

# **Exami-Nation**

## **Design Document**

**Team 13: David Lee, Arka Acharya, Austin Miller, Yixiong He**

## Index

- **Purpose**
  - **Functional Requirements**
  - **Non-Functional Requirements**
- **Design Outline**
  - **High-Level Overview**
  - **Flow of Events**
  - **State Diagram**
- **Design Issues**
  - **Functional Issues**
  - **Non-Functional Issues**
- **Design Details**
  - **Class Diagram**
  - **Description of Class Diagram**
  - **Navigation Flow Map**
  - **UI Mockup**

## Purpose

Many activities today have progressed online including education. Some other developers have created platforms where users can accomplish a teacher-student relationship digitally. However, they fail to take in students' perspectives and show valuable statistics of their progress to the teachers. Exami-Nation is a web-application where not only can teachers create better and faster exams, but also provides a platform for students to build a learning profile while gaining achievements for their hard work.

## Functional Requirements

Exami-nation will have the following functional requirements:

1. For the actions of administrative users such as teachers,
  - 1a. Administrative users will be able to log into their own account with administrative privileges.
  - 1b. Administrative users will be able to choose, modify, and add users to take their tests.
  - 1c. Administrative users will be able to create exams in which they can
    - Choose the number of multiple choice questions
    - Choose the number of free response questions (Essay questions).
    - Set a custom time limit for each exam.
    - Enter multiple exams for each chapter.
    - Change the name of the exam (Cloning)
    -
  - 1d. Administrative users will be able to modify their exams in the following ways
    - Edit the name and questions in the exam
    - Edit the time limit for the exam
  - 1e. Administrative users will be able to decide which exams are open to which users at a specific time as well as review all answers in submissions.
2. For the actions of regular users such as students,

2a. Users will be able to log into their accounts in which they will be able to see which tests they have pending.

2b. Users will be able to see how much time they have left in their exams as they are taking them.

2c. Users will be able to receive their exam grades for multiple choice as soon as they finish, as well as their exam grades for essays questions as soon as those are graded by the administrative user giving the exam.

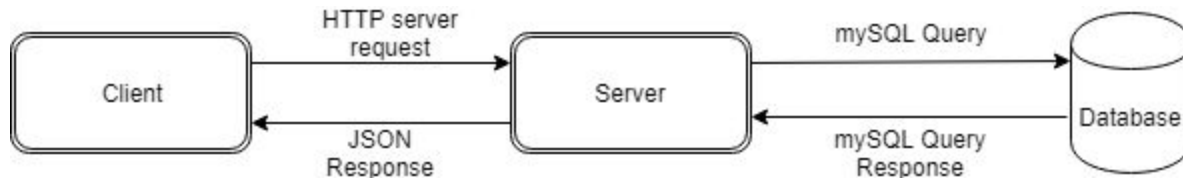
## **Nonfunctional Requirements**

1. The web application should be supported by all web browsers.
2. The server should be able to save all user information into a local database.
3. The server should be able to save all course information and tests into a local database.
4. The information should be saved securely in the database, where grades for exams and its answer key or question pool will not be leaked.
5. The server should be able to handle fast and concurrent requests.

## Design Outline

1. Client
  - a. HTTP requests will be sent to the server by the client.
  - b. Client will receive JSON format responses from the server.
  - c. The data will be interpreted and rendered for use in Javascript.
2. Server
  - a. Server will receive and process all requests from the Client.
  - b. Server will provide appropriate responses for each request.
3. Database
  - a. Database will store all information used in Examination, such as user information, course information, grades, and etc.

