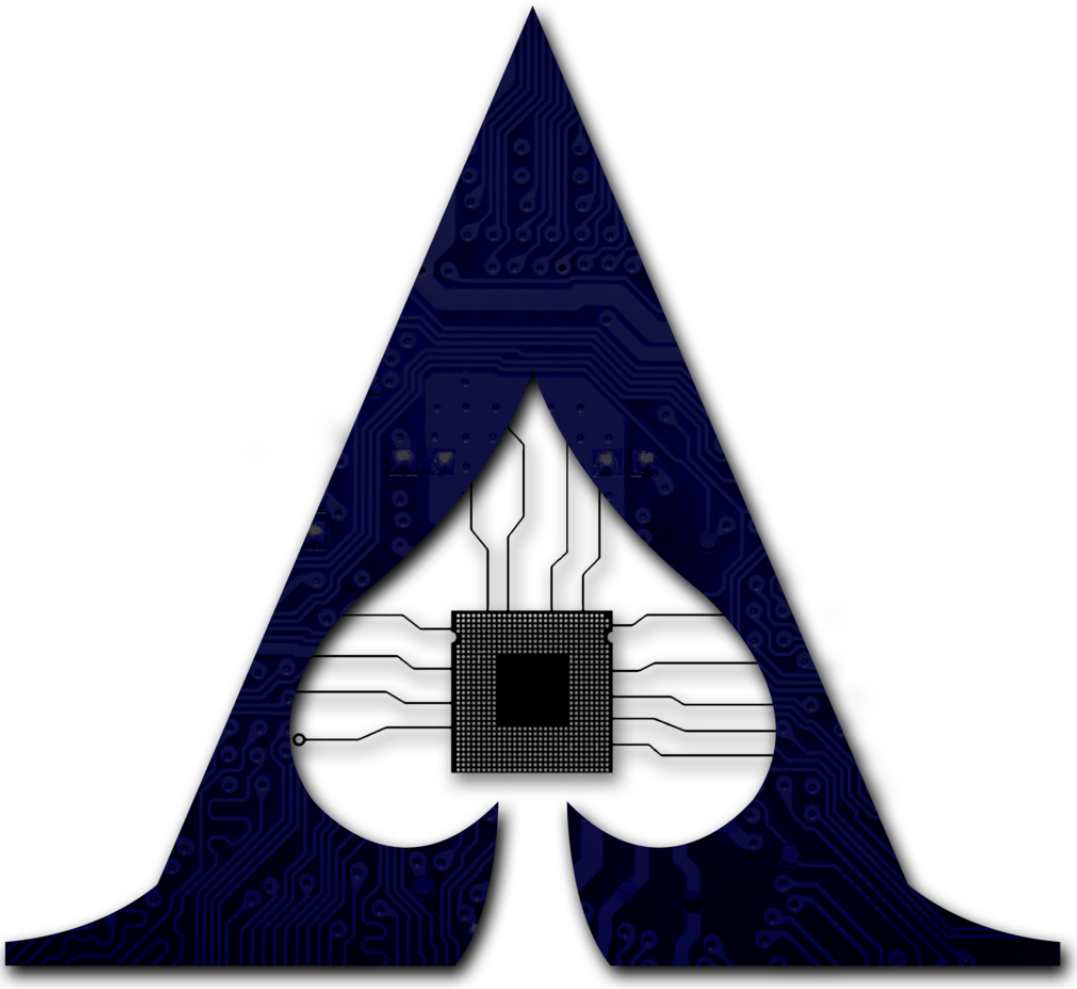
