**Sprint 6 Task Breakdown:**

**User Story 6:** As Jennifer, a student, I want to complete problem sets that is released.

**Task 44:** Implement view problem set screen (3)

- Create interface that shows list of all problem sets
- User should be able to click on problem sets to show list of questions, reusing the view questions table from previous screens

**Task 45:** Implement answering questions screen (5, dependent on 42)

- Use attempt model to keep track of student answers
- User attempts one question at a time, navigating using next and previous buttons

**Task 73:** Implement functions to retrieve all problem sets from the database (2)

- Create a function to directly get all problem set data from the database
- Create a function in the API to convert this data to problem set objects

**User Story 7:** As Apple, an instructor, I would like to be able to search problems and problem sets by tags.

**Task 57:** Implement problem tags in database (3)

- Add a table to store the tags and the problems that they are associated with.
- Add function to DatabaseInserter to insert tag-problem relationship into the database.
- Add function to DatabaseStoreAPI to insert tags into the database for problems.

**Task 58:** Implement problem set tags in database (3)

- Add a table to store the tags and the problem sets that they are associated with.
- Add function to DatabaseInserter to insert tag-problem set relationship into the database.
- Add function to DatabaseStoreAPI to insert tags into the database for problem sets.

**Task 59:** Implement problem set tags in data model (1)

- Add variables to store tags.
- Add functions to add or remove tags.

**Task 60:** Implement problem tags in data model (1)

- Add variables to store tags.
- Add functions to add or remove tags.

**Task 61:** Implement GUI to search problems by tags (2)

- Add search box and controls to view problems screen
- Implement filter functionality through the view problems manager

**Task 62:** Implement GUI to search problem sets by tags (1)

- Add search box and controls to view problem sets screen
- Implement filter functionality through the view problem sets manager

**User Story 8:** As Leroy, an instructor, I would like to be able to add students from a formatted students file.

**Task 69:** Create parser to read delimiter-separated files into command arguments (2)

- Given delimiter and command, return list of string arrays with one interpreter command array per line in the text file

**Task 70:** Implement bulk importing students file to add students (1, depends 69)
- Add bulk import button and instructions to GUI to open file picker
- Implement button behaviour by calling parser from add student manager

**User Story 9:** As Leroy, an instructor, I would like to be able to add problems from a formatted problems file.

**Task 71:** Implement bulk importing problems file to add problems (1, depends 69)
- Add bulk import button and instructions to GUI to open file picker
- Implement button behaviour by calling parser from add problem manager

**User Story 10:** As Apple, an instructor, I would like to view the statistics for each problem set that I have created including completion rate, average mark, et cetera.

**Task 66:** Implement design for problem set statistics table (1)
- Create FXML file with a table to display student best scores and number of attempts
- Use labels to denote aggregate statistics that are computed outside of the table

**Task 67:** Implement controller for problem set statistics table (1)
- Implement functions to calculate aggregate statistics to set label values

**Task 68:** Implement manager for problem set statistics table (2)
- Get all attempts from API to create each table row
- Each row is a distinct student number - problem set combination
- Integrate with controller and layout file

**Task 74:** Implement functions to insert and retrieve previous attempts (2)
- Add functions to the API files to call the functions in the database files.
- Use the previousAttempt objects.

**Technical Debt:**

**Task 50:** Create JUnit Tests to test action, commands, databaseAPI, driver and the io using Mock Objects. (8?)

**Task 63:** Hardcode instructor login(1)
- Create an admin user for instructor at the login level(not in database)

**Task 64:** Redesign GUI for instructor dashboard (1)
- Add text and align buttons to make the instructor panel more visually appealing

**Task 65:** Redesign GUI for student dashboard (1)
- Add text and align buttons to make the student panel more visually appealing

**Task 72:** Refine GUI(4)
- Reorder the buttons in every GUI screen to allow tab and arrow keys to work properly
- Implement the enter key to work with any screen with submit button