1) **Harikrishnan Sukumaran Nair**
    a) **Created the initial framework for the project and integrated the application.**
    b) **Functionalities Implemented:**

i) Login Screen(fxml and controller):

1) **Sign In:** Logs in user to the system based on Login Name and Password
   Validations: 1) Check if the Login Name and Password is present in the database.
   If present, then guide user to Admin or Student Dashboard Screen based on the role of the user.
2) **Signup:** Guides user to a new signup page

ii) Sign Up Screen(fxml and controller):

1) If logged in as default user, then add a new student to the system. If logged in as an admin then add new admin/student user to the system
2) Sign Up button would pop up alert in case of following scenarios
    a) Login Name already exists in the system
    b) If the value of Password and Re-enter Password is same
    c) Role selected is not Student/Admin
    d) If all the values are empty

If all the above validations are passed, then a new user is added to the system.

3) Back Button: On click event of this user is guided back to login screen if the user is student. If the user is admin then the user is guided back to admin dashboard.

iii) Admin Dashboard Screen(fxml and controller):

1) Provides the user following options
    a) Add a new student/admin user to the system.
    b) Upload Questions file to the system.
    c) View the student performance.

iv) **Upload File Screen(fxml and controller)**:

1) Allows user to select a file from the local machine and upload the questions to the database
2) Following validations are done
    1) Appropriate file path is selected. IF not alert the user
    2) Upon successful upload alert the user that file has been successfully uploaded
    3) If the file upload has errors, then convey the error message to user.

2. Naveen Nehru

File DB Utility: Implemented the methods for

1. CSV file reading
2. Validation (error handling scenarios) of file format and data
3. Loading the file data into Question table.
4. Added a couple of methods to load data to the ‘test_results’ and ‘users’ table.

Tested the below scenarios in the CSV files and captured some of the following exceptions.

1. Introduce 'correct / incorrect / true / false' keywords in question description field to check if it loads to DB
2. Introduce leading and trailing spaces, commas and double quotes in question description field to check if it loads to DB
3. Introduce wrong question type, say EA, to see if it gets captured in the error log
4. Introduce wrong difficulty levels, say L, to see if it gets captured in the error log
5. Introduce wrong answer choices, say "fine" other than "correct / incorrect" in MC to see if it gets captured in the error log
6. Introduce multiple correct answers in MC to see if it gets captured in the error log
7. Introduce all answer choices as "incorrect" for MA questions
8. Introduce the answer response(correct/incorrect) in double quotes in file and validate
9. Invalid no. of input fields for question types, say presence of more than 4 field values for question type=FIB

3. Vinaya Sai Davuluri

1.Create the Questions classes

1. Questions (Super class)
2. MultiChoiceQuestion (Subclass of Questions)
3. FillInTheBlanks (Subclass of Questions)
4. TrueOrFalseQuestion (Subclass of Questions)

2. Student Dashboard

The student dashboard has 2 parts:

1. **Take the quiz test**
    a. Where he needs to select the level of difficulty and no. of questions
    b. If there are no questions in the database, it says the same.
    c. If a difficulty level is not having minimum of 3 questions that option will not be shown in drop down.
    d. Minimum of 3 questions are required to start a test.
2. **View the results**
    a. This has recent 7 tests that the student has taken.
    b. If there are no tests. It displays a message saying the same.
3. **View Aggregate Results**
    1. Gives the student details about all the tests he has taken similar to an Admin but it only displays for his tests.

Taking Test

- Once after selecting the difficulty level and number of questions, instructions page comes up.
- Student can go to back page to reselect or start test by pressing that button.

- The first question will never be shown a back button.
- The next button appears only if an answer is given.
- He can skip the question if he wants not to answer it.
- A student can skip only 20% of the total questions. After that he cannot skip.
- He can go back to any question and change his answer.
- Once all questions are done a submit page comes and he can submit the test and get the result or go back to change answers.
- On Submitting the test the result of the quiz is displayed with a graph. And he can go back to dashboard or export the result as PDF or logout of the account.

Student Result Dashboard

- The student can view the result of the last 7 tests he has taken.
- They are sorted to the most recent to least recent.
- By clicking on the test, he is directed to the summary of the test with a graph giving the details of the test.

4. Jose Alberto Rodriguez Garcia

Data Selection Utility.

Implemented methods to read data from the database for the use of the application.

1. Reading the questions stored in the database, then creates question objects according to its question type to start the Quiz.
2. Reading Clobs of SQL data.
3. Counting type of questions stored in the database.
4. Counting the number of questions available in the database depending on their level of difficulty.
5. Counting the number of questions that were answered correctly, and generate all the data necessary for the student result class to be stored in the database.
6. Getting student results stored in the database.
7. Getting users stored in the database.

5. Kuhu Bhadani : -

a. Created students statistics UI for Admin and Student

b. Created one CSS that is implemented in whole project.

c. Created "Export To PDF" functionality for whole project.

d. Created "BarChart" for statistics UI.

e. Developed functionality for representing statistics UI.

f. Functionalities Implemented:

i) Student Statistics Screen(StudentStats.fxml and StudentStatsController.java):

1. **"Select Student" Dropdown:** If an **Admin** is logged in , he selects whether he wants to view report for all students or only one student. If a **Student** is logged in, he can select only his name from the dropdown.
2. **"Select Report" Dropdown:** This gets enabled only once user selects "Select Student" dropdown. The user selects what report he wants to view.
3. **"Select Period" Drop Down:** This gets enabled only once user selects "Select Report" dropdown. The user selects period for which he want to view the report.
4. **"View Stats" button:** This gets enabled only once user selects "Select Period" dropdown. If no test has been taken so far, user will navigate to NoData.fxml. Otherwise, the user will navigate to "NoOfTestTaken.fxml.fxml".
5. **"Back" button**: Takes the user back to admin dashboard.
6. **"Logout" button:** Takes the user to login page.

ii) **NoOfTestTaken.fxml and NoOfTestTakenController.java**

1. **BarChart:** Barchart is drawn according to the data.
2. **Export To PDF**: Exports the barchart to the PDF.
3. **"Back" button**: Takes the user back to admin dashboard.
4. **"Logout" button:** Takes the user to login page.

iii) NoData.fxml and NoData.java

1. This fxml is displayed when there is no test taken by any student.
2. **"Back" button**: Takes the user back to admin dashboard.
3. **"Logout" button:** Takes the user to login page.