FusionIQ Documentation: Mentor Perspective

Step 1: Login Session and Navigation to Mentor Main Page

1. **Login to FusionIQ Platform**:

- **Access the Website**: Start by navigating to the official FusionIQ website via your web browser.
- **Enter Login Credentials**: On the main login page, you will see two fields: one for your **username** or **email address** and one for your **password**. Enter your credentials carefully.
- **Submit Login**: Once the credentials are entered, click on the **Login** button (or press enter). The system will verify the provided information.
- **Successful Login**: If the credentials are correct, you will be redirected to the **Home Page** of FusionIQ.

- In case of incorrect credentials, an error message will appear prompting you to re-enter your login details.

2. **Navigating from Home Page to Mentor Selection:**

- **Home Page Overview**: After logging in, you will arrive at the **Home Page**. The Home Page contains various navigation options, icons, and the main dashboard relevant to your activities.
- **Education Icon in Header**: In the header section (top of the page), you will see an icon representing **Education**. This icon might resemble a graduation cap or book, depending on the design.
- **Accessing the Person Icon**: Once you click on the Education icon, another icon, referred to as the **Person Icon** (or profile icon), will appear near the same header. This icon usually represents a user or profile settings.
- **Opening the Dropdown Menu**:

    - Click on the Person Icon to reveal a **dropdown menu**.
    - In this dropdown menu, you will see two distinct options:
        - **As User**
        - **As Mentor**
    - These options represent the different roles or perspectives available within the platform. The User role allows basic access to the platform, while the Mentor role provides access to specific tools and functionalities designed for mentors.

3. **Choosing the Mentor Role and Accessing Sub-Login:**

- **Selecting "As Mentor"**: From the dropdown options, click on **"As Mentor"**. This selection indicates that you want to access the platform from the mentor's perspective.
- **Redirect to Mentor Sub-Login Page**: After selecting the Mentor option, the system will redirect you to a **Mentor Sub-Login Page**.

- This page is designed specifically for users who are mentors on the platform. It requires a secondary set of login credentials.

4. **Entering Mentor Sub-Login Credentials:**

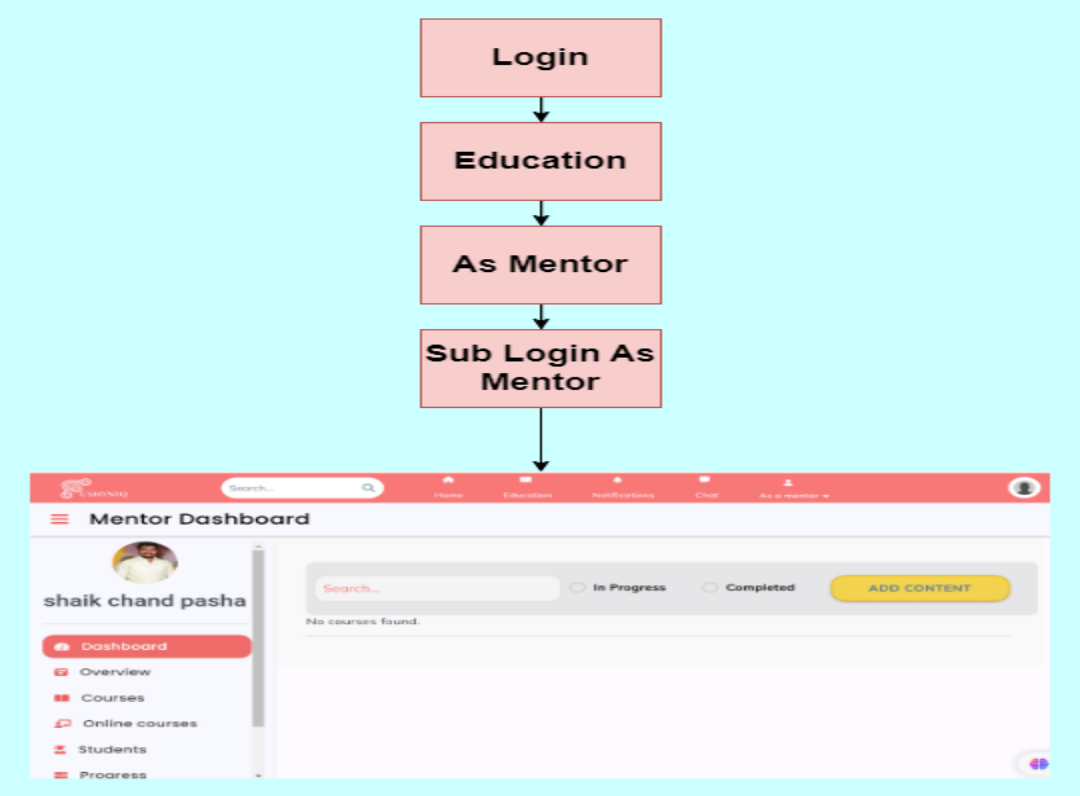
- **Mentor-Specific Login**: On the Mentor Sub-Login Page, you will need to enter your **mentor credentials**. These credentials are different from the ones used to log into the platform as a user.

