**TEST SCENARIOS**

For

**UNIVERSITY DEPARTMENT INFORMATION SYSTEM**

**(UDIS)**

**SUBMITTED BY**

Nikhil Tudaha (19CS10045)

Seemant G. Achari (19CS10055)

Rajas Bhatt (19CS30037)

**MENTOR**

Uppada Vishnu

**DATE**

27 March 2021

**VERSION**

1.0

**BRIEF DESCRIPTION**

We have the following scenarios for testing, as has been mentioned in the test plan document.

* 1. Signing in to the system
  2. Adding a new student to the system
  3. Searching for a student, using both roll number and name queries
  4. Viewing student performance and grade details
  5. Registering courses for a particular student
  6. Entering grades for a student
  7. Adding new items to the department inventory and withdrawing funds from the Department Account
  8. Searching inventory items using type/name
  9. Viewing and adding new courses
  10. Viewing and adding new publications
  11. Viewing and adding new projects and changing their status
  12. Adding funds to the Department Account

**DESCRIPTION**

In the following section, we provide the description for the unit tests of various classes.

1. **Entry Screen/ Login Screen (Filename ES.py)**

In this, we test whether the login functionality works correctly. This is tested on the following credentials.

* 1. Login ID correct and Password incorrect (Exception Path)
  2. Login ID incorrect and Password correct (Exception Path)
  3. Login ID incorrect and Password incorrect (Exception Path)
  4. Any entry left blank, either Login ID or Password (Exception Path)
  5. Login ID correct and Password correct (Happy Path)

Exception paths must show alerts on the GUI interface.

1. **Home (Filename Home.py)**

Check sign out button and try to login again. The user must have to login again and cannot sign in without entering credentials.

1. **Student Main Screen (Filename StudentMain.py)**

Check all the buttons, and whether they lead to their respective screens or not.

1. **View Student Screen (Filename StudentView.py)**

This is an important screen, which has a lot of functionalities. The different scenarios include

* 1. **Empty roll no. empty name**

In the case of nothing filled, all the students in the database **must** be shown. This can be verified by checking the number of students in total.

* 1. **Empty roll no. filled name**
  2. **Filled roll no. and filled name**

In each of the cases above, all the students, which have names and roll numbers similar to the search queries (i.e. substrings), must be displayed

* 1. **Filled roll no. and filled name**

In this scenario, the screen will display all the students whose roll numbers match the roll number mentioned **or** whose names match the names mentioned.

* 1. **Testing the Binding Action**

Clicking on the data entry of each student must open a new popup, whose title must match the roll number of the student. The following things must be checked and none of them should be vacant.

* + 1. Correct name, correct roll number, correct address, correct year of joining and the CGPA (these details can be manually checked).
    2. Performance Section

1. Backlogged subjects should be shown in a different section.
2. SGPA of each semester should be computed accurately up to two decimal places.
3. Subjects which have been backlogged once but have been cleared should display an asterisk (\*).
4. PDF generation functionality – The user should be able to save a PDF which has correct data (i.e. data matches with the data shown on the screen). The PDF should also be saved with the correct name.
5. **Register Courses Screen (Filename StudentCourses.py)**

We are already testing the search student functionality in the previous section. Since both this class and the previous class have been inherited from the same abstract base class, i.e. StudentSearch.py, we do not need to test the same functionality again. We need testing for the following:

* 1. **Testing the Binding Action**
     1. **Cleared courses not to be shown.**
     2. **Backlog courses must be shown. This is because backlog courses can be registered again.**
     3. **Cleared Backlog courses should not be shown.**
     4. **A maximum of 12 courses can be registered in a given semester, if a student has never had a backlog**
     5. **A maximum of eight courses can be registered in a given semester if a student has ever had a backlog**
     6. **The courses, which have been just registered and not graded, must have R grade against them in the database**.

1. **Entering Student Grades Screen (Filename StudentGrades.py)**

We need testing for the following:

* 1. **Testing the Binding Action**
     1. **Only the ongoing courses are to be fetched.**
     2. The subjects can be graded only after filling all the grades, otherwise, alerts must be shown on the screen.
     3. The grades for the cleared subjects should be updated and visible in the performance section.

1. **Adding New Student (Filename StudentNew.py)**

In this screen, we create and add a new student.

