Testing

Group members' names: Khang Dang, Javits Jacob, Brendan Parnell

NOTE: DataInitializer class is used to populate data for testing purposes before GUI is run and ProjectDemo is used to activate LoginWindow to begin testing.

Account class instances: username, password

- **Professor:**
  - alexander, 1234
  - brendan, 1234
- **Student:**
  - khangdang, 1234
  - kail, 1234
  - hamilton, 1234
- **TA:**
  - jackson, 1234
  - michael, 1234
  - javitz, 1234

Test plan is divided into 4 main topics that covers 4 actors/roles: professor, student, TA and Storage. The results of the test plan are retrieved and documented as follows:

Topic 1: Student

- ➔ Log in process:
  - Preconditions: username and password exist within text/binary file and users must know their info
  - Components: a text/binary file to contain a list of username and password separated by comma, Account class to represent each user, Storage class
  - Postconditions:
    - Grant access if the password and username are correct and a window will pop up based on the role of the user after OK button is clicked.
      - Successfully spawn StudentGUIWindow once correct student Account is entered and Log In button is clicked
    - Provide error messages if either of username or password is incorrect (exception handling)
      - Successfully report error message on LoginWindow.
    - Properly exits the system if Cancel button is clicked.
      - Successfully log out once Cancel button is clicked
- ➔ View grades:
  - Preconditions: students must be able to successfully log in to the system.
  - Components: Course class, Assignment Class
  - Postconditions:
    - Print the grades for each assignment of each course once the students hit the VIEW GRADES button
      - Grades of each assignment within a chosen course for a single student Account class is successfully shown in the StudentViewGradesGUI.
- ➔ View assignments: this function is incorporated within View Grades. Students can access this using the View Grades button within StudentOptionsGUIWindow. Assignments name, their descriptions and grades are posted within StudentViewGradesGUI.

Topic 2: T.A

- ➔ Log in process:
  - Preconditions: username and password exist within text/binary file and users must know their info
  - Components: a text/binary file to contain a list of username and password separated by comma, Account class to represent each user
  - Postconditions:
    - Grant access if the password and username are correct and a window will pop up based on the role of the user after OK button is clicked.
      - Successfully spawn TAGUIWindow once correct student Account is entered and Log In button is clicked
    - Provide error messages if either of username or password is incorrect (exception handling)
      - Successfully report error message on LoginWindow.
    - Properly exits the system if Cancel button is clicked.
      - Successfully log out once Cancel button is clicked
- ➔ Access student roster:
  - Preconditions: TAs must be able to successfully log in to the system and navigate to the correct course.
  - Components: Course class, Storage class
  - Postconditions:
    - Print the list of students' info for each course once the Professors choose course based on COURSE buttons and hit the View Roster button.
      - List of students' names of each course is successfully printed
      - Students who drop out of a course is no longer in the student roster list.
- ➔ Enter grades:
  - Preconditions: TAs must be able to successfully log in to the system and navigate to the correct course, access the correct assignment
  - Components: Grade, Course, Assignment class
  - Postconditions:
    - Print the new changes to the grades of a student in TAViewGradesGUI
      - Successfully entered grades of any student within and display them TAViewGradesGUI.
    - Send an error message if Professors try to edit grade of students or enter incorrect input for grades (letters and strings)(exception handling)
      - Non-integer input: java.lang.NumberFormatException is thrown
      - Invalid integer input greater than 100 or less than 0: AddGradeWindow closes without changing any data in ProfessorViewGradesGUI.
- ➔ Edit grade: this option is unavailable for TAs
- ➔ View grades:
  - Preconditions: TAs must be able to successfully log in to the system and navigate to the correct course, access the correct assignment
  - Components: Grade, Course, Assignment class
  - Postconditions:

- Print the grades for each assignment of each course once the hit the View Grade button and choose the assignment
  - Successfully display grades of all student in the course for said assignment.
  - Successfully display changes to grades of any student within said assignment.
  - Successfully maintain data after invalid input from professor when editing or adding grade for a student.

Topic 3: Professor

- ➔ Log in process:
  - ◆ Preconditions: username and password exist within text/binary file and users must know their info
  - ◆ Components: a text/binary file to contain a list of username and password separated by comma, Account class to represent each user
  - ◆ Postconditions:
    - ● Grant access if the password and username are correct and a window will pop up based on the role of the user after OK button is clicked.
      - Successfully spawn ProfessorGUIWindow once correct student Account is entered and Log In button is clicked
    - ● Provide error messages if either of username or password is incorrect (exception handling)
      - Successfully report error message on LoginWindow.
    - ● Properly exits the system if Cancel button is clicked.
      - Successfully log out once Cancel button is clicked
- ➔ Access student roster:
  - ◆ Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course.
  - ◆ Components: Course class, Storage class
  - ◆ Postconditions:
    - ● Print the list of students' info for each course once the Professors choose course based on COURSE buttons and hit the View Roster button.
      - List of students' names of each course is successfully printed
      - Students who drop out of a course is no longer in the student roster list.
- ➔ Access TA roster:
  - ◆ Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course.
  - ◆ Components: Course class, Storage class
  - ◆ Postconditions:
    - ● Successfully print the list of students' info for each course once the Professors choose course based on COURSE buttons and hit the ROSTER button.
      - List of students' names of each course is successfully printed
- ➔ Enter grades:
  - ◆ Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course, access the correct assignment
  - ◆ Components: Grade, Course, Assignment class
  - ◆ Postconditions:
    - ● Print the new changes to the grades of a student in ProfessorViewGradesGUI
    - Successfully entered grades of any student within and display them ProfessorViewGradesGUI.