- **Credential Fields**:

  - **Username/Email**: Enter the unique username or email associated with your mentor account.
  - **Password**: Enter your mentor account password.

- **Submit Mentor Login**: After entering the correct credentials, click the **Login** button to proceed.
- **Successful Mentor Login**: Once the credentials are verified, you will be **redirected to the Mentor Main Page**, where you can access all the tools, resources, and features specific to the Mentor role.

  - In case of invalid credentials, an error message will appear, prompting you to re-enter the correct information.



Step 2: Mentor Dashboard :

After successfully logging in as a mentor, you will be redirected to the **Mentor Dashboard**. The dashboard is divided into two main sections:

- **Left Side Menu**: The left side of the dashboard features a menu that contains various tabs to help navigate through the mentor functionalities.

Tabs available in the Mentor Dashboard menu:

1. **Dashboard**
2. **Overview**
3. **Courses**
4. **Online Courses**
5. **Students**
6. **Progress**
7. **Analytics**
8. **Mock**

This concludes the introduction of the tabs available in the Mentor Dashboard. In the following steps, we'll dive into the specific functionalities of each tab.

Step 3: Dashboard Tab:

Once you click on the **Dashboard** tab from the left-side menu, the right side of the screen displays the main dashboard content, which consists of:

- Search Bar: Located at the top, allowing mentors to search for specific courses or content.
- Radio Buttons:
  - In Progress: Displays the courses that are currently being developed or are ongoing.
  - Completed: Displays the courses that have been fully developed or completed.

Add Content Button: Below the search bar and radio buttons, you will see a button labeled **Add Content**. Clicking this button opens a pop-up overlay with three options displayed as cards. These cards provide different actions mentors can take.

Step 3.1: Add Content Options

Upon clicking the **Add Content** button, a pop-up overlay is displayed with the following three options:

- **Add Course**
- **Schedule A Class**
- **Add Mock Interview**

Step 3.1.1: Add Course

When selecting the **Add Course** option, another pop-up overlay appears. Before proceeding to create a course, the mentor must agree to the platform's **Privacy Policy**.

1. **Privacy Policy Acceptance**:
    - The pop-up presents a detailed **Privacy Policy** that must be read before creating a course.
    - After reading, there are two options:
        - **Checkbox**: If the mentor agrees to the privacy policies, they can check the box.
        - **Decline Button**: A **red "Decline" button** is also available if the mentor does not agree with the policies.
2. **Green Accept Button**:
    - Once the mentor agrees to the privacy policy by checking the box, the **green "Accept" button** is enabled.
    - Clicking the **Accept** button proceeds to the next step of course creation.