* 1. Roll number format checking
     1. Correct format entered (<Last 2 year digits><2 department character identifiers><course identifier digit/character><4 serial number digits> ) (Happy path)
     2. Any other format entered (Exception path), alerts must be shown to enter the correct format roll number.
  2. Year of admission checking
     1. Year entered in the correct format, i.e. a 4 digit number (Happy Path)
     2. Year entered in an incorrect format, i.e. some alphabets or special characters inserted or the year entered is not a 4 digit number (Exception Path)
  3. Duplicate Roll number added (Exception Path)
  4. Some fields are left blank (Exception Path), respective fields shown in alerts.
  5. All entries have been entered correctly and the add button clicked (Happy Path), and then check if the said student can be accessed from the database (test case passed).
  6. Course identifier matches with the course, otherwise alert shown.

1. **Department Main Screen (Filename DepartmentMain.py)**

Check all the buttons, and whether they lead to their respective screens or not.

1. **Department Inventory (Filename DepartmentInventory.py)**

This is the inventory section of the department where all the items and their locations are mentioned.

* 1. **Searching for an item**
     1. **Blank name type all**

All items in the inventory are shown (Happy path), this can be verified by counting the entries in the database.

* + 1. **Name specific type all**

Only the items whose names match with the entered name across all types should be shown. This is also a happy path.

* + 1. **Name blank type specific**

Only the items that match the specific type of the five types mentioned in the drop down menu should be shown.

* + 1. **Name specific type specific**

Items that match with a particular name **and** the type selected should be shown.

All four of these paths are happy paths.

* 1. **Adding an item to the inventory**
     1. **Transaction Failure (Exception Path)**
* Insufficient funds in the Department Account, i.e. this transaction will lead to negative balance.
* Quantity of items is not positive or not a number
* Price is not positive or not a number
* One of the fields is blank
  + 1. **Transaction Succeeds (Happy Path)**
* Price and Quantity positive
* All the blanks filled.
* If location, name and type match with a particular item from the inventory exactly, increase quantity.
* If either one of location, name and type do not match, add a new entry into the inventory.

1. **Department Account Screen (Filename DepartmentAccount.py)**

**Adding money to the account**

* 1. **Money Successfully Inserted into the account (Happy Path)**
* All transaction fields filled using valid data.
* Amount is positive and a number.
* Date format correct (DD/MM/YYYY in this case).
* Inserted money should show in the account passbook with the date, donor details and the current balance.
  1. **Money not inserted (Exception path)**
* Donation amount is not positive or not a number (alert shown)
* Any one field left blank
* Date format incorrect (alert displaying the correct format of the date shown to the user)
* Invalid Date (like 30 Feb 2021)

1. **Department Academics Section Screen (Filename DepartmentAcademics.py)**

Check all the buttons, and whether they lead to their respective screens or not.

1. **Department Courses (Filename DepartmentCourses.py)**

In this section, we check the functionality of adding new courses to the department.

* 1. Searching for Courses
     1. Blank Input (Happy Path) – All courses of the department must be shown
     2. Course name entered (Happy Path) – All courses of the department having names which are similar to the entered course name must be shown.
  2. Adding new Courses
     1. **Course Successfully Added (Happy Path)**
        + Course code not already present.
        + Credits for the course should be a number and must not exceed a reasonable value (2-5) in this case.
        + Newly added course available for registration and visible in when searching.
     2. **Course not Added (Exception Path)**
        + Course with the same code already present.
        + Credits entered is not a number
        + Credits entered is a negative or a large number exceeding five.
        + Either one of name, code or credits column left blank.

1. **Department Publication Section Screen (Filename DepartmentPublications.py)**

In this section, we test the various publications in the department. There is only one scenario, **adding a new publication**.

* + 1. **Publication Successfully added (Happy path)**
       - No field left blank
       - The set of fields entered do not match with any previous data entry.
       - Publication visible in the Publication Section screen.
    2. **Publication not added (Exception path)**
       - One or more of the fields left blank.
       - The set of fields already match with a previous data entry.

1. **Department Projects (Filename DepartmentProjects.py)**
   1. **Adding a new Project**
      1. **Project Successfully Added (Happy Path)**
         * All the fields must be filled.
         * Funds field is a valid integer and the requisite funds are already present in the bank account
         * No project with the same name exists

0

* + 1. **Project Not Added (Exception Path)**
       - One or more of the fields vacant.
       - Funds is not a integer
       - Funds is negative
       - Sufficient funds not present in the bank account
       - Project name already exists before
  1. **Testing the Binding Action to change status of the project**
     + - Status can be changed if the current status is ‘Ongoing’ to ‘Completed’
       - Status cannot be changed if the current status is ‘Completed’