Task Priority and Effort Estimate Version

Task ID	Task	Task Description	Priority 1: Highest 5: Lowest	Effort Estimate (Days)	Map to	Acceptance Criteria	Acceptance Tests
Frontend 1.1	**Implement a Side-Bar Component**	Develop a side-bar component that will be seamlessly integrated across all front-end pages within the system. This side-bar should be functionally robust, ensuring users have easy access to key sections or features from any page they navigate. Additionally, it is imperative that this component remains uniform in its presentation and functionality, providing users with a predictable and user-friendly interface. The component must also be responsive, catering to various screen dimensions to maintain its functional integrity and user experience.	1	1	**User Story5.1**	1. The sidebar must be visible on all pages of the system. 2. Must contain links to key sections or features. 3. Must have a consistent design across all pages. 4. Must be responsive to different screen sizes. 5. Must be functionally robust and fault-tolerant.	1. Navigate to various pages, and confirm the sidebar is present and identical on each. 2. Click on each link in the sidebar and confirm it redirects to the correct section or feature. 3. Resize the browser window and observe the sidebar's responsive behavior. 4. Simulate slow network conditions to ensure the sidebar still functions correctly. 5. Validate HTML/CSS/JS through linting and check for accessibility compliance.

Frontend 1.2	**Implement a Header Component**	Develop a header component to be consistently integrated across all front-end pages of the system. This footer should support the inclusion of image and video file types.	1	1	**User Story 5.1**	1. Header should appear at the top of all pages. 2. Header should support image and video file types. 3. Header should maintain a consistent look and feel across all pages.	1. Confirm that the header is present on all pages and retains a consistent design. 2. Upload image and video files to the header and confirm they are displayed correctly. 3. Validate HTML /CSS/JS through linting and check for accessibility compliance.
Frontend 1.3	**Implement a Footer Component**	Develop a footer component to be consistently integrated across all front-end pages of the system. This footer should support the inclusion of image and video file types.	1	1	**User Story 5.1**	1. Footer should be visible at the bottom of all pages. 2. Footer should support image and video files. 3. Footer should maintain a consistent look across all pages.	1. Confirm that the footer is visible on all pages and retains a consistent design. 2. Upload image and video files to the footer and confirm they display correctly. 3. Validate HTML /CSS/JS through linting and check for accessibility compliance.
Frontend 2.1.1	**Create "Add New Module" Page**	Create a user-friendly "Add New Module" page that displays a pop-up box for module creation. Inside the pop-up, incorporate a dropdown menu for selecting desired file types. Implement the functionality to allow users to select file types from the dropdown.	3	3	**User Story 2.1**	1. A pop-up box must appear when the user chooses to add a new module. 2. The pop-up box must contain a dropdown menu for file type selection. 3. Dropdown menu must enable selection of different file types.	1. Click the "Add New Module" button and confirm a pop-up box appears. 2. Validate that the dropdown menu is present within the pop-up box. 3. Select various file types from the dropdown and confirm that the selection is accepted.

Frontend 2.1.2	**Create a Pop-Up Window for "Add New Module"**	Integrate a pop-up window for adding materials to a module. This pop-up window should feature a dropdown menu for selecting desired file types, and users should be able to choose file types from the dropdown.	3	3	**User Story 2.1**	1. A pop-up window must appear for adding materials to a module. 2. Must include a dropdown menu for selecting file types. 3. Must enable file type selection from the dropdown.	1. Click on an existing module to add materials and confirm that a pop-up window appears. 2. Validate that the dropdown menu is present within the pop-up window. 3. Select various file types from the dropdown and confirm the selections are accepted.
Frontend 2.2.1	**Create Assessment Preview Component**	Design an MCQ (Multiple Choice Question) template component that can be reused for various MCQs within the system. Ensure its compatibility with different assessments.	3	3	**User Story 6.2**	1. The component should only support MCQ type templates. 2. The component should be reusable across different MCQs in the system.	1. Add the component to a page and ensure only MCQs are displayed. 2. Reuse the component on a different page with different MCQs and ensure it functions as expected.
Frontend 2.2.2	**Create File Material Preview Component**	Develop a component that allows users to preview file materials within the system. This component should support only image and video file types, offering clear and optimal viewing capabilities for both. It should handle different image formats and video playback functionalities, ensuring seamless user experience when accessing these materials.	3	1	**User Story 3.3**	1. The component should support image and video file previews. 2. It should provide clear viewing capabilities for both image and video files. 3. Video playback functionalities should be smooth and seamless.	1. Upload and preview an image file using the component. 2. Upload and preview a video file using the component, testing playback functionalities. 3. Check with different image and video file formats.