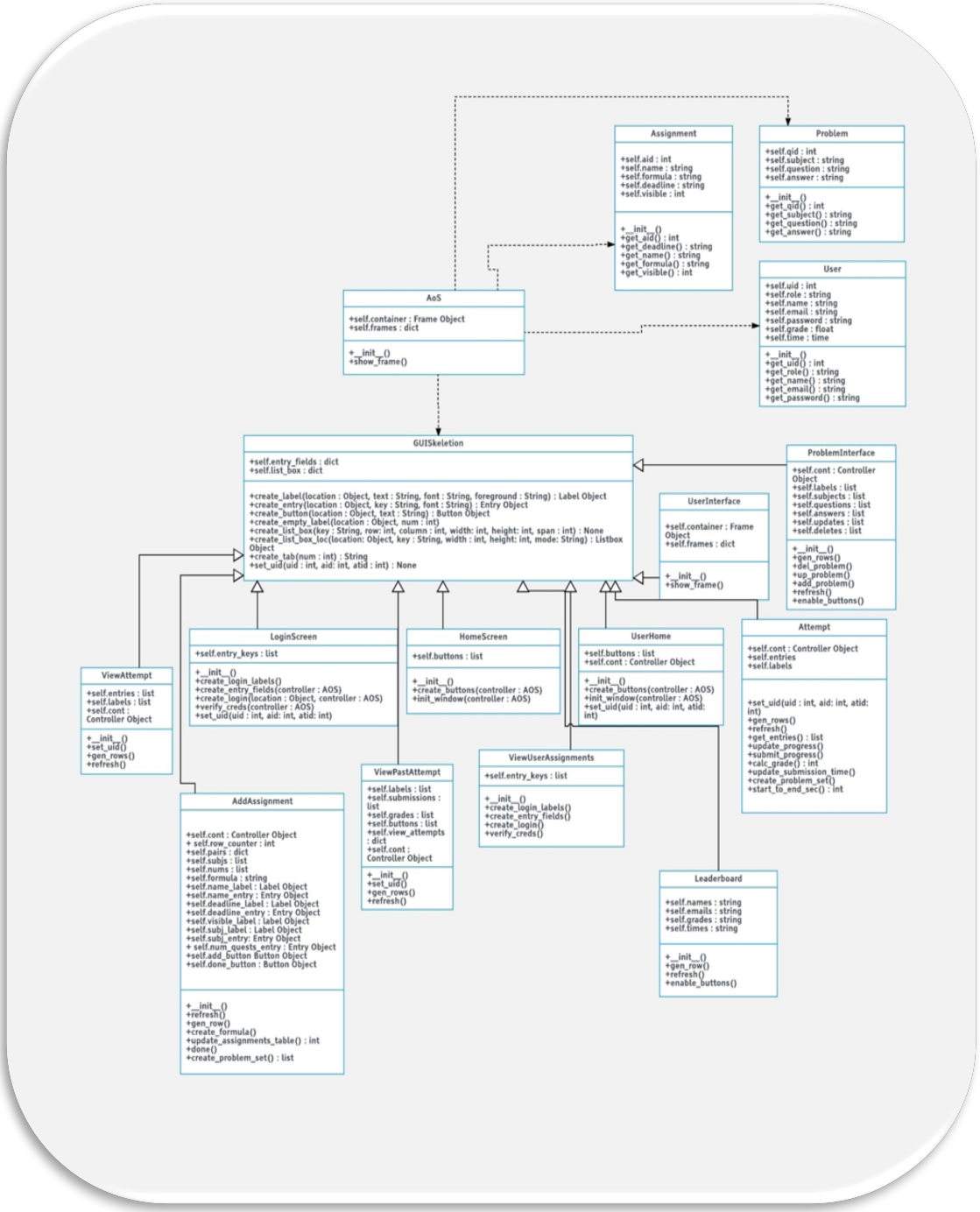
Deliverable 5

