Login Page Functional Requirements

1. Purpose

The Login Page serves as the gateway for all users (Students, Teachers, and Admins) to access the web app. It ensures secure authentication, directs users to their respective dashboards upon successful login, and provides a mechanism to recover forgotten passwords.

2. Functional Requirements

2.1. User Authentication

- **Input Fields:**
  - **Username/Email Field:**
    - Accepts a valid email address or username.
    - Must include validation to ensure the format is correct.
  - **Password Field:**
    - Accepts a user password.
    - Should hide characters as they are typed.
    - Include a "show/hide" toggle if necessary.
- **Login Button:**
  - Initiates the authentication process.
- **Validation & Error Handling:**
  - Display clear error messages if:
    - The email/username is not in the correct format.
    - The password field is empty.
    - Credentials do not match any existing records.

2.2. Forgot Password Workflow

- **Forgot Password Link/Button:**
  - Visible on the login page.
  - Clicking the link redirects the user to the Forgot Password Page.
- **Email Verification Process:**
  - **Input Field for Email:**
    - Users must enter the email associated with their account.
    - Validate the email format and confirm existence in the database.
  - **Send Verification Code:**
    - A verification code is automatically sent to the provided email address.
    - The code should be time-sensitive (expires after a predefined duration).
  - **Verification Code Entry:**
    - Provide a field for users to enter the received code.
    - Validate the code for correctness and timeliness.
  - **Password Reset Option:**
    - Once the code is verified, allow the user to enter a new password.
    - Enforce password complexity requirements (Need t implement this).

2.3. User Experience Considerations

- **Responsive Design:**
  - The login page should be accessible and visually consistent across devices (desktop, tablet, mobile).
- **Accessibility:**
  - Ensure that the page meets accessibility standards (e.g., proper labels for screen readers, adequate contrast, keyboard navigation support).
- **User Feedback:**
  - Provide immediate visual feedback on actions (e.g., loading spinner after clicking "Login", success message upon successful password reset).

3. Non-functional Requirements

- **Security:**
  - Ensure that the connection is secure (use HTTPS).
  - Encrypt sensitive data such as passwords in storage and during transmission.
- **Performance:**
  - The login process should be responsive, with minimal latency during authentication.
- **Scalability:**
  - The system should handle multiple concurrent login requests efficiently.

4. Error and Exception Handling

- **Incorrect Credentials:**
  - Display a general error message such as "Invalid email or password."
- **Account Lockout:**
  - Implement a temporary lockout after a predefined number of failed login attempts.
- **Timeouts:**
  - Handle timeouts gracefully by providing a retry option or a clear error message.

5. Integration Points

- **Backend Authentication Service:**
  - The login page communicates with the backend authentication service to verify user credentials.
- **Database (MongoDB):**
  - User credentials and account statuses are stored and retrieved from MongoDB.
- **Email Service:**
  - Integration with an email service provider to send out verification codes during the password recovery process.

Admin Dashboard Functional Requirements

1. Overview

The Admin Dashboard is the control center for administrative users. It provides a set of tools to manage the user base and maintain system integrity. The dashboard is divided into six primary functional areas:

- Add User
- Enable/Disable User
- Reset Password
- Show All Users
- Update Phone
- Update Email

Each function is accessible via a dedicated tab and is integrated with the backend database (MongoDB) and other system services.

---

2. Functional Requirements

2.1. Add User

- **Purpose:**
  Enable admins to add new users to the system (either Students or Teachers).
- **Input Fields:**
  - Name
  - Role selection (Student/Teacher)
  - Email(if role = student, then get parents number otherwise get teachers number)
  - Phone number (if role = student, then get parents number otherwise get teachers number)
  - Grade (only for student)
  - School name(only for student)
  - Gender
- **Validations:**
  - Ensure the email address follows the proper format and is unique in the system.
  - Validate that the phone number is in the correct format – only numbers.
  - Confirm that all required fields are populated before submission.
- **Process:**
  1. Admin fills out the form and submits the data.
  2. The system creates a new user record in MongoDB.
  3. A confirmation message is displayed upon successful creation.
- **Error Handling:**
  - Display clear error messages for missing or invalid input.
  - Prevent duplicate user creation by checking for existing email addresses.