Frontend 2.2.3	**Create Text Material Preview Component**	Develop a component that facilitates the preview of text materials within the system. The component should render text content in a readable and organized manner, ensuring users can comfortably view and interact with the provided textual information. Additionally, it should support various text formatting features to maintain content integrity during the preview.	3	1	**User Story 3.3**	1. The component should render text clearly and legibly. 2. It should support various text formatting features.	1. Preview text with various formatting using the component and ensure it's displayed correctly. 2. Test bold, italics, underlined, and other formatted text for consistent rendering.
Frontend 2.2.4	**Create Module Preview Page**	- Design a page to list all modules with a brief description. - Include options to view, edit, or delete the modules.	3	3	**User Story 2.2**	1. The page should list all modules with a brief description. 2. Options to view, edit, or delete the modules should be visible and accessible.	1. Navigate to the module preview page and ensure all modules are listed. 2. Attempt to view, edit, and delete a module to ensure functionalities work.
~~**Frontend 2.2.5**~~	~~**Enhance Update Function for Sub-Module Movement**~~	~~- Add drag-and-drop functionality to rearrange the order of sub-modules.~~ ~~- Save the new arrangement automatically or via a "Save Changes" button.~~	~~5~~	~~1~~	~~**User Story 2.3**~~	1. ~~Users should be able to drag-and-drop to rearrange the order of sub-modules.~~ 2. ~~The new arrangement should be saved either automatically or via a "Save Changes" button.~~	1. ~~Drag a sub-module to a new position and ensure its position changes.~~ 2. ~~Refresh or navigate away from the page and come back to confirm the new order is preserved.~~
~~**Frontend 2.2.6**~~	~~**Adjust the Order of Materials in Modules**~~	~~- Add an option to move training materials up or down in the list.~~ ~~- Ensure that the new order is saved and reflected immediately.~~	~~5~~	~~1~~	~~**User Story 2.3**~~	1. ~~Users should be able to move training materials up or down in the list.~~ 2. ~~The new order should be immediately saved and reflected.~~	1. ~~Move a training material up or down the list.~~ 2. ~~Refresh the page to check if the new order is preserved.~~

Frontend 2.2.7	**Add new texture material into existing module**	Develop a feature to allow the addition of new "texture" materials into existing modules.	5	1	**Maps to User Story3.1**	1. Users must be able to add new texture materials to existing modules. 2. The texture materials should be viewable within the module upon addition. 3. Supported texture formats should be clearly indicated.	1. Open an existing module and add a new texture material; confirm it is added correctly. 2. Preview the added texture within the module. 3. Attempt to add unsupported formats and confirm that the system prevents you from doing so.
Frontend 2.2.8	**Add new file material into existing module**	Develop a feature to allow users to add new file materials into existing modules.	5	1	**Maps to User Story3.1**	1. Users must be able to add new file materials into existing modules. 2. Supported file formats should be clearly listed. 3. Users should be able to view the file within the module upon addition.	1. Open an existing module and add a new file material; ensure it gets added. 2. Preview the added file within the module. 3. Attempt to upload an unsupported file format and confirm that it's not allowed.
Frontend 2.2.9	**Add new assessment into existing module**	Create a feature to allow the addition of new assessments (MCQs, true/false questions, etc.) into existing modules.	5	1	**Maps to User Story6.1**	1. A “New Assessment” button should be available in the existing module. 2. Users must be able to add questions and answer options. 3. The newly added assessment should be viewable within the module.	1. Add a new assessment to an existing module and ensure it gets added. 2. Preview the added assessment to confirm it's functional. 3. Check that questions and answer options are displayed correctly.