Step 3.1.1.2: Course Title and Creation

After accepting the privacy policy, another pop-up appears, prompting the mentor to provide essential course details:

1. **Course Title Input**: A placeholder input field appears where the mentor can enter the **course title**.
2. **Buttons**:
    - **Create Course Button**: Once the title is entered, the mentor clicks the **Create Course** button to finalize the process.
    - **Close Button**: If the mentor decides not to proceed, they can click the **Close** button to exit the course creation process.

Step 3.1.1.3: Course Creation Success

Upon successful creation of the course:

1. An **alert message** appears confirming, "Course Created Successfully."

1. The mentor is then **redirected to the Course Planning section** where further course details and plans are outlined.

Step 3.1.2: Course Planning Section Overview

After creating the course, you are redirected to the **Course Planning Page**, which contains five essential tabs for course management:

1. **Course Landing**
2. **Course Trailer**
3. **Course Planning**
4. **Curriculum**
5. **Course Project**
6. **Pricing & Promotions**

Step 3.1.2.1: Course Landing Tab

In the **Course Landing** tab, the mentor provides essential details that form the foundation of the course structure and presentation. This section includes 7 input fields that guide the mentor in specifying the course's key attributes.

1. **Course Type**:
    - The mentor can select the level of the course based on the intended audience. There are three options:
        - **Beginner Level**
        - **Medium Level**
        - **Advanced Level**
    - Based on the selection, the mentor defines the standard or complexity of the course content.
2. **Course Mode**:
    - The mentor selects the mode of delivery for the course, choosing between:
        - **Online**
        - **Offline**
    - This choice specifies whether the course will be delivered digitally (online) or in a traditional classroom setting (offline).
    - **Conditional Input**: If the mentor selects **Offline**, an additional input field for **Course Term** will appear.
3. **Course Term** (Appears only if Course Mode is set to Offline):
    - The **Course Term** allows the mentor to define the course duration and format. There are two types of courses:
        - **Long Course**:
            - This is a comprehensive course with **multiple modules**. Each module contains several lessons, and each lesson includes related videos.
        - **Short Course**:
            - This course format skips modules and directly presents **lessons**. Each lesson contains related videos, but the overall duration is shorter, allowing learners to complete it quickly.

4. **Course Objective**:
    - In this field, the mentor provides a detailed **description of the course objectives**. This section outlines what the course aims to achieve and what students will gain by completing it.
5. **Course Image**:
    - The mentor can **upload an image** that represents the course. This image will be displayed on the **enrollment page**, so it should be relevant to the course content.
6. **Why Enroll in this Course**:
    - The mentor must explain the **importance of the course** by providing **at least five points**. This section highlights the unique value of the course and motivates potential students to enroll.
7. **Duration (in weeks)**:
    - The mentor estimates the **total duration** of the course and enters the approximate time it will take students to complete the course, measured in **weeks**.
8. **Language**:
    - The mentor selects the **language** in which the course will be delivered. The selected language will determine the language used for course materials and instructions.

Step 3. 1.2.2: Course Trailer Tab

In the **Course Trailer** tab, the mentor can add promotional materials for the course, such as video trailers and course-related documents. This section is divided into two primary areas: **Course Trailer** and **Course Document**. Below is a detailed breakdown of the functionalities within each area.

Course Trailer Section

1. **Video Trailers**:
    - The mentor can upload **multiple video trailers**, each of which includes a title and a description to highlight different aspects of the course.
    - **Dynamic Video Uploads**: The form dynamically adjusts based on the number of trailers added, allowing mentors to add as many videos as needed.
2. **Upload Functionality**:
    - **File Input**: A file input allows mentors to upload **video files** as trailers.
    - **Video Title Input**: A text input field is used to enter a **title** for each trailer.
    - **Video Description Dropdown**: A dropdown menu is provided to select the **description level** for each trailer, with the following options:
        - **Beginner**
        - **Intermediate**
        - **Advanced**

3. **Existing Trailers**:
    - A list of already uploaded trailers is displayed using the `videoTrailers` array.
    - Each trailer in the list shows its title and description, along with an option to **delete** the video if necessary.