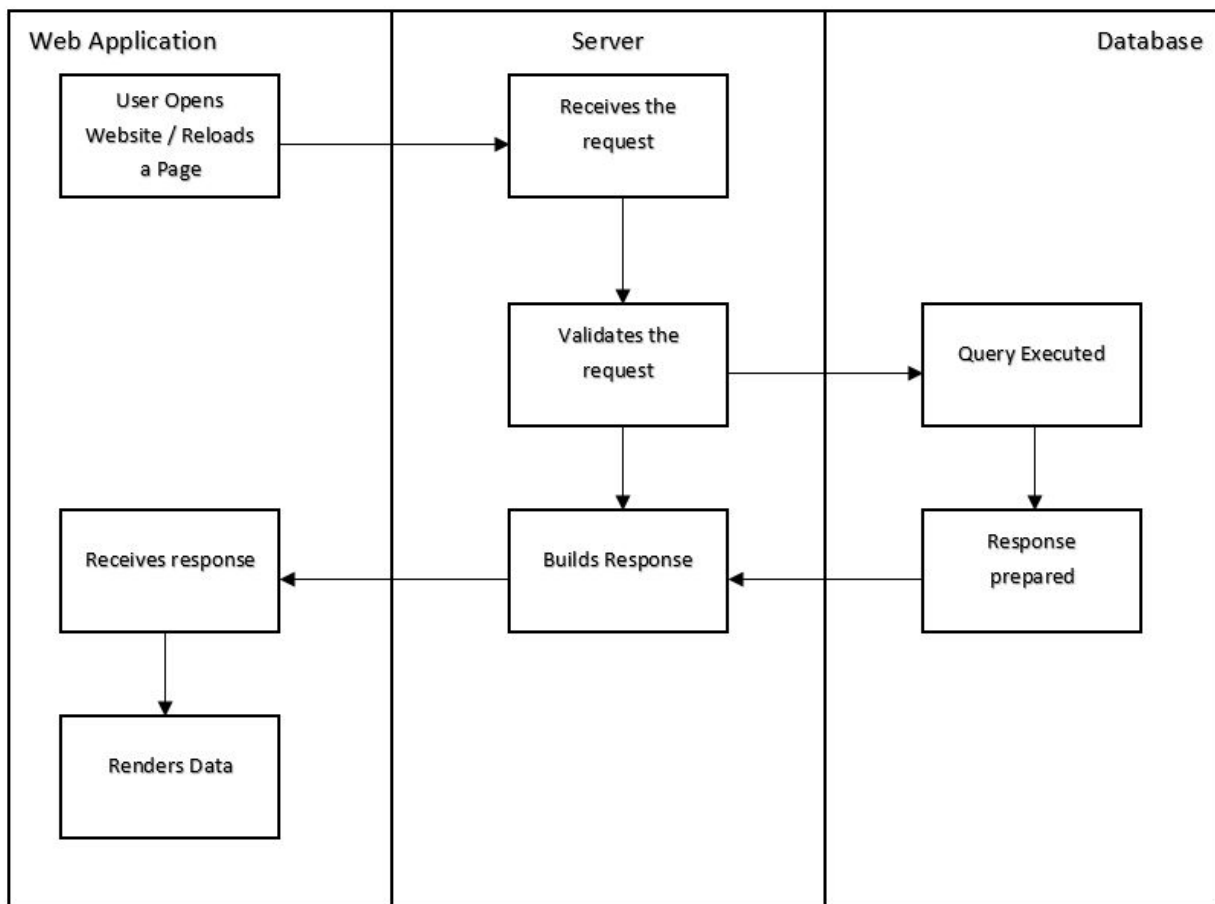
## High-Level Overview



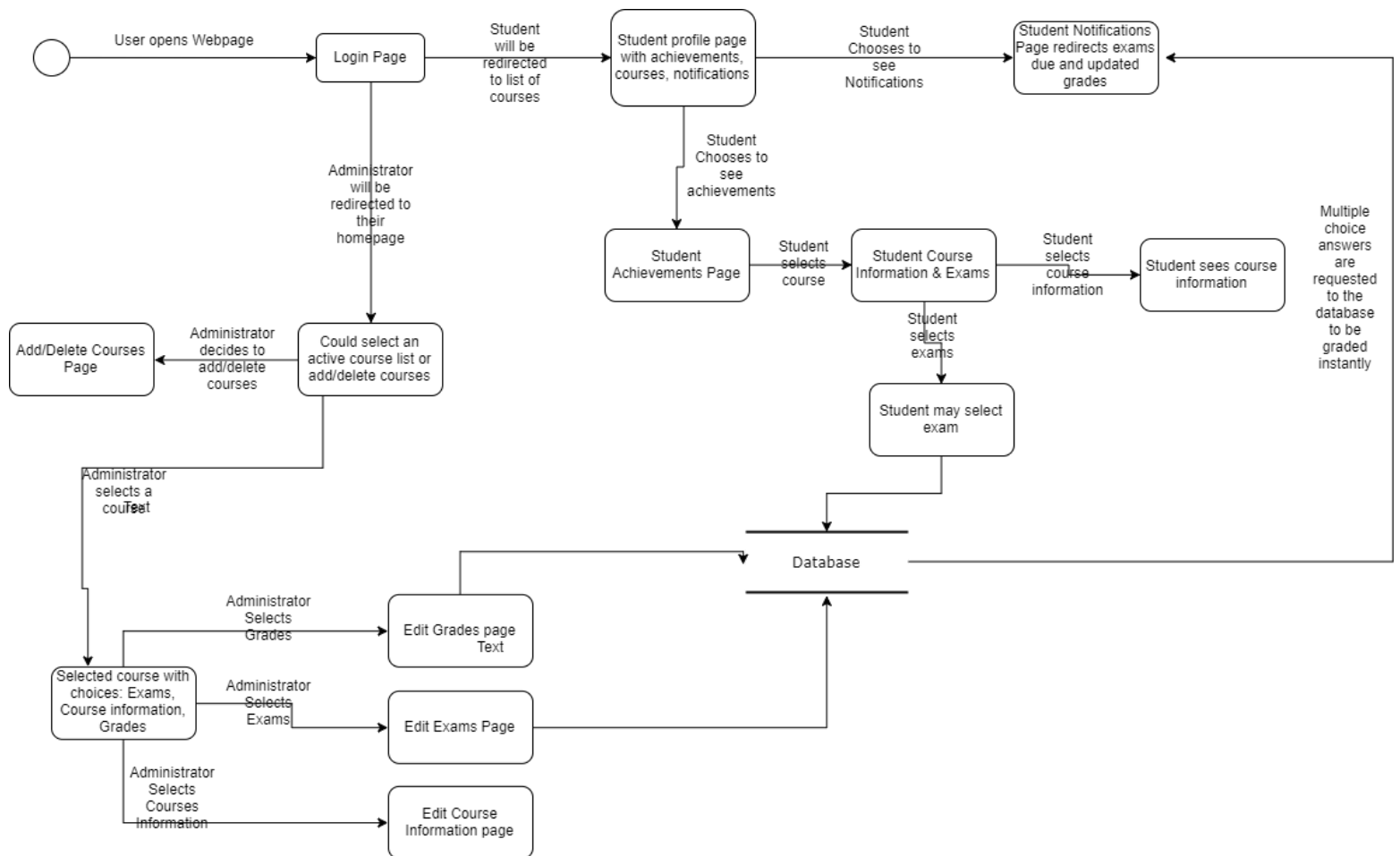
We will be using the client-server model for this project. The client will send requests which the server will get a response in JSON format. The model above shows the high-level overview of the system.

## Flow of Events

The typical flow of events begins after the user logs into the application. When the web application requires data from the server to complete a page or updates something in the database, there is an AJAX request that is sent to the server. From there, the Server validates the fact that the user may access or update the requested information. If the request is valid, the server will query the database. There, depending on whether there was a get request or post request, the response prepared will be different. The Server will parse the data that is received into a JSON format and send that back to the android application. There, the App will parse the JSON token to render the pages on the application.



## State Diagram



The state diagram illustrates how users will login, and what kind of choices they will be able to navigate through Exami-Nation.

## Design Issues

### Functional Issues

1. How will we ensure the maximum amount of fairness regarding test taker?

Option 1: Monitor when the user is using other services, or if the web-app is in the background.

Option 2: Don't monitor anything, and hope the user is academically honest.

Decision: We are going to attempt to make sure that no other tabs or windows are open in the test taker's browser. If such an event does take place, we will immediately terminate the exam and give whatever has been completed so far for grading.

2. How will we ensure the maximum amount of fairness regarding test creator/ graders (teachers)?

Option 1: Automate grading multiple-choice questions, but not the essay questions.

Option 2: Automate grading both multiple-choice questions and essay questions.

Decision: We are going to automate grading only for the multiple choice questions, but not the essay questions. We will have an answer key against which the teacher will grade the essay question. If time allows we will try to automate the grading of the essay questions and then take the average of the automated score and the teacher score.

3. Will all users need to have an account made that is tied to their identity, or can anyone come on to the application and choose to take a test as practice or to be graded?

Option 1: All users will need accounts in order to partake in tests.

Option 2: Some tests may be public, and any user, even ones without an account, may take it at any time.

Decision: We are going to require that all users have an account. There is not much benefit for outsiders to take exams and courses, as it would just overload the work of the administrative users. Furthermore, by requiring all users to have accounts, there will be more long term improvement in the student's skills.