2.2. Enable/Disable User

- **Purpose:**
  Allow admins to control user access by enabling or disabling user accounts.
- **User Interface:**
  - A search box to type in the name of the user
  - List displaying a user with their current status (enable/disable).
  - A toggle button or switch beside each user to change their status.
- **Validations:**
  - Confirmation dialog before changing the status to prevent accidental toggling.
- **Process:**
  1. Admin selects a user and toggles the enable/disable option.
  2. The user's status is updated in MongoDB.
  3. The change is immediately reflected in the UI and affects login capabilities.
- **Error Handling:**
  - Notify the admin if the status update fails and provide retry options.

2.3. Reset Password

- **Purpose:**
  Allow admins to reset a user's password upon request.
- **Input Fields:**
  - A search box to type in the name of the user
  - Selection of the user whose password needs to be reset.
  - New password entry field.
- **Validations:**
  - Enforce password complexity rules (e.g., minimum length, special characters).
  - Confirm new password via a second entry if required.
- **Process:**
  1. Admin selects a user and enters a new password.
  2. Upon submission, the system updates the user's password in MongoDB (using encryption).
  3. A confirmation message is displayed to the admin.
- **Error Handling:**
  - Display an error if the password does not meet complexity requirements.
  - Provide appropriate feedback if the password reset process fails.

2.4. Show All Users

- **Purpose:**
  Display a comprehensive list of all users in the system.
- **User Interface:**
  - A table or list view that includes details such as user name, role, email, and account status.
  - Options to search, sort, or filter the user list by various criteria (e.g., role, enabled status).
- **Process:**
  1. On accessing this tab, the system retrieves all user records from MongoDB.
  2. The list is displayed with pagination if necessary for large datasets.
- **Error Handling:**
  - Inform the admin if there is an issue retrieving user data.
  - Allow retrying the data fetch operation.

2.5. Update Phone

- **Purpose:**
  Enable admins to update phone numbers for users.
- **Input Fields:**
  - For Students: Parent's phone number.
  - For Teachers: Their own phone number.
- **Validations:**
  - Ensure the new phone number is in the correct format. – only numbers
  - Confirm the phone number is not left blank.
- **Process:**
  1. Admin selects a user and enters a new phone number.
  2. The system updates the phone number in MongoDB.
  3. A success message is displayed upon completion.
- **Error Handling:**
  - Notify the admin if the phone update fails due to invalid input or a system error.

2.6. Update Email

- **Purpose:**
  Allow admins to update email addresses for users.
- **Input Fields:**
  - For Teachers: A single email field.
  - For Students: Two email fields – one for the student and one for the parent.
- **Validations:**
  - Validate that the email addresses are correctly formatted.
  - Check for the uniqueness of the new email address in the system.
  - Ensure both student and parent emails are handled correctly if applicable.

- **Process:**
    1. Admin selects a user and updates the email(s).
    2. The system updates the email information in MongoDB.
    3. A success message confirms the update.
- **Error Handling:**
    - Provide specific error messages for invalid email formats or duplicate emails.
    - Inform the admin if the update process encounters any issues.

---

3. Non-functional Requirements

- **Security:**
    - Ensure all admin operations are accessible only to authenticated and authorized admin users.
    - Use secure protocols (HTTPS) for all communications.
    - Encrypt sensitive data such as passwords and personal contact information.
- **Performance:**
    - Operations (add, update, reset) should execute with minimal latency.
    - The system should handle multiple concurrent admin operations without performance degradation.
- **Usability:**
    - The dashboard must be intuitive and responsive, with clear navigation between tabs.
    - Provide immediate feedback (e.g., loading indicators, confirmation messages) during user interactions.
- **Scalability:**
    - The system should efficiently manage large user datasets, especially in the “Show All Users” view.

---

4. Integration Points

- **MongoDB:**
    - All CRUD operations for user management (add, update, reset, status changes) interact with the MongoDB database.
- **Email Service (if applicable):**
    - For password resets and email updates, integration with an email service provider ensures notifications are sent out as needed.

Teacher Dashboard – Functional Requirements

1. Overview

The **Teacher Dashboard** is designed to provide teachers with the tools they need to manage students, create and track assignments, and generate reports. It is divided into five primary tabs:

1. **Dashboard**
2. **List of Students**
3. **Question Bank**
4. **Create Assignments**
5. **Generate Reports**

Each tab focuses on a distinct set of tasks, ensuring a clear and intuitive workflow for teachers.

