story point] Create a general Question class that includes attributes about a question such as the question_id, type of question, question body, formula and its associated methods to add a question to a file for storage.

ZL

3c) [1 story point] Add subclasses for the Question class to encompass all the types of questions that can be made. Depends on a

ZL

3d) [3 story points] Design and implement forms / windows for the different types of questions (Multiple choice, fill in the blanks)

ZL

3e) [2 story points] Connect the front end UI elements to the back end methods. Depends on a, b, c

ZL

Completed Work

4c) [1 story point] Implement method to randomize the given variables of a question each time it's retrieved from the .txt file and inserted into another. Depends on a, b

W

5a) [3 story points] Implement methods to retrieve questions from a .txt file and insert them into another .txt file that will contain the questions for a particular assignment.

HF

5c) [1 story point] Keep track of how many questions are chosen from the .txt file.

HF

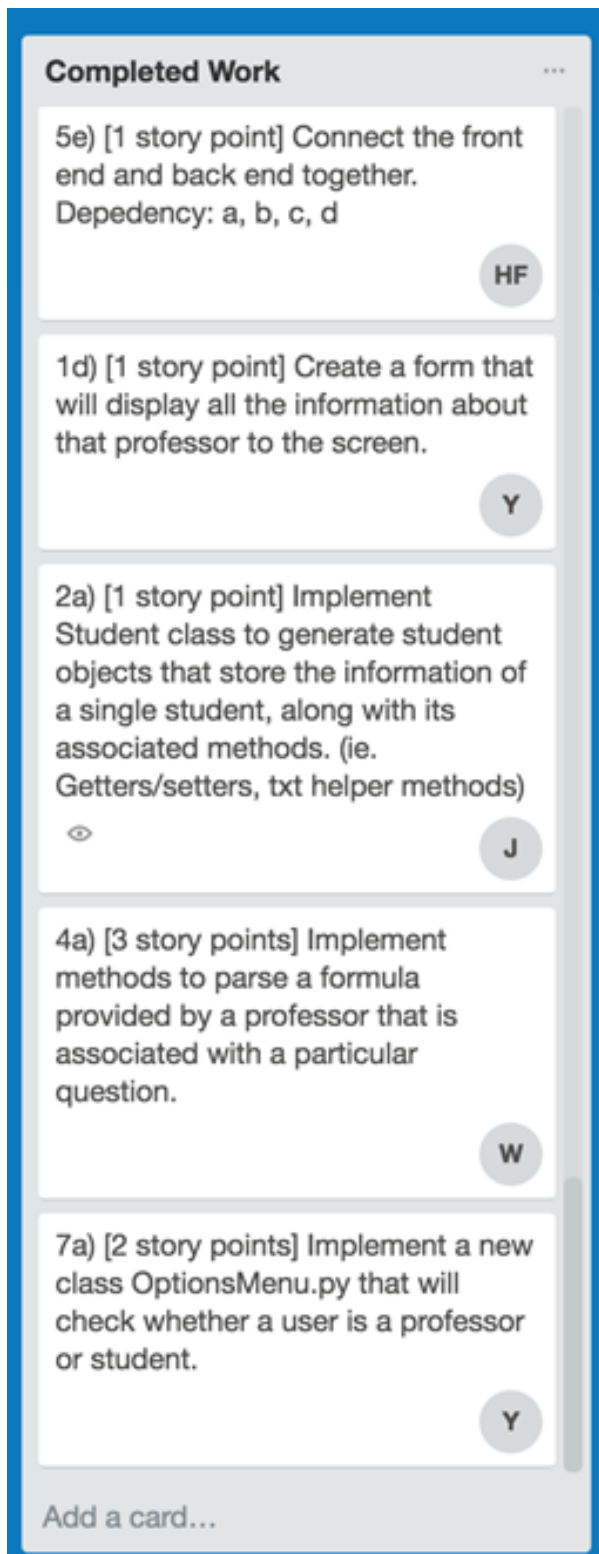
5d) [1 story point] Design and implement a UI that shows all the questions that have been chosen for an assignment.

HF

5e) [1 story point] Connect the front end and back end together. Dependency: a, b, c, d

HF

Add a card...



Completed Work

8c) [2 story points] AssignmentAttempts.csv will hold all the results of an assignment, overwriting the respective row with the most recent attempt. J

10a) [2 story points] Add a button that takes the AssignmentDashboard GUI menu for the professor beside each assignment. W

10b) [3 story points] Implement back-end methods so that when the time length is set in the SetTimer GUI, and the timer is up, the visibility will be automatically turned off for an assignment. W

10c) [2 story points] Connect the front end to the back end. Dependent on a, b W

Add a card...

