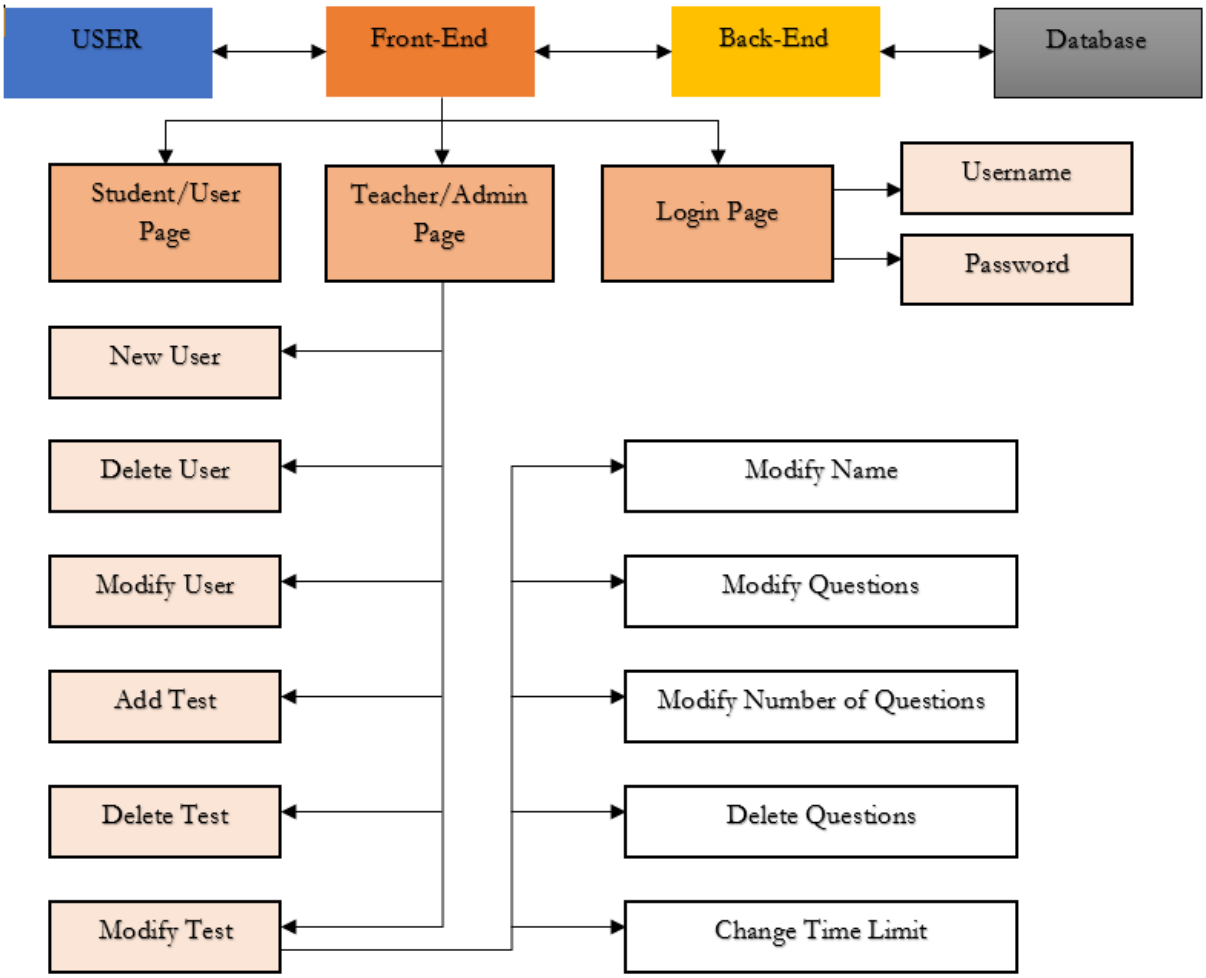
On the Exam Line

Regression Testing and Incremental Testing

Gunjan Gauri, Arka Acharya, Joyce Kao, Shivaganesh Balasundaram

1. Classification of Components

1.1 Defining all Components



1. **User:** The person who will be using our application.