Frontend 2.2.10	**Delete texture material into existing module**	Implement a feature to allow users to delete texture materials from existing modules.	5	1	**Maps to User Story3.5**	1. A delete option must be available for each texture material in a module. 2. The texture material should be removed from the module upon deletion.	1. Delete a texture material from a module and confirm it gets removed. 2. Refresh the module to ensure the texture material remains deleted.
Frontend 2.2.11	**Delete file material into existing module**	Implement a feature to allow the deletion of file materials from existing modules.	5	1	**Maps to User Story3.5**	1. A delete option should be available for each file material in a module. 2. The file should be removed from the module upon deletion.	1. Delete a file material from a module; confirm it is removed. 2. Refresh the module to ensure the file material stays deleted.
Frontend 2.2.12	**Delete assessment into existing module**	Implement a feature to allow the deletion of assessments from existing modules.	5	1	**Maps to User Story6.4**	1. A delete option must be visible for each assessment within a module. 2. The assessment should be removed from the module upon deletion.	1. Delete an assessment from a module; confirm it gets removed. 2. Refresh the module to make sure the assessment remains deleted.
Frontend 2.3.2	**Create Search Result Display Area Component**	Create a search result display area that showcases content from specific modules based on keywords.	1	1	**User Story 5.2**	1. The component should display content from modules based on keywords. 2. The display should be clear and organized.	1. Enter a keyword into the search and ensure relevant module content appears in the component. 2. Validate the clarity and organization of the displayed results.

Frontend 3.1	**Create 'Admin Profile' Page**	Develop a responsive 'Admin Profile' page for administrators to view and manage their details. Features should include viewing basic details.	3	1	**User Story 5.3**	1. The page should be responsive across devices. 2. Admins should be able to view their basic details.	1. Navigate to the 'Admin Profile' page and ensure basic details are visible. 2. View the page on different devices or screen sizes to test its responsiveness.
Frontend 3.2	**Create 'Edit Admin Profile' Page**	- Design a form that allows editing the admin’s personal information. - Include fields for name, email, password, and profile picture.	3	1	**User Story 5.3**	1. The page should present a form with fields for name, email, password, and profile picture. 2. All edits should be savable and reflected immediately upon saving.	1. Navigate to the 'Edit Admin Profile' page and ensure all fields are present. 2. Edit the fields and save changes, then verify the edits have been implemented.
Frontend 3.3	**Create ‘Admin Login’ Page**	- Implement a secure login page using secure practices. - Include fields for username and password and a “Remember Me” checkbox.	3	1	**User Story 4.4**	1. The page should have fields for username and password. 2. A "Remember Me" checkbox should be present. 3. The login process should be secure.	1. Navigate to the ‘Admin Login’ Page and ensure all fields and checkboxes are present. 2. Attempt to log in with correct and incorrect credentials to verify security.
Frontend 4.1	**Firebase File Upload in React**	- Integrate Firebase storage to enable file uploads. - Ensure that uploaded files are saved to Firebase and their URLs are stored in the database.	5	1	**User Story 3.1**	1. The system should be able to upload files to Firebase storage. 2. The URLs of uploaded files should be saved in the database.	1. Upload a file and verify that it's stored in Firebase. 2. Check the database to ensure the file's URL is saved.