Completed Work

10c) [2 story points] Connect the front end to the back end. Dependent on a, b W

14a) [2 story points] Design and implement a list layout using Tkinter to display all the assignments, with it's name, due date, number of questions and assignment ID for a specific student. J 2

14b) [2 story points] Implement back end method to retrieve data from the csv file, format and display this data to the implemented UI. (dependent on a) J

14c) [2 story points] Connect the front end UI to the back end methods (depends on a and b) J 2

Main class for connecting all the current classes HF

Completed Work

11a) [2 story points] Implement methods to iterate through the .csv file to find the particular assignment the student is working on.

HF

11b) [3 story points] Implement methods to extract the student's answers from AssignmentAttempts.csv, and compare these answers with the calculated answer in part a).
Dependent on a)

HF

11c) [1 story point] Design and implement a UI using Tkinter that will display the results of a students answers when compared to the actual answer.

HF

11d) [1 story point] Connect the front-end and the back-end.
Dependent on a, b, c

HF

8a) [2 story points] Design and implement a UI using Tkinter to allow the student to see the questions as a list and have a corresponding text field to type their answer.

J

11) Create Assignment.py and Question.py to make the whole program object-oriented

HF

Verification and Validation

Verification: All unit tests for methods in this project are placed under the tests/ directory.

Validation:

In order to perform an acceptance test based on our user stories, we were fortunate to have a stats major as well as multiple students who have taken STAB22 to test out the software. To perform our acceptance test, we first gave the users a list of completed user stories and made sure that they understood each one. From that, each user had an understanding of what the software should do. We then gave the users a copy of the software to use and asked them to perform the user stories they read and tell us the accuracy in how well the software did what it was supposed to. From doing this acceptance test, we realized that all students were able to perform the functionality outlined by the given user stories, however, they indicated that there should also be iterations certain aspects of the software such as design.

Code Review Strategy:

Our team's guidelines to conduct code reviews are as follows. We decide to meet in person because we realized that as a team, we are more productive when we are working along side each other and it is easier to communicate. Once we've met up, we sit in a roughly circular form and we code reviewed the person to the right of us.

Some things that we look for in particular during the code review is the style of code, comments, readability and maintainability, functionality, modularity, tests and ability to use design patterns when possible. We created a checklist for this so that it would be easier for us to identify and go through them when actually reviewing our team member's code.

Once we defined our strategy, we looked at each other's branches from their respective computers and aimed to review approximately 200 lines of code, which lasted roughly about 1 hour.

Code Review Summary:

For Eddie's Code:

In the user_story_3.py in the user_story_3_different_question_type branch, 2 functions named FBQ() and MCQ() do essentially the same thing with very minute differences. Most of the code inside the functions were copy and pasted so a design pattern could be used to solve this and make it more maintainable. The only difference in functionality was changing the window title and a the value of a csv value being inserted to indicate the type of question. Eddie also has many indentation errors and needs more comments and docstrings to increase readability. The code works as it's supposed to.

For Will's Code:

Will's code is working, but some things can improve.

His code is not following the object-oriented concept, and it is a little bit difficult to connect his code to others' related code.

His code only handles questions with the simple operators (+-*/), it will crash when using the complicated operator. There is also a lack of comments.

For Harrison's code:

His code works fine, but it has too many instance for his SelectQuestions class. When I check his code i feel a little confused about all the instance, he actually create some useless instances. And for each instance the comments is missing or its not clear to the others who looks at his code. So when I try to modify or use the instances he has, it is difficult to edit and I kind of lost to how to connect the other part with his code.

Perhaps move the CSV Helper functions into another file, as it doesn't directly relate to how Users work?

For Jason's code:

The ProfessorProfileIndex and StudentProfileIndex have shared functions that could be made to be inherited from a shared class called, presumably, UserProfileIndex.

The Student and Professor Classes have functions (insertProfessor and insertStudent) that are just calling the version in their parent, insertUser, of the User class. As the users of the function don't see the functions themselves, those functions that just call parents functions could just be removed and the parent's functions called instead.

Professors and Students also both use a list of some kind that operate similarly, storing courses and assignments, specifically. It may be possible to create code in User that does the same thing, eliminating redundancy.

For Andy's code:

Andy's code works fine, there are style errors which should be changed to follow PEP-8. Andy also needs to remove useless comments to improve readability. For a lot of the comments, he's just talking to himself instead of explaining what your code does and why you chose this way of doing it. There is also a strange import for "import tkinter.messagebox". Since he already imported everything in the tkinter library ("from tkinter import *"), this import statement is redundant.

Code Debriefing Meeting:

The code debriefing meeting can be found on the GitHub repo:

https://github.com/CSCC01F17/L02_04/tree/master

The video is called CSCC01 Code Review Debriefing Meeting.mp4.

Download and play.