Course Document Section

1. **Upload Course Materials**:
    - Mentors can upload **course documents** such as PDFs, Word documents, and other materials that are relevant to the course.
    - **File Input**: Provides a way to select and upload multiple files.
2. **File Management**:
    - Before uploading, mentors can see the list of **selected files**. This allows them to review and, if necessary, remove any unwanted files before they are uploaded.
    - After the documents are uploaded, a list of **uploaded documents** is displayed, each with a **link** for viewing or downloading.
3. **Document Deletion**:
    - Each document in the list has a **delete option**, enabling mentors to remove any uploaded documents they no longer want associated with the course.

Save Button

- At the bottom of the form, a **Save** button is provided. This button saves all the video trailers and documents uploaded by the mentor.
- **Functionality**: Upon clicking the Save button, the entire form is saved, which may also **trigger navigation** to the next section of the course creation process.

Step 3. 1.2.3: Course Planning Tab

In the **Course Planning** tab, the mentor can define key components that shape the structure of the course, including the tools, skills, and prerequisites. This section is divided into three main areas: **Tools**, **Skills**, and **Prerequisites**. Each section allows mentors to dynamically add multiple entries for better course customization.

1. Tools Section

This section allows the mentor to specify the tools that will be used or taught in the course.

- **Tool Name Input**:
    - A required input field where the mentor can enter the **name of the tool**.
    - This field helps to define the tools associated with the course content.
- **Tool Image Upload**:

    - A **file input** where the mentor can upload an image representing the tool.
    - Once the image is uploaded, a **preview** is displayed so the mentor can confirm the correct image was selected.
- **Add Tool Button**:
    - A button labeled **Add Tool** is available to **dynamically add** more tool input sections.
    - The mentor can add as many tools as needed by clicking this button, expanding the list of tools for the course.

2. Skills Section

This section allows the mentor to define the skills that students will gain from the course.

- **Skill Name Input**:
    - A required input field where the mentor can specify the **name of the skill** students will learn.
- **Skill Image Upload**:
    - A **file input** to upload an image that represents the skill.
    - Similar to the tools section, a **preview** of the uploaded image is displayed to confirm the correct image selection.
- **Add Skill Button**:
    - A button labeled **Add Skill** allows mentors to **dynamically add** more skill input sections.
    - Mentors can add multiple skills by clicking this button, ensuring a comprehensive list of course-related skills.

3. Prerequisites Section

This section enables mentors to specify any prerequisites that students must meet before enrolling in the course.

- **Course Prerequisite Input**:
    - A required input field where the mentor can enter a **prerequisite** for the course.
    - This field helps ensure that students meet the necessary background or knowledge required to succeed in the course.
- **Add Prerequisite Button**:
    - A button labeled **Add Prerequisite** is provided to add up to **two prerequisite inputs**.
    - There is a limit of **two prerequisites**, allowing mentors to specify critical requirements without overwhelming potential students.

4. Save Button

- A **Save** button is provided at the end of the form to **submit** all the inputs, including tools, skills, and prerequisites.
- Clicking the **Save** button triggers form submission, ensuring that all the sections are saved together for the course.

Step 3. 1.2.4: Course Project Tab

The **Course Project** tab allows the mentor to define a project that students will work on as part of the course. This section includes four main input fields for creating a project, along with functionality to manage project documents and save the project.

1. Project Title

- **Project Title Input**:
  - A required input field where the mentor can enter the **title of the project**.
  - The project title serves as the main identifier for the project in the course.

2. Project Description

- **Project Description Input**:
  - A required field where the mentor provides a **detailed description** of the project.
  - This description explains the project goals, tasks, and expectations for students.

3. Deadline

- **Project Deadline Input**:
  - A field to set the **deadline** for the project.
  - The mentor can specify the exact date by which students need to complete the project.

4. Project Document

- **Document Upload**:
  - A file input to upload **documents** related to the project. Supported formats include **PDF**, **DOC**, and **DOCX**.
  - This allows the mentor to provide additional resources, guidelines, or templates for the project.