---

2. Tabs and Their Functional Requirements

2.1. Dashboard (Home Tab)

Purpose:

- Provide a quick overview of key metrics, including:
    - **Total number of students** the teacher manages.
    - **Number of active assignments** currently assigned.

Functional Requirements:

Display Summary Data:

- Show real-time counts of total students and active assignments.
- Data is pulled from the database based on the teacher's user ID or assigned classes.

Non-functional Requirements:

- **Performance:** Summary counts should load quickly without noticeable delay.
- **Usability:** Present data in a clear, concise format (e.g., simple cards or labeled sections).

2.2. List of Students

Purpose:

- Provide a searchable list of all students associated with the teacher.
- Allow the teacher to quickly review student details such as name, parent email, gender, school, grade, and account status.

Functional Requirements:

1. **Table View:**
    - Display students in a tabular format with columns for:
        - Student Name
        - Email
        - Parent Email
        - School
        - Grade
        - Status (Active/Inactive)
2. **Search Functionality:**
    - A search box to filter students by name or email.
    - Should handle partial matches (e.g., searching “sid” should show “sid@student.com”).

Validations & Error Handling:

- **Empty Table:** If no students are found, display a “No students available” message.
- **Search Errors:** If an invalid search query is made (e.g., special characters), show an appropriate message or simply return no results.

2.3. Question Bank

Purpose:

- Provide an overview of the available questions, organized by topic and sub-topic.
- Display the number of questions for each difficulty level (Easy, Medium, Hard).

Functional Requirements:

1. **Question Bank Overview:**
    - List all **Topics** and **Sub-Topics**.
    - For each sub-topic, show:
        - **Total Questions** count.
        - Distribution of questions by difficulty (Easy, Medium, Hard).
2. **Data Source:**
    - All question details are fetched from MongoDB.
    - The teacher should be able to scroll or page through the data if there are many records.

Validations & Error Handling:

- **Data Retrieval Errors:** If the database query fails, display an error message (e.g., "Unable to load question bank. Please try again later.").
- **Empty Results:** If no questions exist for a topic, display a zero count.

2.4. Create Assignments

Purpose:

- Enable teachers to view and manage existing assignments.
- Allow teachers to create new assignments by selecting topics, sub-topics, and assigning them to specific students or groups.

2.4.1. Current Assignments

1. **Assignments Table:**
    - Columns:
        - **Topic**
        - **Sub-Topics**
        - **Created Date**
        - **Deadline**
        - **Status** (Active/Inactive)
        - **Assigned Students**
        - **Grade**
    - Data is filtered by the logged-in teacher, so only their assignments are shown.
2. **Assignment Status Management:**
    - Teachers should see whether an assignment is active or completed/expired.
    - (Optional) Provide an option to edit or remove assignments if needed.

2.4.2. Create New Assignment

1. **Topic & Sub-Topic Selection:**
    - **Topic Dropdown:** Displays all available topics from the question bank.
    - **Sub-Topic Dropdown:** Once a topic is selected, the relevant sub-topics appear.
    - **Multiple Sub-Topic Selection:** Teachers can select one or more sub-topics simultaneously.
2. **Student Selection:**
    - **Continue to Student Selection** button triggers the next step.
    - **Select Students by Grade:**
        - Teachers can choose one or more grades.
        - Teachers can select individual students within a grade or select "All" to assign to every student in that grade.
    - **Assignment Uniqueness:**

        - If an assignment has already been assigned to a student, show an error message preventing duplication.
3. **Set Deadline:**
    - **Deadline Date & Time:** Teachers must choose a future date/time.
    - Validate that the deadline is not in the past.
4. **Submit Assignment:**
    - On submission, the new assignment is saved to the database and displayed in the "Current Assignments" table.

Validations & Error Handling:

- **Mandatory Fields:** Topic, at least one sub-topic, at least one student, and a valid deadline are required.
- **Duplicate Assignments:** If the teacher tries to reassign the same sub-topic to the same student(s), display an error.
- **Past Deadline:** Show an error if the deadline date/time is earlier than the current date/time.

2.5. Generate Reports

Purpose:

- Provide insights into student activity and performance.
- Allow the teacher to generate both summary and detailed reports, with an option to send summaries to parents.