Ace of Spades



Diana Soto, Gerrit Steinbach, Bar Slutsky,
Kezaram Tharmasegaram, Jonathan Jung-Kyun An

UML Diagram- Sprint 4 & Sprint 5 Focus



Product Backlog (Admin & Student)

- ✓ As Nate/Sohee**, I** would like to be able to Login to the admin part of the application
- ✓ As Nate/Sohee, I would like to be able to add problems
- ✓ As Nate/Sohee, I would like to be able to remove problems
- ✓ As Nate/Sohee, I would like to be able to edit problems
- ✓ As Nate/Sohee, I would like to be able to add users to the software
- ✓ As Nate/Sohee, I would like to be able to edit existing users in the software
- ✓ As Nate/Sohee, I would like to be able to delete users from the software
- ✓ As Nate/Sohee, I would like to be able to create a assignment.
- ✓ As Nate/Sohee I would like to be able to set a due date for an assignment.
- ✓ As Nate/Sohee, I would like to be able to view all and individual student results on specific assignments.
- ✓ As Nate/Sohee, I would like to be able to view a student's grade.
- ✓ As Nate/Sohee, I would like to be able to sort student assignment results by grade, student uid, and number of attempts for an assignment.
- ✓ As Nate/Sohee, I want to be able to see a leaderboard of my student sorted by highest to lowest overall grade then, in case of a tie, lowest to highest completion time."
- ✓
- ✓ As Sohee I would like to be able to change due date for an assignment.
- ✓ As Brian, I would like to be able to Login as a student to the application
- ✓ As Brian, I would like to be able to view posted assignments
- ✓ As Brian, I would like to be able to complete an assignment
- ✓ As Brian, I would like to be able to receive immediate results for my submission
- ✓ As Brian, I would like to be able to view info about assignments (e.g. due dates and progress)
- ✓ As Brian, I would like to be able to save my progress in the middle of work, so that I can close the app and come back to it later.
- ✓ As Brian, I would like to be able to review my entire work after submission.
- ✓ As Brian I want to be able to have a list of all my past attempts for any particular assignment so I can view my grade, submission date and a button option to view attempt progress"

**Pending*

- As Nate/Sohee I would like to be able to edit a student's assignment mark.

Personas & User Stories

Admin

Professor Nate River

- *Male, Mid 40s.*
- University of Toronto Professor - Architectural Design.
- Very busy with meetings, doesn't have much time in between lectures.
- Most of the grading is done by the TAs after he provides a rubric.
- Large office, but very messy with papers.

- Does not spend a lot of time with students.
- Does not hold many office hours.
- Knows how to use computers well.
- Likes smart-phones because he does not have to use the computer.
- Does not want to spend a lot of time on the computer.
- Likes marking on paper.
- Gives a lot of informative feedback on student's submissions.
- Serious about his work.
- Very meticulous about the course running smoothly, and passes off announcing/dealing with technical delays to the TA's.
- Does not want to deal with any program errors, but would have the developers answer students directly about any issues.

Professor Sohee

- Female, mid 50s
- Professor of Statistics at University of Toronto
- Speaks English fluently
- Not knowledgeable of advanced computer systems and operations
- Will use a software if it's simplified, and there is a way for her to get support
- Holds office hours for students as scheduled on a weekly basis
- Enjoys students engaging with their homework and would like them to have an easier tool to complete homework online
- Administers a number of courses at the University, therefore very busy schedule, relies on organized and planned routines to get through tasks per day
- Prefers to students to get immediate feedback on their homework, making marking more efficient.

Student

Brian Thompson

- *Male, 20 years old.*
- 2nd year computer science specialist.
- Commutes to school every day, total of 2 hours.
- Comfortable doing school work on the bus, and pretty much anywhere.
- Spends a lot of time planning learning-strategies because he needs structure when studying.
- Good at managing his time, and likes to be able to estimate tasks' lengths.
- Likes math and comfortable with quantitative subjects.
- Very good with computers and technology in general.
- Likes tinkering and being able to customize things.
- Uses sites like Khan academy and likes the benefits from the concept of going through material, followed by questions that cover that material.
- Cares a lot about achievements, and a perfectionist.
- Gets Highly motivated by positive feedback
- Starts assignments as soon as assigned, tries to finish as fast as possible before starting anything new.