4. How will we make sure students do not work on the exams together?

Option 1: Create a question bank and get random questions out of them on each exam

Option 2: Change the order of the questions



**Decision:** Although this may be a little more work to the administrators we thought it was better to have a larger question bank for the exam and have random questions from a question bank. This makes it so that even if someone has already taken the exam before another student, the student will not necessarily see the same questions or in the same order.

## Non-Functional Issues

1. Which server framework are we using for our backend?

Option 1: Django

Option 2: Laravel

Option 3: Flask

**Decision:** We had different experiences with Laravel and Flask, which could have made this web-app easier to make, but we thought it might be better for us to use this chance to learn a new framework together as we have experience in creating web applications. We chose Django because it deals very well with concurrency as well as an easy step into learning because of its well made documentation on its website.

2. How will we host our backend service?

Option 1: Amazon Web Service

Option 2: Offline third party apache interpreter and SQL simulator (XAMPP)

**Decision:** We decided to use an offline apache interpreter and SQL simulator like XAMPP as that would allow us to code the entire backend on our own without any restrictions. Using Amazon Web Services come with a few strings attached like using their NoSQL database, and coding a server using their lambda functions. As we are unfamiliar with these languages, we decided to go with XAMPP.

3. Which database software are we using?

Option 1: MySQL

Option 2: MongoDB

Option 3: FireBird

**Decision:** We decided to use MySQL because it was the database software we had the most experience with. Also, we have two people who took CS 348 which teaches queries in MySQL. Therefore, if we were to choose MySQL we thought we wouldn't have a lot of troubles with data in queries. We chose the most confident software for databases because when we have lots of data it is hard to keep track of incorrect information and it gets harder to test. It gives us more room for experimenting with a query language we already know.

**4. How will people be signing up in order to login?**

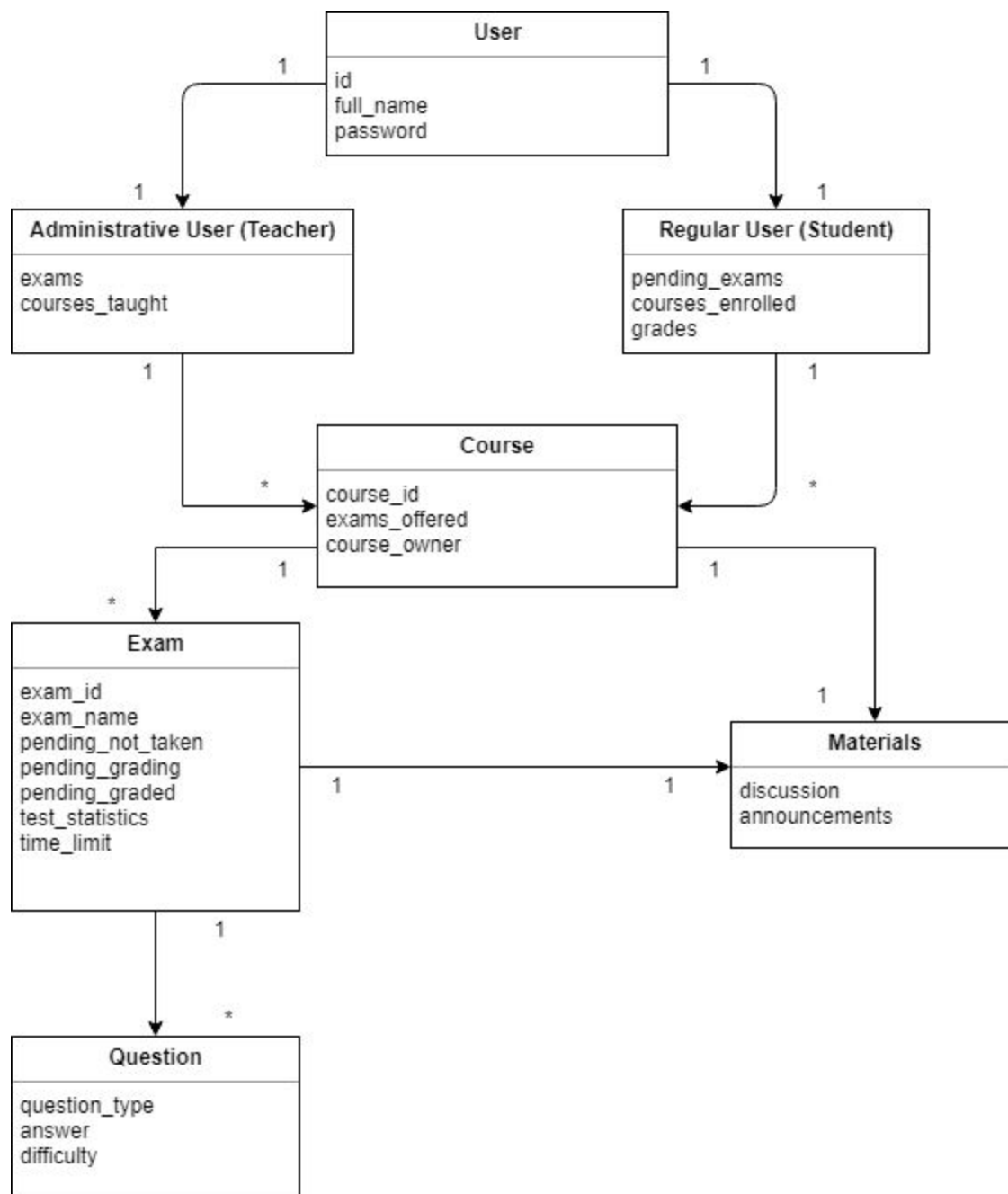
**Option 1:** By connecting to Facebook or Email