Document Preview and Download

- **Preview the Document**:
  - Once a document is uploaded, the system allows the mentor to either **view** or **download** the document by clicking the respective buttons (**View Document** or **Download Document**).
  - This ensures that the correct document was uploaded and is accessible for review or download.

Save the Project

- **Save Project Button**:
  - After filling in all the required fields, the mentor can click the **Save Project** button to submit the project details.
  - The button is only enabled when the form is valid, meaning all required inputs are completed correctly.
- **Switch to Pricing Tab**:
  - Upon successful project submission, the system automatically switches to the **Pricing** tab, allowing the mentor to move on to the next step of the course creation process.

Step 3. 1.2.5: Pricing and Promotion Tab

The **Pricing and Promotion** tab enables the mentor to define the pricing structure of the course and apply discounts through a coupon system. The tab consists of two main sections: **Course Fee** and **Discount**.

1. Course Fee Section

- **Choose a Currency**:
  - The mentor selects the appropriate **currency** from a dropdown menu. Available currencies could include **INR**, **USD**, **EUR**, etc.
- **Enter the Course Fee**:
  - The mentor specifies the **total fee** for the course based on the selected currency.
  - The system will calculate any discount automatically if applied later in the discount section.

---

2. Discount Section

- **Discount Percentage**:
  - The mentor can enter the **discount percentage** that will be subtracted from the total course fee when the coupon is applied.
- **Expiration Date**:
  - The mentor chooses an **expiration date** for the discount using a date picker. This date determines how long the discount will be available to potential students.
  - If no expiration date is selected, the system will prompt the user to choose one before proceeding with the course submission.

Creating a Discount Coupon

- **Submit Course**:
  - After entering the course fee, discount percentage, and expiration date, the mentor clicks the **Submit Course** button to complete the course creation process.
- **Coupon Generation**:
  - Upon submission, the system generates a unique **discount coupon** with the applied percentage and expiration date.
- **Confirmation**:
  - The system will display a confirmation alert, informing the mentor that the coupon has been successfully created and showing the coupon details.

Redirect to Dashboard

- After successfully creating the course and its corresponding discount coupon, the system will **redirect the mentor to the Mentor Dashboard**, where they can review the course pricing and available promotions.

Step 3. 1.2.6.1: Curriculum – Long Course Format

When mentors choose the **long-term** option during the **course landing** phase, the curriculum page adapts to a modular format where lessons, videos, quizzes, and assignments are associated with individual modules. This format is designed for courses with an extended time period, allowing the mentor to structure content in a hierarchical format, with multiple lessons, quizzes, and assignments within each module.

Curriculum Page: Long Course Format

Upon entering the **Curriculum** page for a long course, mentors are presented with the following features:

Module Management

- **Default Module Name**:
  - The page begins with a **default module** titled "Module 1." Mentors can rename this module as needed.
- **Save Module**:
  - After renaming the module (if desired), the mentor clicks the **Save Module** button. This action creates the first course module.
- **Module Navigation**:
  - Once the module is saved, a **dropdown navigation** button appears. When clicked, it expands to reveal the **Lessons** section, along with associated quizzes and assignments for that module.

Lesson Management

- **Lesson Title**:
  - The mentor provides the **title** for each lesson within the module.
- **Lesson Content**:
  - This field outlines what the lesson will cover. It is the main content body that can include text, explanations, or topics.
- **Lesson Description**:
  - Here, the mentor provides a summary or **description** of what is covered in the lesson, along with any additional content or objectives.
- **Lesson Duration**:
  - The mentor enters the **estimated duration** for completing the lesson (in minutes or hours).
- **Save Lesson**:
  - Upon filling in all the lesson details, the mentor clicks the **Save Lesson** button, which generates a **lesson ID** that associates the lesson with the module.

After saving, mentors are now able to proceed to add **videos**, **quizzes**, and **assignments** for the specific lesson.