- Will always go the extra mile in order to maximize his results. Welcomes any opportunity for extra work.
- Would be more engaged with online learning material if there were questions that had hints and additional resources, such as the textbook, where could refer to for extra help.

Sprint #4: November 7th - November 12th

Story 1: (Total = 6 points) Kezaram

"As a student, I want to be able to have a list of all my past attempts for any particular assignment so I can view my grade, submission date and a button option to view attempt progress"

- Task 1: (2 points)
  Retrieve grade and submission date for each attempt from specific assignment table according to the assignment button the user clicked.

- Task 2: (2 points)
  Create Date of Submission + Grade labels and View Attempt buttons dynamically on GUI. This was based of the number of attempts the user had for a particular assignment.
  – Dependency T1

- Task 3: (2 points)
  Connect the GUI to data from database. When the user clicks the "Past Attempts" button, their screens should be switched to a screen that creates the labels and view buttons automatically.
  –Dependency T1 & T2

Story 2: (Total = 11 points) Jon

" As an admin, I want to be able to see a leaderboard of my student sorted by highest to lowest overall grade then, in case of a tie, lowest to highest completion time"

- Task 1: (1 points)
  Create functionality that retrieve a list of all students with the following order:
  -The highest to lowest grade
  -In case of a tie in grades, lowest to highest overall completion time.
  Update user and user's table to have "grade" and "time" columns.

- Task 2: (2 points)
  -Create a screen for the leaderboard that will dynamically create entries for every student in the database.

- Task 3: (2 points)

-Create functionality that calculates the number of seconds passed between two points in time according to the date/time format used in the application.

- Task 4: (3 points)
  -Calculate user's total time and average grade for every latest submission for all assignments.
  –Dependency T3

- Task 5: (1 point)
  Every time student makes a new submission, update users' time and grade in the database.
  –Dependency T3 & T4

- Task 6: (2 points)
  Connect GUI to the database functionality. When user navigates to Leaderboards screen, it should display a list of users by the
  –Dependency on T1->T5

Story 3: (Total = 7 points) Bar

"As a student, I want to be able to work on an assignment by inputting my solution and having the ability to both save my progress and submit for marking"

- Task1: (0.5 points)
  add "save" and "submit" buttons to GUI

- Task2: (2.5 points)
  create function for save button: function will fetch the texts from all the entries in the frame, (make all entries into list so can iterate and get all texts) then it will create a string in format: 'ans1','ans2',...and store that string into the attempt row in the a# table for that user

- Task 3: (3 points)
  write a function to compare each answer to the one in the database and create an algorithm for calculating the grade.
  create submitting function which will update the progress(task1), grade and submission time columns in the a# table

- Task 4: (0.5 points)
  make the row generation function update the entries with existing progress upon creation

- Task 5: (0.5 points)