**Option 2:** By registering a completely new account

**Option 3:** Both

**Decision:** We chose to use both connecting an email and creating a new account because it would be easier for us to organize people's information that way and use emails for notifications. There would be an option for people to connect to their email if they would prefer notifications for upcoming deadlines or a heads up if they haven't connected in a while.

## Design Details



## Description of Class Diagram

### User:

- **Id:** Each user will have a unique id that they will use for login as well as for identification.
- **Full\_name :** The full name of the user
- **Password:** Password that the user will use to log into their account

### Administrative User (Teacher):

- **Courses:** The list of courses that the teacher is currently teaching
- **Exams:** All the exams that are currently being offered by the teacher. These may be open or closed as the course progresses.

### Regular User (Student):

- **Pending exams:** When a student is enrolled in a course, they will receive notifications whenever an exam they are required to take has been opened. They may choose to be notified via email, directly through the website, or both.
- **Courses enrolled:** All the courses that the student is taking at the current moment.
- **Grades:** Grades for all exams that the student has taken, whether they are pending grading or completely graded.

### Course:

- **Course ID:** A unique way to identify courses. Each course will have it's own ID.
- **Exams offered:** Throughout a course, exams will be opened or closed depending on how the administrative user decides to give out the tests.
- **Course owner:** The teacher of the course. There may be multiple if administrative users wish to collaborate on a course.

### Exam:

- **Exam id and name:** These are ways to identify which test the exam is, as well as which course it is being offered in.
- **Pending:** The exam can have multiple states. When a student has not taken it, it will be pending to be taken by the student. After the student has taken it, it will be graded automatically according to the answers provided by the teacher. However, if there are essay questions or short answer questions, the test will be pending to be graded by the teacher manually, such that they can review the exam and give an appropriate score according to the student's work.
- **Test statistics:** This will be a way to see the statistics of the exam, such as average score. It can be public or private depending on the administrative user's choice.
- **Time limit:** Administrative users may decide to put a time limit on the exam. If the user does not finish the test within the time limit, they will receive the score they had at the time of the time's expiration.