2.5.1. Generate Report Action

1. **Generate Report Button:**
    - When clicked, the system compiles the latest data on student activities and progress.
2. **Tabs for Different Views:**
    - **Tab 1: Overview**
        - High-level summary of student performance and activity (e.g., number of completed assignments, average scores, daily practice consistency).
        - This summary can be shared with parents.
    - **Tab 2: Detailed Data**
        - A more granular, tabular view for the teacher's reference (e.g., assignment completion dates, question-level performance, etc.).
    - **Tab 3: Recommendations**
        - (Planned for future) Potential areas for improvement, suggestions for additional practice, etc.
3. **Sending Reports to Parents:**
    - (Optional) Provide an option to email the Overview to parents.
    - If implemented, confirm success or failure of email delivery.

Validations & Error Handling:

- **Data Retrieval Issues:** If data is unavailable or if there's a server error, display an appropriate message (e.g., "Unable to generate reports at this time.").
- **No Activity Data:** If a student has no recent activity, handle gracefully in the summary.

---

3. Non-functional Requirements

1. **Security:**
    - Only authorized teachers can access this dashboard.
    - All data transactions (CRUD operations) must be secure and follow best practices (e.g., HTTPS, encrypted passwords).
2. **Performance:**
    - Loading of lists (students, assignments, question bank) should be efficient and not cause timeouts.
    - Generating reports should complete in a reasonable time.
3. **Usability:**
    - The layout should be intuitive, with clear navigation between tabs.
    - Provide user-friendly messages and tooltips where necessary.
4. **Scalability:**
    - The dashboard should handle an increasing number of students, assignments, and questions without significant performance degradation.
5. **Maintainability:**
    - Code should be structured to allow easy updates or feature additions, such as expanding the reporting system or refining the assignment creation process.

Student Dashboard – Functional Requirements

1. Overview

The **Student Dashboard** provides students with a clear view of their assigned work, including assignments that are currently in progress and those that have been completed. It allows students to resume ongoing work or review past assignments to see their performance and correct answers.

---

2. Key Features

2.1. Assignment List & Status Grouping

1. **Assignment Grouping:**
    - **In Progress:** Lists all assignments the student has started but not yet completed or whose deadline has not passed.
    - **Completed:** Lists all assignments the student has finished or whose deadline has expired.
2. **Assignment Cards/Rows:**
    - Each assignment entry displays:
        - **Topic** and **Sub-Topic** name(s).
        - **Deadline** (date by which the assignment must be completed).
        - **Progress Metrics** (e.g., total questions, attempted questions, number correct).
3. **Expandable Details:**
    - When the student clicks or expands an assignment, additional information is revealed:
        - **Total Questions** in the assignment.
        - **Attempted**: Number of questions the student has answered so far.
        - **Correct**: Number of questions answered correctly.
        - **Deadline**: Confirmation of the final submission date and time.

2.2. Action Buttons

1. **Resume:**
    - Appears on assignments that are in progress.
    - Clicking **Resume** navigates the student to the question page where they can continue answering questions.
    - Once a student completes all the questions from an assignment the label name of this button changes to COMPLETE
    - Validation: If the deadline has passed, the system should either:
        - Prevent the student from resuming and show a relevant message (e.g., “Deadline exceeded”), OR
        - Automatically move the assignment to “Completed” status.
2. **Review:**

- Appears once the student has attempted at least some portion of the assignment (and possibly after it's completed).
- Clicking **Review** shows:
  - A detailed list of all questions in the assignment.
  - Correct solutions and the student's chosen answers.
- Validation:
  - If the assignment is not started or no questions have been answered yet, the "Review" button might be disabled or hidden.
  - The system may only allow reviewing after the assignment deadline or after the student has submitted all questions.

---

3. Additional Functional Details

1. **Dynamic Status Updates:**
   - As the student answers questions, the assignment's status (In Progress vs. Completed) and progress metrics update in real-time or upon refreshing the dashboard.
2. **Deadline Handling:**
   - The dashboard should visually indicate if an assignment is close to or past its deadline (e.g., color-coding or a warning message).
   - Once the deadline is reached, the assignment automatically moves to "Completed" (even if the student did not attempt all questions).
3. **Data Synchronization:**
   - All assignment data (progress, attempt count, correctness) is stored in MongoDB.
   - The dashboard fetches updated data whenever the page loads or refreshes, ensuring the student sees the latest progress.