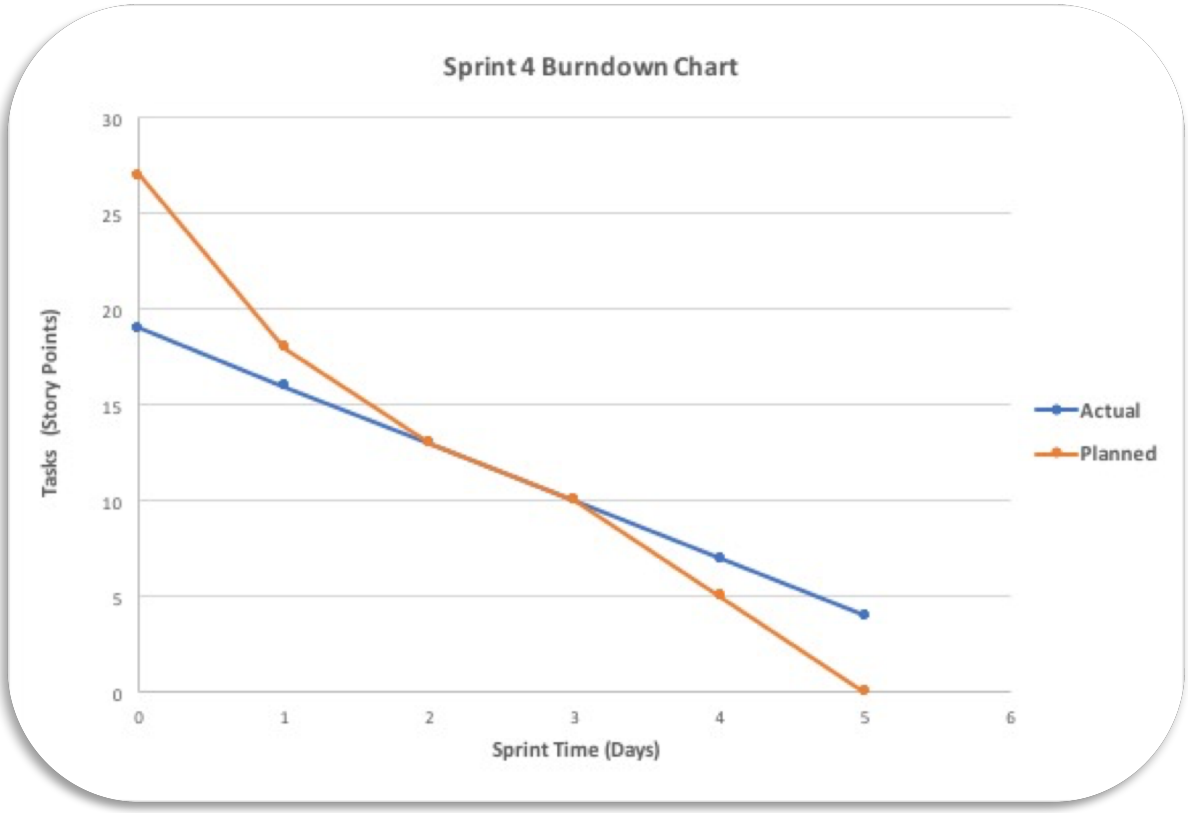
create the links between the buttons in the “view user attempts” menu and the corresponding functions to display the correct new screen

Story 4: (Total = 3 points) Bar

As a student, I want to be able to see a specific attempt for an assignment.
–Dependency S1

- Task 1: (2.5 points)
  Create a class based on the view attempt class, which will have labels instead of entries It will also include the functionality for submission and progress saving.

- Task 2: (0.5 points)
  create the links between the buttons in the “view past attempts” menu and the corresponding functions to display the correct new screen



Sprint #5: November 13th - November 20th

Developing

S2 Continued (6 points)- Jon

"As an admin, I want to be able to see a leaderboard of my student sorted by highest to lowest overall grade then, in case of a tie, lowest to highest completion time."

- Task 1: (3 points)
  Calculate user's total time and average grade for every latest submission for all assignments.
- Task 2 : (1 point)
  Every time student makes a new submission, update users' time and grade in the database.
  –Dependency T1
- Task 3 : (2 points)
  Connect GUI to the database functionality. When user navigates to Leaderboards screen, it should display a list of users by the
  –Dependency T1 & T2

Story 5: (Total= 8.5) Diana + Gerrit

"As an admin, I want to be able to see all student grades and progress for any given assignment"

- Task 1: (0.5 points)
  Create a class for ViewStudentGrades which is where the GUI screen will be coded and frame to main.py

- Task 2: (2 points)
  Create database function to get users that have worked on an assignment

- Task 3: (6 points)
  Code GUI labels, buttons, entries, listbox and dropdown to work with the database functions to show the admin grades and progress for a selected assignment from the dropdown
  –Dependency Story T1 & T2

Story 6: (Total=) Gerrit + Diana

"As an admin, I want to be able to filter the results from the previous screen according to username."