2. **Front-End:** Our front end is a simple web application that is written in HTML, PHP, JavaScript, and CSS. It is dependent on the backend server which is simulated by using XAMPP. It takes text, radio buttons, and pushbuttons as inputs from the user, and makes the appropriate calls to the database for data retrieval. After processing the information, it's output

is updated on the same page, or the user is redirected to the appropriate page. As it is the front end, no other component depends on it. Front end itself is broken down into several components which have their own dependencies.

a. Login Page:
    i. Username
    ii. Password
b. Teacher/Administrator Page:
    i. New User
    ii. Modify User
    iii. Delete User
    iv. Add Test
    v. Delete Test
    vi. Modify Test:
        1. Modify Name
        2. Modify Questions
        3. Modify Number of Questions
        4. Delete Questions
        5. Change Time Limit
c. Student Page

3. **Back End:** Our backend is responsible for the apache interpreter. As it is provided by XAMPP we didn't want to change it very much. We used it to implement features of PHP which could be interpreted only by an apache interpreter. It is mainly used to send values of variables from one page to the other. It plays a vital role in the application as the front end heavily depends on it.

4. **Database:** The PHP files communicate to the database through the XAMPP server. This connection is established through XAMPP's secure network using specific statements. The MySQL database effectively stores all the data in the tables.

1.2 Form of Incremental Testing Used

We used the **bottom-up** incremental testing strategy to test On-The-Exam-Line. Using drivers, we began testing the lower-level modules, eventually working our way up to the main high-level modules. We created test cases for each individual lower-level component and then tested the components that relied on those on the lower level, moving up the hierarchy. For a short example, we tested the components for modify name, modify questions, modify number of questions, delete questions, and change time limit. When those were working, we then moved up a level and tested the modify test component. It was easier for us to write test cases for individual modules that were already existing, so bottom-up testing was the best strategy for us.

2. Incremental and Regression Testing

A) MODULE | COMPONENT A - FRONT END

A1) LOGIN MODULE

→ **Incremental Testing**

Defect No.	Description	Severity	How to Correct
1	The password was showing up instead of being hidden.	2	The input type was changed from text to password.
2	Minimizing the window resulted in misalignment of the textboxes in the log in page.	3	The alignment was adjusted in the HTML file using a wrapper and some CSS functions.

→ **Regression Testing**

Defect No.	Description	Severity	How to Correct
1	Fixing the misalignment issue led to the alignment not auto adjusting when the page was manually resized using the drag option.	3	The alignment values were changed to percentages rather than specific values.

A2) ADD TEST MODULE

→ **Incremental Testing**

Defect No.	Description	Severity	How to Correct
1	When the text box next to the chapter name slot is adjusted, the whole page misaligns.	3	The alignment values in the HTML file had to be adjusted.

→ **Regression Testing**

Defect No.	Description	Severity	How to Correct
1	Adjusting the text box now lead to the box overlapping other content in the page.	3	This was fixed by making the other content wrap around the adjustment made.

B) MODULE |COMPONENT C - DATABASE

→ **Incremental Testing**

Defect No.	Description	Severity	How to Correct
1	The tables didn't have enough columns at strategic places to accommodate for some information and foreign keys.	1	The required columns were added to store the information and create foreign keys.

→ **Regression Testing**

Defect No.	Description	Severity	How to Correct
1	The queries stopped working after the changes in the database were made and more columns were included which let to warning messages and incorrect information being displayed in the website.	1	The old query statements were using old column names which when changed started to work again.

