##### Question Paper & Answer Management system with multi-level approval and user permissions in your ASP.NET MVC application

**Deadline:** [**19-02-2024**]

**Setup and Configuration:**

Begin by creating a new ASP.NET MVC project in Visual Studio.

Set up your database using Entity Framework Data First approach.

**Authentication and Authorization:**

Implement user login and registration functionalities.

Define user roles and permissions to manage different levels of access.

**User Roles and Permissions:**

Identify different user types (e.g., admin, teacher, student).

Assign specific permissions to each user type (e.g., admin can create, edit, and delete question papers, while teachers can only create question papers, students can only view).

**Question Paper Creation:**

Create a UI for users to input questions, answers, and any other relevant details.

Design database tables to store question paper details, questions, answers, etc.

Implement logic to save question papers to the database.

**Workflow Management:**

Define the workflow for question paper approval. This might include multiple levels of approval.

Implement logic to handle the workflow, including sending notifications to the appropriate users for approval.

**Multilevel Approval:**

Each level of approval might require a different set of permissions.

Implement logic to manage the approval process, including rejecting or accepting the question paper.

**UI Design:**

Create user-friendly interfaces for login, registration, question paper creation, and approval workflows.

Utilize HTML, CSS, and JavaScript along with any frontend frameworks like Bootstrap or jQuery UI for responsive and interactive design.

**Testing:**

Perform user acceptance testing to validate that the system meets the requirements.

**Workflow:**

**Question Paper Creation Workflow:**

Step 1: Teacher creates a draft question paper.

Step 2: Teacher submits the draft for review.

Step 3: Admin reviews the draft and either approves or sends it back for revision.

Step 4: If approved, the question paper becomes active and available to students.

**Database Architecture:**

**User Table:**

Stores user details including username, password hash, email, and role.

**Question Paper Table:**

Stores information about each question paper such as title, description, creation date, status, etc.

**Question Table:**

Contains questions associated with each question paper, including the question text, options, correct answer, etc.

**Approval Table:**

Tracks the approval status of each question paper, including who approved it and when.

**Permission Table:**

Defines permissions for different user roles (e.g., admin, teacher, student).

**Permissions:**

**Admin:**

Can create, edit, and delete question papers and managing users.

Can review and approve or reject question papers.

**Teacher:**

Can create, edit, and delete question papers.

Can submit question papers for review.

**Student:**

Can view active question papers.

**Flow of the Application:**

**User Authentication and Authorization:**

Users log in using their credentials.

Upon login, their role and permissions are loaded.

**Question Paper Creation:**

Teachers navigate to the "Create Question Paper" page.

They input details such as title, description, and questions.

After creating the draft, they can either save it or submit it for review.

**Approval Workflow:**

When a teacher submits a question paper for review, an entry is made in the approval table with the status pending and the ID of the question paper and the teacher.

The admin receives a notification or can see pending question papers in their dashboard.

The admin can review the question paper and either approve it or send it back for revision.

If approved, the status of the question paper changes to active, and it becomes available to students.

**Access Control:**

The application checks the user's role and permissions at various points:

Admins have access to all functionalities.

Teachers can create and submit question papers.

Students can view active question papers.

**UI Flow:**

The UI presents different options and functionalities based on the user's role and permissions.

For example, an admin sees options for managing users, approving question papers, etc., while a teacher sees options for creating and submitting question papers.

**Error Handling and Notifications:**

Users receive notifications about the status of their submitted question papers (e.g., whether it was approved or rejected).

Error handling mechanisms are implemented throughout the application to handle unexpected scenarios gracefully.

By following this workflow and architecture, you can create a robust question paper management system with multi-level approval and user permissions in your ASP.NET MVC application.

**Enhanced Database Architecture:**

**User Table:**

Stores user details including username, password hash, email, and role.

Question Paper Table:

Stores information about each question paper such as title, description, creation date, status, etc.

**Question Table:**

Contains questions associated with each question paper, including the question text, options, correct answer, etc.

**Answer Table:**

Stores the answers submitted by students for each question paper.

Includes fields like user ID, question ID, answer text, submission timestamp, etc.

**Approval Table:**

Tracks the approval status of each question paper, including who approved it and when.

**Permission Table:**

Defines permissions for different user roles (e.g., admin, teacher, student).

Admins can review and approve question papers.

Teachers can create and submit question papers.

Students can view and attempt question papers.

Question Paper Viewing and Answer Submission:

Students navigate to the "View Question Paper" page.

**Admin Panel:**

Allow admins to view and manage users (create, edit, delete).

Provide functionality to assign roles and permissions to users.

**Database Enhancement:**

Add an Answer Table: This table will store the answers submitted by students for each question paper.

Fields may include user ID, question ID, answer text, submission timestamp, etc.

**UI Implementation:**

Create a UI for students to view question papers and submit their answers.

When students view a question paper, they should see the questions along with options to select their answers.

Provide a form or interface for students to submit their answers.

**Logic Implementation:**

When a student submits an answer:

Capture the user's ID, question ID, selected answer, and submission timestamp.

Save this information in the Answer Table.

**Access Control:**

Ensure that only students have access to the functionality of viewing and answering question papers.

Students should not have access to functionalities such as creating or approving question papers.

**Integration with Workflow:**

Update the workflow to include the student's role:

After a question paper is approved, it becomes available for students to view and answer.

Ensure that students can only view and submit answers for question papers that are in the active status.

**UI Flow:**

Design the UI to guide students through the process of viewing and answering question papers.

Provide clear instructions and intuitive controls for selecting answers and submitting them.

**Error Handling and Notifications:**

Implement error handling to notify students if there are any issues with submitting their answers.

Provide notifications to students upon successful submission of their answers.

**Data Retrieval and Analysis:**

Implement functionality for teachers or admins to view submitted answers for analysis.

Allow them to view statistics or reports on student performance for each question paper.

By incorporating these steps, the question paper management system will have a complete Answer module for students, allowing them to view question papers and submit their answers efficiently.