- Task 1: (1 point)
  Add filter, and sort buttons/labels to the GUI from S5
  –Dependency Story 5

- Task 2: (3 points)
  Update the existing table created in T1
  Create a clear filter method that restores the table to its original state
  Create methods to see

- Task 3: (2 points)
  Connect methods to GUI for admin to edit see filtered results in the listbox without switching screens, update the database and the screen after changes

Testing

Test "Manage Problems" Cases (3 points)- Kezaram

- Task 1 : TestProblem
  - test_init = Initialize extreme cases of a Problem Object
  - test_get_qid = Tests if correct value and type returned
  - test_get_subject = Tests if correct value and type returned
  - test_get_question = Tests if correct value and type returned
  - test_get_answer = Tests if correct value and type returned
  -
- Task 2: TestAddingProblem
  - test_empty_subject_add = Tests if a problem's subject can be added with empty string
  - test_empty_question_add = Tests if a problem's question can be added with empty string
  - test_empty_answer_add = Tests if a problem's answer can be added with empty string
  - test_add_valid_answer = Tests if a valid problem can be added

- Task 3: Test RemovingProblem
  - test_delete_problem = Tests if valid problem can be deleted
  - test_delete_non_existant_problem = Tests if invalid problem can be deleted

- Task 4: TestUpdatingProblem
  - test_empty_subject_update = Tests if a problem's subject can be updated with empty string
  - test_empty_question_update = Tests if a problem's question can be updated with empty string
  - test_empty_answer_update = Tests is a problem's answer can be updated with empty string
  - test_update_non_existant_problem = Tests if a invalid problem can be updated

Test "View User Assignment" Cases (3 points) Kezaram

- Task 1: TestViewUserAssignments
  - test_view_assign_details = Tests if correct assignment details returned
  - test_view_assign_name_type = Tests if correct type returned
  - test_view_assign_formula_type = Tests if correct type returned
  - test_view_assign_deadline_type = Tests if correct type returned
  - test_view_assign_visibility_type = Tests if correct type returned
  - test_view_non_existant_assign = Tests if invalid assignment can be viewed

- Task 2: TestAddUserAttempts
  - test_add_attempt = Tests if a new attempt can be added
  - test_add_empty_attempt = Tests if an empty attempt can be added
  - test_duplicate_submission_dates = Tests if submission dates are duplicated

  - test_add_attempts_for_non_existant_assign = Tests if invalid assignment attempt can be added

- Task 3: TestViewUserAttempts
- test_get_user_attempts = Tests if correct user attempts are retrieved
- test_get_non_existant_nth_attempt = Tests if invalid nth attempt can be retrieved
- test_get_user_nth_attempt = Tests if valid nth attempt can be retrieved
- test_get_user_assignment_progress = Tests if correct progress info of assignment retrieved
- test_view_graded_submission_upon_submit = Tests if grade and submission date co-exist

Test "Login" Cases (2 points) Bar

- test_email_blank – Try logging in with blank email
- test_email_not_in_system – Try logging in with email that is not stored in the database
- test_password_blank – Try logging in with a blank password
- test_password_wrong – Try logging in with the wrong password for an existing email
- test_both_email_password_blank – Try logging in with both fields blank
- test_correct_combo_admin – Try logging in with the right credentials for an admin user
- test_correct_combo_student – Try logging in with the right credentials for a student user