### Materials:

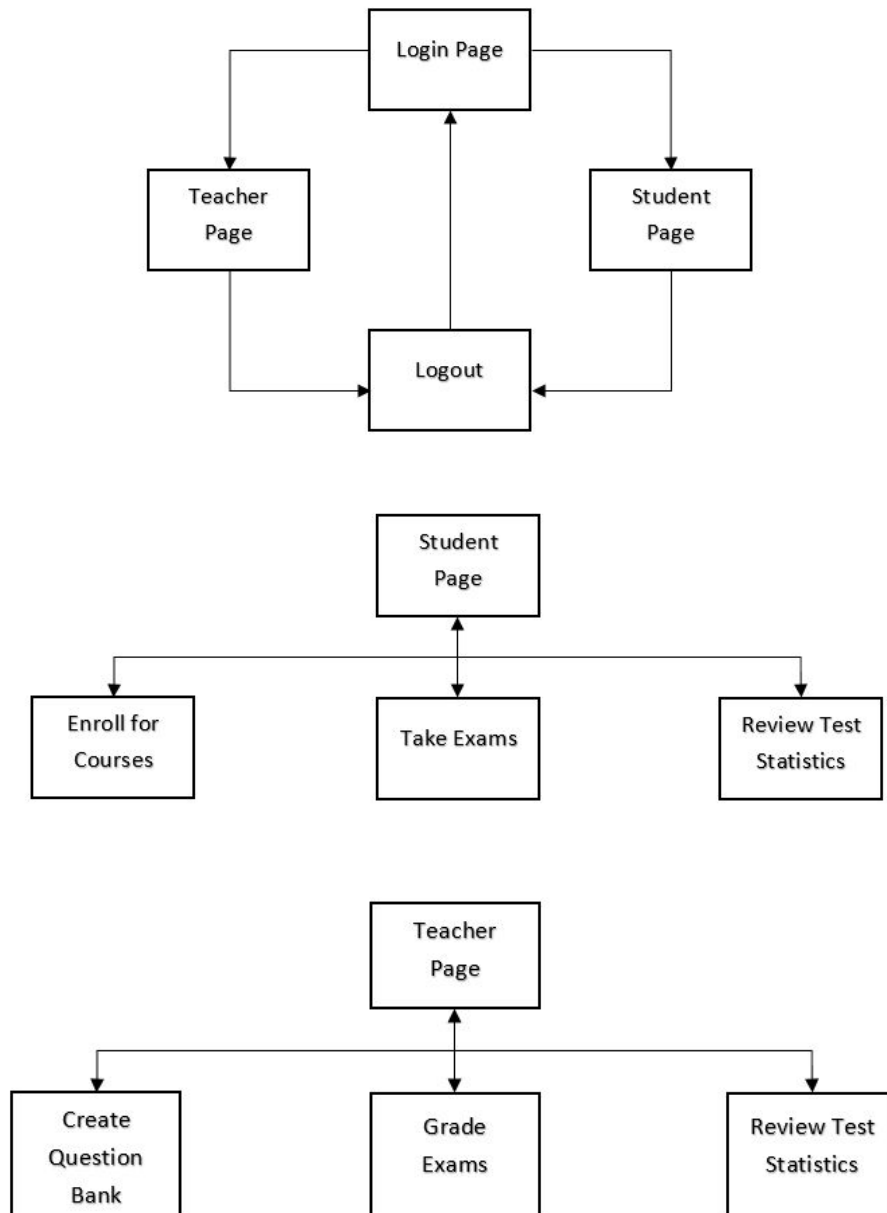
- **Discussion:** The discussion section will be similar to a forum, in which students can discuss amongst themselves as well as with teachers about course materials and exams.
- **Announcements:** Whenever a new exam has been opened or closed, students will receive notifications regarding exams they are eligible to take.

**Question:**

- **Question\_type:** Our web app will be offering both multiple choice questions as well as free response essay type questions. Multiple choice questions will be graded according to an answer key as soon as the student has finished his or her exam. In the case of free response questions, the test will be sent to the teacher governing the course, and they will be able to read over and grade the work. Questions are drawn from a pool of questions, so that collaboration and dishonest activity will be discouraged.
- **Difficulty:** Each question will have difficulty levels, such that when a student is provided with an exam, each student will have an equal opportunity to achieve the grade they want.