        - Send an error message if Professors try to edit grade of students or enter incorrect input for grades (letters and strings)(exception handling)
            - Non-integer input: java.lang.NumberFormatException is thrown
            - Invalid integer input greater than 100 or less than 0: AddGradeWindow closes without changing any data in ProfessorViewGradesGUI.
            - Edit grade instead of adding: AddGradeWindow closes without changing any data in ProfessorViewGradesGUI
- ➔ View grades:
    - Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course, access the correct assignment
    - Components: Course, Assignment class, Storage class
    - Postconditions:
        - Print the grades for each assignment of each course once the hit the View Grade button and choose the assignment
            - Successfully display grades of all student in the course for said assignment.
            - Successfully display changes to grades of any student within said assignment.
            - Successfully maintain data after invalid input from professor when editing or adding grade for a student.
- ➔ Edit grades:
    - Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course, access the correct assignment
    - Components: Course, Assignment class
    - Postconditions:
        - Professors can edit already filled in Grade fields of each student within a list
            - Successfully changed grades of any student within ProfessorViewGradesGUI.
        - Send an error message if Professors try to add grade to students or enter incorrect input for grades (letters and strings)(exception handling)
            - Non-integer input: java.lang.NumberFormatException is thrown
            - Invalid integer input greater than 100 or less than 0: EditGradeWindow closes without changing any data in ProfessorViewGradesGUI.
            - Add grade instead of editing: EditGradeWindow closes without changing any data in ProfessorViewGradesGUI.
- ➔ Adding assignment:
    - Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course
    - Components: Assignment, Course class
    - Postconditions:
        - Professors can create a course using a String prompt for name and another String prompt for description.
            - New assignment name and its description is successfully displayed in ProfessorViewAssignmentsGUI.

- Every student within the course can see their new assignment via StudentViewGradesGUI.

- Error when professor enter an already existing assignment name
  - Successfully display error message on AddAssignmentWindow that an assignment with the same name already exists.

➔ Remove assignment:

- Preconditions: Professors must be able to successfully log in to the system and navigate to the correct course, and the assignment must not be graded and must exist in the course
- Components: Assignment, Course class
- Postconditions:
  - Professors delete an ungraded assignment.
    - Successfully display the new data without the old assignment name and description.
    - java.util.ConcurrentModificationException maybe thrown onto terminal window due to multiple changes to data at the same time. (repeated clicks are required to bypass this Exception)
  - Professor try to delete an assignment not in list (assignment name not in the list) or a grade assignment
    - Successfully display error message on the RemoveAssignmentWindow and terminal.

Topic 4: Storage

- ➔ Students drop out of a course:
  - Preconditions: Student click on Drop Course button.
  - Components: Course class, Account class Storage
  - Postconditions:
    - Students' grades no longer in each assignment of a course
      - Successfully remove student username and grade from ProfessorViewGradesGUI of any assignment of said course.
    - Students' names no longer on the roster list of a course
      - Successfully remove student's name in the roster list of said course
    - Error if the course cannot be removed (exception handling).
      - java.util.ConcurrentModificationException is thrown if there are more than one change is implemented at the same time. (repeated click is required to bypass this exception)
- ➔ Remove finished course:
  - Preconditions: Professor deems a course to be complete at any point, desire for the course to be deleted and click on the Remove Course button in ProfessorOptionsGUIWindow
  - Components: Course class, Assignment class, Account class, Storage class
  - Postconditions:
    - The course is deleted from the database.
      - The course is successfully removed from all three account access of professor, TA and student.
- ➔ Persistently store assignments for each course:
  - Preconditions: A new change is made to the data (enter or edit grades and assignments)
  - Components: Course class, Assignment classes and Storage class
  - Postconditions:
    - New changes is synchronized and stored on the database
      - The system successfully rewrites changed data into binary data files
      - "Writing Account List from Binary File" and "Writing Course List from Binary File" are shown on terminal if changes are successfully written

Conclusion

Overall the main use cases for each actors are covered and successfully implemented even though there are some problems such as the java.util.ConcurrentModificationException. For such an issue, the group may need to research more on the cause and nature of the exception further and try to anticipate the error with the try and catch statements in its codes. Other problems may arise as well but so far this is what the group has found.