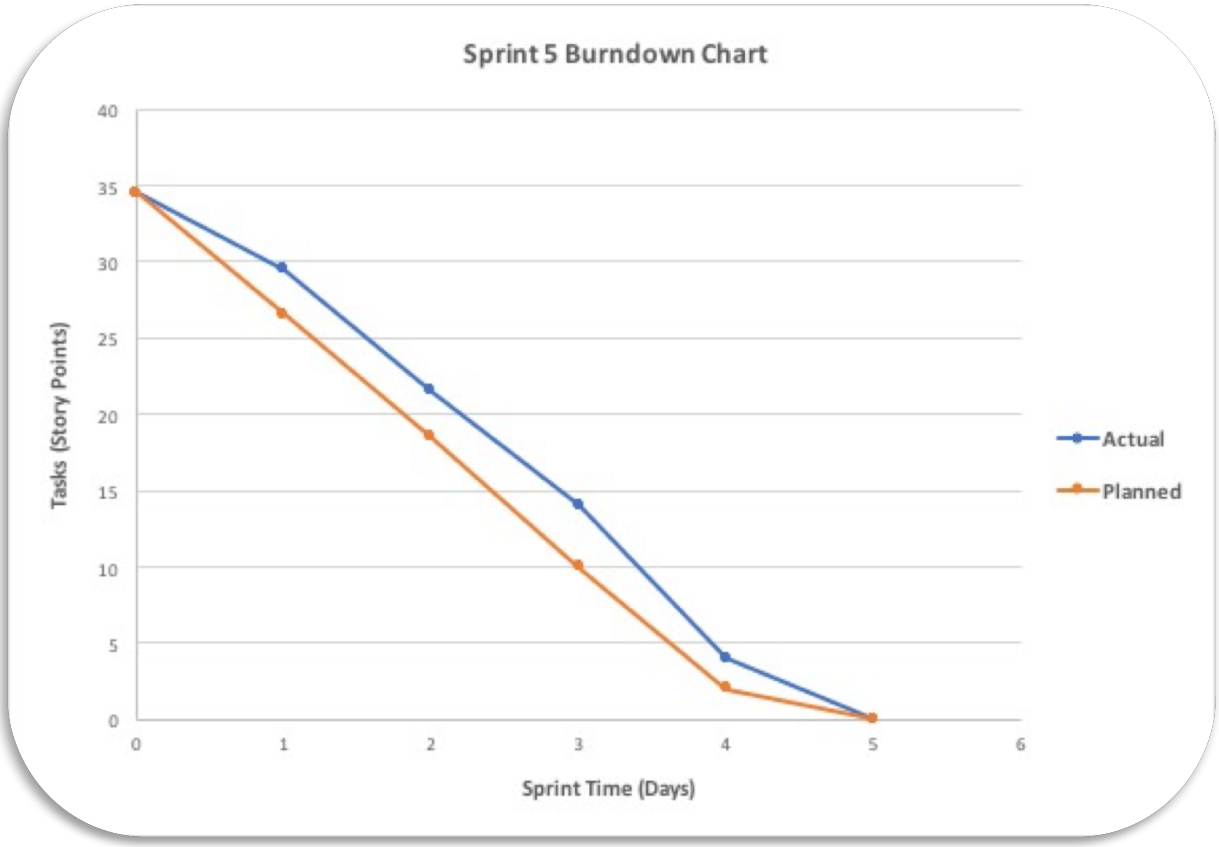
Test "Add User" Cases (3 points) Bar

- test_empty_role_add – Try adding a new user with a blank role
- test_empty_name_add – Try adding a new user with a blank name
- test_empty_email_add – Try adding a new user with a blank email address
- test_empty_role_update – Try updating a user's details with a blank role
- test_empty_name_update – Try updating a user's details with a blank name
- test_empty_email_update – Try updating a user's details with a blank email address
- test_invalid_role_add – Try adding a new user with a role other than student or admin
- test_invalid_role_update - Try updating a user's role to a new role other than student or admin
- test_delete_user – Try deleting a user
- test_add_user – Try adding a user with valid information

Test "Create Assignment" Cases (3 points) Bar

- test_empty_name – Try creating a new assignment with an empty name
- test_empty_deadline – Try creating a new assignment with an empty deadline
- test_empty_visible – Try creating a new assignment with an empty visibility indicator
- test_empty_subject – Try adding a subject row to the assignment without indicating a subject
- test_empty_number_questions – Try adding a subject row to the assignment without indicating a number of questions
- test_non_natural_number_questions – Try adding a subject row to the assignment with input that is not a positive integer