---

4. Validations & Error Handling

1. **Invalid Assignment Access:**
   - If a student tries to access an assignment that does not belong to them (e.g., via a direct URL), display an error message and deny access.
2. **Deadline Exceeded:**
   - If the student clicks **Resume** after the deadline has passed, display an error or move the assignment to "Completed."
3. **Data Fetching Errors:**
   - If the system cannot retrieve assignment data, display a user-friendly message (e.g., "Unable to load assignments. Please try again later.").
4. **Review Restrictions:**
   - If the assignment is not yet started or has no answered questions, the **Review** button should be disabled or hidden.

---

5. Non-functional Requirements

1. **Performance:**
    - Loading the assignment list and status should be quick, even with many assignments.
    - Updating the assignment status or progress should be near real-time or upon page refresh.
2. **Usability & Accessibility:**
    - The interface should be intuitive, with clear labels and icons for **Resume** and **Review**.
    - Use responsive design principles so students can access the dashboard on various devices.
3. **Security:**
    - Only authenticated students should access their own assignments.
    - All communication should be over secure protocols (HTTPS).
4. **Scalability:**
    - The system should handle large numbers of students and assignments without performance degradation.

Question Page – Functional Requirements

1. Overview

The **Question Page** is where students actively work on an assignment. It presents one question at a time, along with multiple-choice answers, and allows students to submit their response, view immediate feedback, and move to the next question.

---

2. Key Features

2.1. Question Display

1. **Question Text:**
    - Displays the current question in a clear format.
    - Shows relevant details (e.g., topic, sub-topic, difficulty level, question number out of total).
2. **Answer Options (Multiple Choice):**
    - Four options, with only one correct answer.
    - The student can select exactly one option at a time.
3. **Progress Bar or Indicator (Optional Enhancement):**
    - A visual cue (like a progress bar) indicating how many questions have been completed out of the total.

2.2. Student Actions

1. **Select Option:**
    - The student clicks on one of the four options to select their answer.
    - Only one option can be selected at a time.
2. **Submit Button:**
    - When clicked, the system evaluates the student's selected option.
    - If the answer is correct, display a checkmark or success indicator.
    - If the answer is incorrect, display an 'X' or failure indicator.
    - The **solution** or explanation is then shown to the student (e.g., how to arrive at the correct answer).
3. **Next Button:**
    - Appears (or becomes enabled) only after the student has submitted an answer.
    - Clicking **Next** loads the next question in the assignment.
    - The student's progress metrics (e.g., attempted questions, number correct) are updated in the database only after clicking **Next**.
    - If there are no more questions, clicking **Next** could navigate to a completion summary or back to the dashboard.
4. **Back to Dashboard Button:**
    - Returns the student to the **Student Dashboard** page.
    - Students cannot log out directly from the question page; they must go back to the dashboard to log out.

---

3. Validations & Error Handling

1. **No Option Selected:**
    - If the student clicks **Submit** without selecting an option, display an error or disable the button until an option is chosen.
2. **Incorrect Answer:**
    - Provide clear feedback indicating which answer was correct.
    - Optionally, highlight the chosen incorrect option and the correct option.
3. **Connectivity Issues:**
    - If there is a problem saving the response or fetching the next question (e.g., network failure), display an error message and allow the student to retry.
4. **Question Not Found or Already Completed:**
    - If a question is unavailable or the assignment is already completed, redirect the student to the dashboard with an appropriate message.

---

4. Data Flow & Integration

1. **Database Updates (MongoDB):**
    - Upon clicking **Next**, the student's response is saved (correct/incorrect, timestamp, etc.).

    - The assignment's metrics (attempted, correct) are updated accordingly.
2. **Real-time Feedback:**
    - After submitting an answer, the app fetches and displays the solution from the database (or local data structure).
3. **Assignment Progress Sync:**
    - The updated metrics (e.g., total attempted, total correct) are reflected in the **Student Dashboard** when the student returns or refreshes the dashboard.

---

5. Non-functional Requirements

1. **Usability:**
    - The page layout should be simple and clear, focusing on the question and answer options.
    - Immediate visual feedback (tick or X) upon submission is crucial for user engagement.
2. **Performance:**
    - Loading the next question should be seamless, with minimal delay.
    - Submitting an answer should be quick, and feedback should appear promptly.
3. **Security:**
    - Ensure that only authorized, logged-in students can access the question page.
    - Use secure protocols (HTTPS) for data transmission.
4. **Scalability:**
    - The system should handle multiple students answering questions simultaneously without performance issues.