Fronten d_4.1.2	**Firebase Delete File Function**	Integrate Firebase functionality to allow the deletion of files from the database.	5	1	**Maps to User Story3.5**	1. Files should be removed from the Firebase database upon deletion from the module. 2. The deletion should be secure and authenticated.	1. Delete a file from a module and check Firebase to confirm that it's removed. 2. Attempt to delete a file without proper authentication to ensure that it fails.
~~**Fronten d_5.1.1**~~	~~**Able to display a list of age groups**~~	~~Implement a frontend feature where admins can view a comprehensive list of age groups available on the platform. The list should be presented in a readable format, preferably in a table or list structure, allowing easy scanning and selection.~~	~~5~~	~~1~~	~~**User Story 2.2**~~	~~1. Admin should see a complete, updated list of age groups.~~ ~~2. The list should be organized in an ascending or logical order.~~ ~~3. Each entry should be selectable to view more details or perform actions like edit or delete.~~	~~1. Navigate to the age group page.~~ ~~2. Confirm all age groups in the database are displayed.~~ ~~3. Ensure no duplicate or missing entries.~~ ~~4. Select an age group to view its details.~~
~~**Fronten d_5.1.2**~~	~~**Pop-up windows for update name of age group**~~	~~Design and implement a pop-up window that prompts the admin when they wish to update the name of an existing age group. The window should provide an input field to make the desired changes and confirm/save button.~~	~~5~~	~~1~~	~~**User Story 2.3**~~	~~1. On selecting to edit an age group, a pop-up should appear.~~ ~~2. Current age group name should be displayed in an editable field.~~ ~~3. Changes should be saved only upon confirmation.~~	~~1. Click on the 'edit' option for an age group.~~ ~~2. Verify that the current name displays correctly in the pop-up.~~ ~~3. Change the name and click the save or confirm button.~~ ~~4. Ensure the list updates with the new name and the old name no longer appears.~~
Fronten d_5.1.3	**Pop-up windows for create a new empty age group**	Design and implement a pop-up window that allows the admin to create a new age group. The window should have a relevant input field to enter the age group's name and a button to confirm the creation	5	1	**User Story 2.1**	1. An option /button to add a new age group should be visible. 2. On selecting to add, a pop-up should appear with a clear indication of its purpose. 3. There should be an input field to type in the new age group's name and a button to confirm.	1. Click on the 'add' option for age groups. 2. Confirm the pop-up displays correctly with the necessary fields. 3. Input a new age group name and confirm. 4. Ensure the new age group appears in the list

Fronten d_5.2.1	**Single AgeGroup Detail Page - Modules Table**	Implement a table displaying a list of related modules.	5	1	**User Story 2.2**	1. Detailed page for an age group should list all associated modules. 2. The table should have relevant headers like 'Module Name', 'Description', 'Date Added', etc. 3. Modules should be clickable to view more details.	1. Navigate to an age group's detailed page. 2. Confirm the presence of a module table. 3. Verify the table headers against expected headers. 4. Click on a module to ensure it leads to detailed module information.
Fronten d_5.2.2	**Single AgeGroup Detail Page - Coaches Table**	Implement a table displaying a list of assigned coaches.	5	1	**User Story 2.2**	1. Detailed page for an age group should list all coaches associated with it. 2. The table should have relevant headers like 'Coach Name', 'Contact Info', 'Assigned Since', etc. 3. Coaches should be selectable to view detailed profiles or for further actions like editing.	1. Navigate to an age group's detailed page. 2. Verify the presence of a coaches table. 3. Confirm the table headers match expected headers. 4. Select a coach to view their detailed profile.
Fronten d_5.2.3	**Create an overview information display block for Single AgeGroup Detail Page**	Develop a dedicated section on the age group detail page that provides an overview or summary of the age group. This could include total modules, total coaches, average student performance.	5	1	**User Story 2.2**	1. An overview section should be present at the top or a prominent position on the age group detailed page. 2. The overview should provide summarized details like total number of modules, total number of coaches, etc. 3. Data should be up-to-date and accurately reflect the database's content.	1. Navigate to an age group's detailed page. 2. Check for the presence of an overview section. 3. Verify the data displayed matches the expected total counts from the database or other sections.
Fronten d_6.1.1	**Coaches Display Page - Search Bar**	Implement a search bar to search for a specific coach.	1	1	**User Story 5.2**	1. A search bar should be visible at a prominent position on the coaches display page. 2. Typing into the search bar should provide real-time results or suggestions. 3. Entering a full name or part of it should return relevant results.	1. Navigate to the coaches display page. 2. Confirm the search bar's presence and enter a known coach name. 3. Check if relevant results or suggestions appear as you type. 4. Finalize the search and verify the displayed results.