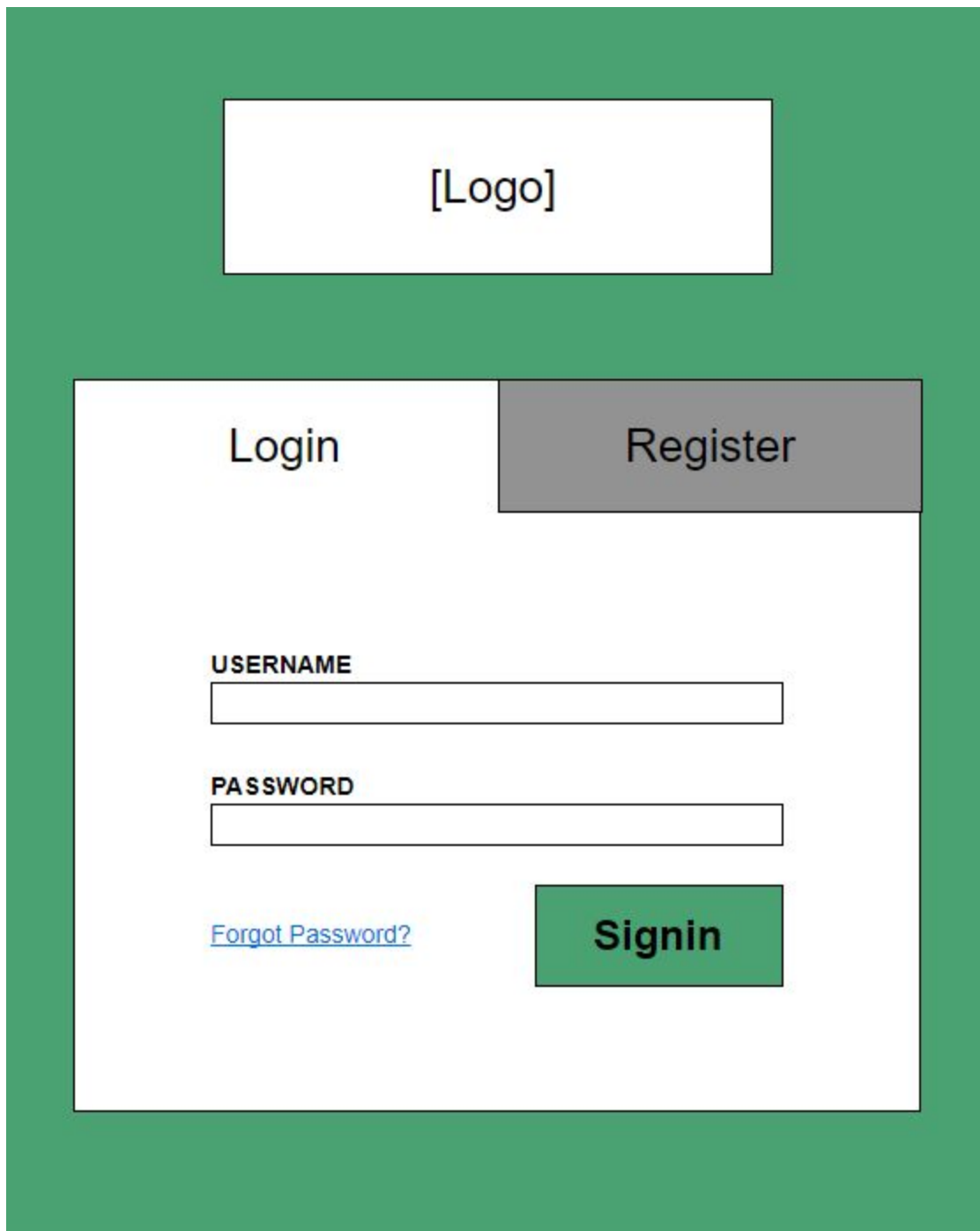
## Navigation Flow Map

Our navigation flow map emphasizes on the ease of access and simplicity of redirection of our web application. There are very few and clearly marked options for both the users and the teachers. The login page gives fast access through Facebook and Email.



## User Interface Mockups

### Login Screen



The mockup shows a login screen with a green background. At the top center is a white box labeled "[Logo]". Below it is a white container with a grey "Register" button on the right and a "Login" label on the left. Inside the container are two input fields: "USERNAME" and "PASSWORD". Below the "PASSWORD" field is a blue link "Forgot Password?". At the bottom right of the container is a green "Signin" button.

[Logo]

Login Register

USERNAME

PASSWORD

[Forgot Password?](#)

Signin

## Exam Screen

The image shows a screenshot of an exam interface. It has a green header bar at the top. Below the header, there is a white box containing the text "EXAMPLE TEST TITLE". Below this, there are two question sections, each with a white background and a green border. The first section is titled "Question 1" and contains the text "This is an example of a short answer question." followed by a large text input field with a vertical scrollbar. The second section is titled "Question 2" and contains the text "This is an example of a mutiple choice question." followed by three radio button options: "Choice 1", "Choice 2", and "Choice 3".