- test_too_many_number_questions – Try adding a subject row to the assignment with a number of questions that is greater than the available number of questions for that subject
- test_non_existing_subject – Try adding a subject row to the assignment with a subject that is not associated with any problem in the database
- test_assignment_name_taken – Try creating a new assignment with a name that already
- exist in the database



Code Review

Guideline

- Quality of Documentation- PEP8
- Can lines of code be simplified/eliminated
- Bugs what is the impact? What are the solutions?s
- Error Handling
- Cleanliness of code
- Re-factoring. Is there repeated code/functions

Jon's Review

- Code needs to be adjusted to follow PEP8

- There are numerous areas where blocks of codes are commented out; there is a need to review whether they are still relevant to the needs of the application.
- Purpose of methods need to be clearly indicated.
- Docstring should be added to each function
- Certain docstrings need to be edited to more accurately describe the purpose of the function
- Certain docstrings are unedited copy and pasted text from other functionalities.
- Certain functions either need to accept more general range of inputs or limitations must be clearly stated and accounted for in the functionality.
    - Ex: The application currently only properly creates new Assignments with names in the format “a#”.
- When creating a new Assignment, it shouldn’t allow the admin to add subjects that don’t exist.
- Functions should follow a certain format for consistency:
    - Ex. All functions that create a screen should follow the format “FooScreen”

Kezaram’s Review:

- We can have a universal back button. This can be a separate method instead of multiple calls.
- A lot of repeated methods such as .gen_rows(), .refresh() and .enable_buttons(), etc in all classes causes clunkiness.
- verify_cred() can be broken into smaller methods: One that can validate the user against the role. One that clears the entry_fields. Or separate out to a LoginClass.
- gen_rows() can be broken into smaller methods: One that creates the widgets. One that places the widget on GUI.
- Switching frames is also inefficient. We can have a separate class with single responsibility to switch frames that pass core info around.
- Initializing GUI classes can be hard while unit-testing as they don’t return anything.
- Stay consistent with .pack() and .grid() widget managers?
- Create custom Exception classes unique to our software to allow more fluidity.
- There are some methods that have not been documented yet such as gen_rows() or done().
- A lot of unnecessary comments of old functions can be taken out.
- Using “from import ____” is considered bad practice. Also, only import files that are needed, otherwise the file becomes big.

Gerrit’s Review

- Using the built in methods from GUISkeleton to create the buttons and labels
- Creating proper docstrings with documentation for each parameter that is being called
- When calling functions from other classes having a comment that describes the output of the function called, so it’s easier to manipulate the code without having to open the file
- Removing the commented out functions

- Assignment names are being hardcoded with “a”+id. What if the admin wanted a different name?
- Remove any long navigations such as a().b().c().....
- If we have setter methods, then we can call those, instead of directly going to database and back.

Diana’s Review

- Docstring need to explicitly outline function return values and parameters, and the purpose of the function
- The code needs to be cleaned up, there are many ‘testing’ print statements need to be removed, especially from user_assignments, assignments.py, problems.py
- Too many repeated lines of code in different files
    - Ex: creating a list box
        - Same functionality for assignments, users and problems but function is repeated because different database functions are called
    - Re-factor to have one function, with a call back or parameter to distinguish which database function to call
- Function variables need to have clearer and more useful names
- Exceptions and code errors, need to be handled with pop-ups to the users or by providing the user options that do not give errors

Bar’s Review

- Creating a new assignment method has a bug which needs to be fixed, can only add a new assignment with a subject already in the database, exception handling needs to be implemented (may crash-HIGH impact) and it needs to have a pop up to the user, or change the layout of the screen to having a dropdown instead of adding the subject manually
- When logging out or clicking “back”, screen labels/entries need to be destroyed/cleared out because when you re-click on the button for “Student Grades” or “View Assignment” the screen has the same display as before clicking “back” (Medium Impact)
- Line commenting needs to be concise, easy and direct lines don’t need to be commented

Code Review Sample Video

https://www.youtube.com/watch?v=NLa6-6tPuZ0