Frontend_6.1.2	**Coaches display page- Table to display all coaches**	Create a comprehensive table on the coaches' display page that lists all coaches in the system. Each row should represent a coach, showing pertinent details and possibly options to view more, edit, or delete.	1	1	**User Story 4.1**	1. The coaches page should showcase a table listing all coaches. 2. Relevant details like name, contact info, and assigned age group should be displayed. 3. The table should allow sorting or filtering based on various parameters.	1. Visit the coaches display page. 2. Ensure all coaches from the database are listed. 3. Verify the presence of relevant columns in the table. 4. Try using sorting or filtering features to check their functionality.
Frontend_6.1.3	**Coaches display page- Pop-up windows for create a new coach**	Design and develop a pop-up window that appears when an admin wishes to add a new coach. The window should contain necessary input fields for coach details and a confirmation button.	1	1	**User Story 4.1**	1. An option to add a new coach should be visible on the coaches page. 2. Clicking this option should open a pop-up with input fields for coach details. 3. On saving, the new coach should be added to the database and reflected in the list.	1. On the coaches display page, click on the 'add new' option. 2. Check if the pop-up appears with necessary input fields. 3. Fill in the details and save. Navigate back to the coaches list and verify if the new coach is displayed.
Frontend_6.2.1	**Display Coach profile**	Design a detailed coach profile page that displays all relevant information about a coach, including their coaching category, assigned age groups, and contact details.	1	1	**User Story 4.3**	1. On selecting a specific coach, a detailed profile page should be displayed. 2. The profile should display all relevant information about the coach, including their coaching category, contact details, and associated age groups. 3. The information should be organized in a clear and readable format.	1. Navigate to the coaches list and select a specific coach. 2. Verify that the detailed profile page displays. 3. Cross-check all displayed details against the database or known data to ensure accuracy.
Frontend_6.2.2	**Edit Coach profile**	Single coach profile page	5	1	**User Story 4.3**	1. There should be an option to edit the details on the coach's profile page. 2. On selection, input fields should become editable, allowing modifications. 3. Changes should be saved only upon confirmation, and the updated details should reflect immediately on the profile and wherever the coach details are displayed	1. Navigate to a specific coach's profile. 2. Click the 'edit' button or equivalent. 3. Modify some details and save. Confirm that the updates reflect immediately on the profile and other relevant areas where the coach's data appears.