**EXAMPLE TEST TITLE**

**Question 1**

**This is an example of a short answer question.**

**Question 2**

**This is an example of a mutiple choice question.**

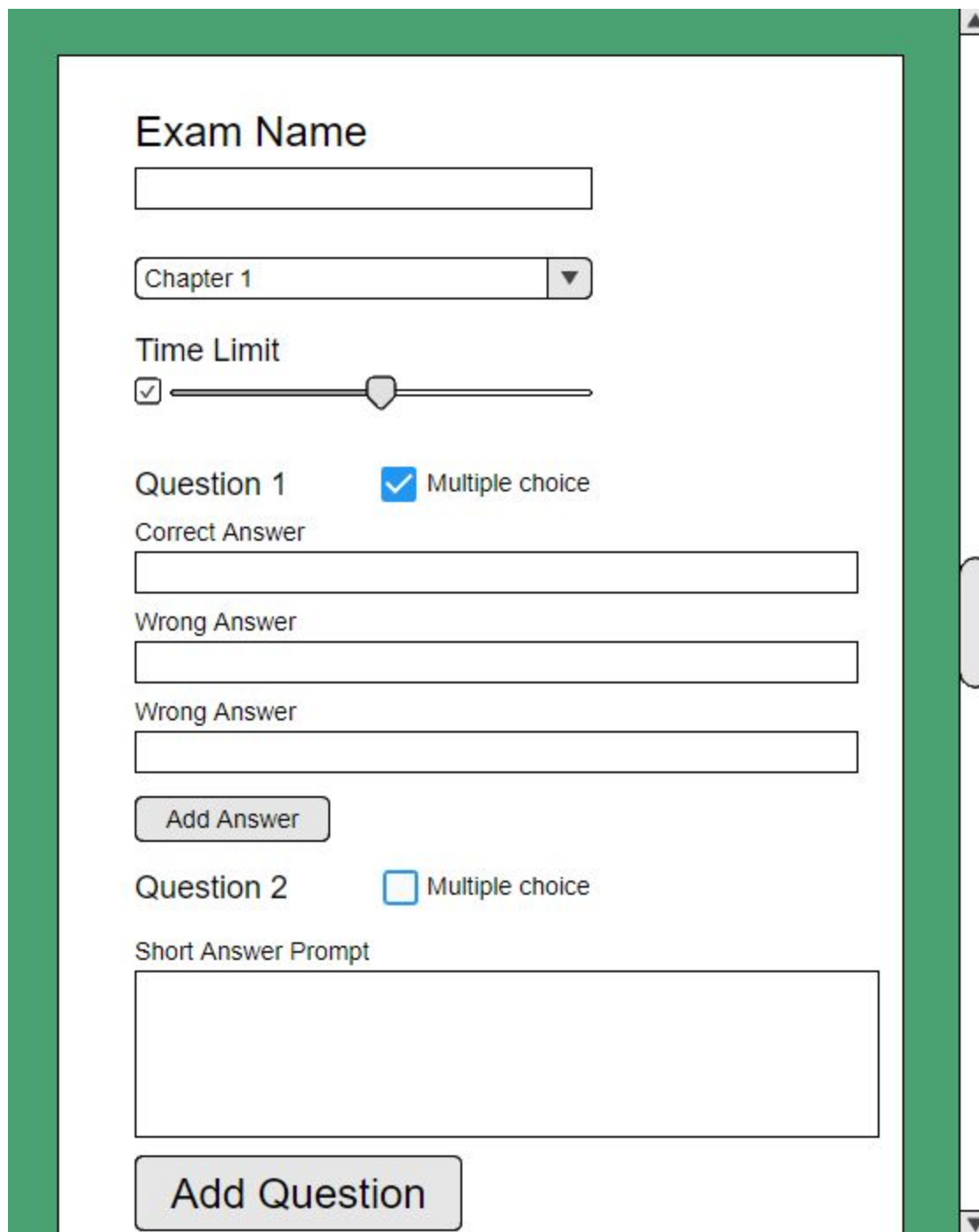
☐ Choice 1

☐ Choice 2

☐ Choice 3



## Exam Creation Screen




The image shows a web-based 'Exam Creation Screen' with a green border. It contains several input fields and controls for creating an exam. At the top, there is a text input for 'Exam Name'. Below it is a dropdown menu currently showing 'Chapter 1'. Underneath is a 'Time Limit' section with a checked checkbox and a horizontal slider. The screen lists two questions. 'Question 1' is a multiple-choice question, indicated by a checked checkbox. It has three text input fields for 'Correct Answer', 'Wrong Answer', and another 'Wrong Answer'. An 'Add Answer' button is located below these fields. 'Question 2' is a short-answer question, indicated by an unchecked checkbox. It has a single large text input field for the 'Short Answer Prompt'. At the bottom of the form is a large 'Add Question' button.

Exam Name

Chapter 1

Time Limit

☒ 

Question 1 ☒ Multiple choice

Correct Answer

Wrong Answer

Wrong Answer

Add Answer

Question 2 ☐ Multiple choice

Short Answer Prompt

Add Question