3. Updated Product Backlog

Functional Requirements: Administrator

Backlog Id	Functional Requirement	Hours	Status	UPDATED STATUS
1.	Log in to the administrator account.	2	Sprint 1	Completed in Sprint 1
2.	Add new users.	4	Sprint 1	Completed in Sprint 1
3.	Modify existing users.	3	Sprint 1	Completed in Sprint 1
4.	Delete existing users.	2	Sprint 1	Completed in Sprint 1
5.	Add new exam: Choose the number of multiple choice questions.	3	Sprint 1	Completed in Sprint 1
6.	Add new exam: Choose the number of free response questions (Essay questions).	3	Sprint 1	Completed in Sprint 1
7.	Add new exam: Set a custom time limit for each exam.	3	Sprint 1	Completed in Sprint 1
8.	Add new exam: Enter multiple exams for each chapter.	3	Sprint 1	Completed in Sprint 1

9.	Clone an exam: Copy all the questions into a new exam with a different name.	3	Sprint 2	Planned for Sprint 2
10.	Modify existing exam: Edit the name of the exam.	3	Sprint 1	Completed in Sprint 1
11.	Modify existing exam: Edit existing questions.	3	Sprint 1	Completed in Sprint 1
12.	Modify existing exam: Add new questions.	3	Sprint 1	Completed in Sprint 1
13.	Modify existing exam: Delete existing questions.	3	Sprint 1	Completed in Sprint 1
14.	Modify existing exam: Change time limit.	3	Sprint 1	Completed in Sprint 1
15.	Change the name of an existing chapter.	3	Sprint 2	Planned for Sprint 2
16.	Delete an existing exam.	2	Sprint 1	Completed in Sprint 1
17.	Grade free response questions (essay questions) entered by users.	3	Sprint 2	Planned for Sprint 2
18.	Decide which chapter's exam the user should be able to take.	3	Sprint 1	Completed in Sprint 1
19.	Review the test scores of all the test takers.	3	Sprint 2	Planned for Sprint 2
20.	Review the answers of the users against the correct answers of that exam.	3	Sprint 2	Planned for Sprint 2
21.	Log out of the account.	2	Sprint 1	Completed in Sprint 1

Functional Requirements: User

Backlog Id	Functional Requirement	Hours	Status	UPDATED STATUS
1.	Log in to their user account.	2	Sprint 1	Completed in Sprint 1
2.	Take a test.	3	Sprint 2	Planned for Sprint 2
3.	Be assigned an exam randomly from the selected chapter to users (if chapter has multiple variations).	4	Sprint 2	Planned for Sprint 2
4.	See time remaining during a test.	3	Sprint 2	Planned for Sprint 2
5.	See multiple choice grade as soon as test is over.	3	Sprint 2	Planned for Sprint 2
6.	See overall grade after free response questions (essay questions) are graded.	3	Sprint 2	Planned for Sprint 2

7.	Log out of their user account.	2	Sprint 1	Completed in Sprint 1

Functional Requirements: Database

Backlog Id	Functional Requirement	Hours	Status	UPDATED STATUS
1.	User information.	4	Sprint 1	Completed in Sprint 1
2.	List of chapters.	4	Sprint 1	Completed in Sprint 1
3.	Test information.	4	Sprint 1	Completed in Sprint 1
4.	Test questions.	3	Sprint 1	Completed in Sprint 1
5.	Answers to multiple choice questions.	3	Sprint 1	Completed in Sprint 1
6.	Answers entered by the user.	3	Sprint 2	Planned for Sprint 2
7.	User's grades.	4	Sprint 1	Completed in Sprint 1
8.	Chapters the users are allowed to take a test for.	4	Sprint 1	Completed in Sprint 1

Non -Functional Requirements

Backlog Id	Functional Requirement	Hours	Status	UPDATED STATUS
1.	The application should be supported by all web browsers.	4	Sprint 1	Completed in Sprint 1
2.	The application should be attractive and easy to use.	4	Sprint 1	Completed in Sprint 1
3.	As a developer, I would like to store information in a MySQL database.	4	Sprint 1	Completed in Sprint 1
4.	Database should handle requests.	4	Sprint 1	Completed in Sprint 1
5.	Separate administrator privileges.	3	Sprint 1	Completed in Sprint 1