Frontend_6.3	**Achieve redirections between AgeGroup detail page with Coach profile page**	Ensure seamless navigation between the age group detail page and the coach profile page, possibly through clickable names or dedicated buttons. This should offer the admin a fluid experience while navigating.	5	1	**User Story 5.1**	1. From an age group detail page, coaches listed should be clickable. 2. On clicking a coach, the system should redirect to that specific coach's detailed profile page. 3. Navigational flow should be intuitive and seamless without unnecessary redirections or errors.	1. Go to the detail page of an age group. 2. Click on a listed coach's name or profile link. 3. Confirm that you're correctly redirected to the chosen coach's detailed profile page.
Backend 1.1	**Create Operations for the Module**	- Develop RESTful API endpoints to create a new module in the system. - Ensure the operations validate incoming data before processing.	3	1	**User Story 2.1**	1. API should be RESTful. 2. The operations should validate incoming data before processing.	1. Call the API with valid and invalid data to test validation. 2. Verify that a new module is created in the system when valid data is used.
Backend 1.2	**Delete Operations for the Module**	- Implement API endpoints to delete existing modules based on their unique IDs. - Add security measures to prevent unauthorized deletions.	3	1	**User Story 2.4**	1. The API should delete a module based on its unique ID. 2. Security measures should prevent unauthorized deletions.	1. Attempt to delete a module with and without authorization to test security measures. 2. Confirm that the module is removed from the system database when properly authorized.
Backend 1.3	**Update Operations for the Module**	- Create API endpoints to update existing modules, validating incoming data. - Include the option to update name, description, and associated class details.	3	1	**User Story 2.3**	1. The API should allow updating the name, description, and associated class details. 2. It should validate incoming data.	1. Update a module's details with valid and invalid data to test validation. 2. Confirm that the updates are reflected in the database.
Backend 1.4	**Retrieve Operations for the Module**	- Develop API endpoints for retrieving module information based on their unique IDs.	3	1	**User Stories 2.1-2.4**	1. The API should retrieve module information based on its unique ID.	1. Query a module by its ID and verify that the returned data matches what is stored in the database.

Backend 1.6.	**Find Children Operations for the Module**	- Implement an API to find all sub-modules and materials under a parent module.	3	1	**User Story 5.2**	1. The API should find and return all sub-modules and materials under a parent module.	1. Add sub-modules and materials to a parent module and then use the API to confirm they are returned correctly.
Backend 1.7	**Get All Materials of Module by ID**	- Develop an API to retrieve all materials associated with a module based on its unique ID.	3	1	**User Story 5.2**	1. The API should retrieve all materials based on the module's unique ID.	1. Query the API with a module's unique ID and confirm that all associated materials are returned.
Backend 1.8	**Delete All Related Materials When Deleting Module**	- Modify the deletion API to also remove all associated materials and sub-modules when a module is deleted.	5	1	**User Story 2.4**	1. Deleting a module should also remove all associated materials and sub-modules.	1. Delete a module and then confirm that all related materials and sub-modules are also deleted.
Backend 2.1	**Update Admin Information**	- Create an API for updating admin profile information.	3	1	**User Story 5.3**	1. The API should allow updating the admin profile informatio n.	1. Use the API to update the admin's informatio n and check that the changes are reflected in the database.
Backend 2.2	**Retrieve Admin Information**	- Develop an API for retrieving the stored profile information of the admin.	3	1	**User Story 5.3**	1. The API should retrieve the stored profile informatio n of the admin.	1. Use the API to fetch the admin profile and verify that the returned data matches the database.
Backend 2.3	**Create Admin Information**	- Implement an API for storing admin profile information during registration.	1	1	**User Story 5.3**	1. The API should store the admin's profile informatio n during registratio n.	1. Register a new admin and confirm that the informatio n is properly stored in the database.

Backend 2.4	**Delete Admin Information**	- Create an API for deleting an admin profile, with added security measures.	1	1	**User Story 5.3**	1. The API should delete an admin profile. 2. Extra security measures should be in place.	1. Attempt to delete an admin profile without proper authorization to test security. 2. Delete an admin profile with proper authorization and verify that it is removed from the database.
Backend 3.1	**Update Coach Information**	- Develop an API for updating the profiles of coaches.	1	1	**User Story 4.2**	1. The API should allow for updating coach profiles.	1. Update a coach's profile and confirm that the changes are saved in the database.
Backend 3.2	**Create Coach Information**	- Create an API for registering new coaches.	1	1	**User Story 4.2**	1. The API should enable the registration of new coaches.	1. Register a new coach and confirm that the details are saved in the database.
Backend 3.3	**Delete Coach Information**	- Implement an API to remove coach profiles.	1	1	**User Story 4.2**	1. The API should allow for the removal of coach profiles.	1. Delete a coach profile and confirm that it is removed from the database.
Backend 3.4	**Retrieve Coach Information**	- Develop an API for fetching coach profiles based on various criteria.	1	1	**User Story 4.2**	1. The API should fetch coach profiles based on various criteria.	1. Query the API using different parameters to fetch coach profiles and validate that the returned data is accurate.
Backend 4.1	**Create Operations for the Class**	- Create an API for adding new classes.	1	1	**User Story 1.1**	1. The API should allow for adding new classes.	1. Add a new class and confirm that it's saved in the database.
Backend 4.2	**Delete Operations for the Class**	- Develop an API to delete existing classes.	1	1	**User Story 1.4**	1. The API should allow for deleting existing classes.	1. Delete an existing class and verify that it is removed from the database.

Backend 4.3.	**Update Operations for the Class**	- Implement an API to update class information.	1	1	**User Story 1.3**	1. The API should allow for updating class information.	1. Update a class's information and confirm that the changes are reflected in the database.
Backend 4.4	**Find Operations for the Class**	- Create an API to query classes based on different parameters.	5	1	**User Story 5.2**	1. The API should allow for querying classes based on different parameters.	1. Query the API with different parameters and validate that the returned class data is accurate.
Backend 5	**Admin Login**	- Develop secure login functionality with features like rate-limiting and account locking after multiple failed attempts.	3	1	**User Story 4.4**	1. The login functionality should be secure, with features like rate-limiting and account locking.	1. Attempt multiple failed logins to test rate-limiting and account locking.
Backend 5.1.	**Create Operations for the Materials**	- Implement an API for adding new materials.	1	1	**User Story 3.1**	1. The API should allow for adding new materials.	1. Add new materials and confirm that they are saved in the database.
Backend 5.2	**Update Operations for the Materials**	- Develop an API for updating existing materials.	1	1	**User Story 3.4**	1. The API should allow for updating existing materials.	1. Update a material and confirm that the changes are reflected in the database.
~~**Backend 5.3**~~	~~**Find Operations for the Materials**~~	~~- Create an API to search for materials based on various criteria.~~	~~5~~	~~1~~	~~**User Story 5.2**~~	1. ~~The API should allow for searching materials based on various criteria.~~	1. ~~Query the API using different parameters to fetch materials and validate that the returned data is accurate.~~
Backend 5.4	**Remove Materials When Deleting**	- Modify the deletion API to also remove associated materials when a file or assessment is deleted.	5	1	**User Story 3.5**	1. The API should remove associated materials when a file or assessment is deleted.	1. Delete a file or assessment and confirm that associated materials are also removed from the database.

~~**Backend 5.5**~~	~~**Implement Login Attempt Count**~~	~~- Add a counter in the backend to track failed login attempts for admin login.~~	~~5~~	~~1~~	~~**User Story 4.4**~~	1. ~~The backend should track failed login attempts for the admin login.~~	1. ~~Make a series of failed login attempts and check if they are tracked.~~
~~**Backend 5.6**~~	~~**Implement Login Whitelist Filtering**~~	~~- Add a whitelist of IP addresses that are allowed to access the admin login page.~~	~~5~~	~~1~~	~~**User Story 4.4**~~	1. ~~The backend should have a whitelist of IP addresses allowed to access the admin login page.~~	1. ~~Attempt to access the admin login page from an IP address not in the whitelist to verify that access is denied.~~
~~**Backend 5.7**~~	~~**Store Login Tokens in the Database**~~	~~- Implement secure token storage in the database to keep track of logged-in sessions.~~	~~5~~	~~1~~	~~**User Story 4.4**~~	1. ~~The backend should securely store login tokens in the database.~~	1. ~~Log in as an admin and check if the login token is securely stored in the database.~~
Backend 6	**Admin Logout**	- Develop a secure logout functionality that invalidates the session.	5	1	**User Story 4.4**	1. The backend should invalidate the session upon logout.	1. Logout from an admin account and confirm that the session is invalidated.