Videos Section

- **Add More Videos**:
  - Mentors can click the **Add More Videos** button to start uploading lesson videos.
- **Choose File**:
  - The mentor uses the **choose file** option to upload a video file.
- **Video Description**:
  - After selecting a video, the mentor enters a **description** to provide additional context for the video content.
- **Upload Video**:
  - Once the video file and description are provided, the mentor clicks the **Upload** button to save the video.
- **Add Additional Videos**:
  - If more videos are required, the mentor can click the **Add More Videos** button again and repeat the process. Multiple videos can be uploaded this way.

Assignment Section

Mentors can create assignments for each lesson using the following inputs:

- **Title**:
  - The title of the assignment.
- **Topic**:
  - A brief description of the assignment's **topic** or subject matter.
- **Start Date**:
  - The date on which the assignment will be made available to students.
- **End Date**:

    - The deadline for completing the assignment.
- **Review Meet Date**:
    - A date for the mentor to review the assignment submissions with students (if applicable).
- **Choose File**:
    - Mentors can attach supporting documents (e.g., PDFs, DOCs) related to the assignment using the **Choose File** option.
- **Upload Assignment**:
    - After entering all the details, the mentor clicks **Upload** to save the assignment. The system will generate an assignment ID linked to the lesson.

Quiz Section

Mentors can create quizzes for each lesson as follows:

- **Add Quiz Creator**:
    - The mentor clicks **Add Quiz Creator**, opening a popup window with the following inputs:
    - **Quiz Name**:
        - The name of the quiz.
    - **Start Date and Time**:
        - When the quiz will become available.
    - **End Date and Time**:
        - The deadline for completing the quiz.
- **Save Quiz**:
    - After filling out the quiz details, the mentor clicks the **Save Quiz** button, generating a quiz ID for the lesson.

Once the quiz ID is created, mentors can proceed to add questions:

- **Question Creation**:
    - A popup appears where the mentor can enter a **question** and provide four multiple-choice **options**. The correct answer must be selected.
- **Add Question**:
    - After entering a question and its choices, the mentor clicks **Add Question** to store the question. This process can be repeated to add multiple questions to the quiz.
- **Create and Save Quiz**:
    - Once all questions are added, the mentor clicks **Create and Save** to finalize the quiz creation.

Summary of the Long Course Format

In the **long course format**, mentors can:

1. **Create and name modules**.

2. **Add lessons** to each module with details like title, content, description, and duration.
3. **Upload multiple videos** for each lesson.
4. **Create assignments** with associated files and deadlines.
5. **Develop quizzes** with multiple-choice questions.

This modular format allows for deep content structuring, with each module serving as a distinct learning unit comprising lessons, videos, quizzes, and assignments.

Step 3. 1.2.6.2: Curriculum – Short Course Format

In the **short-term course format**, the curriculum creation process is streamlined compared to the long-term course format. Instead of dealing with modules, the mentor directly adds lessons, which can contain videos, quizzes, and assignments. The process is similar to the long-term course but without the module hierarchy.

Curriculum Page: Short Course Format

After choosing the **short-term** course option in the **Course Landing** page, mentors are directed to the **Curriculum** section where they can immediately start adding lessons.

Lesson Management

- **Add Lesson**:
  - Mentors can directly click the **Add Lesson** button to create a new lesson. Since there are no modules in short courses, each lesson is created independently.

Once the mentor clicks the **Add Lesson** button, they will be presented with the following fields:

- **Lesson Title**:
  - Input the **title** for the lesson.
- **Lesson Content**:
  - This field outlines the **content** of the lesson, describing what the lesson will cover.
- **Lesson Description**:
  - A brief **description** providing an overview of the lesson's objectives and additional content.
- **Lesson Duration**:
  - The **duration** of the lesson (in minutes or hours).
- **Save Lesson**:
  - After filling in all details, the mentor clicks the **Save Lesson** button, which generates a **lesson ID**. This ID will be used for associating videos, quizzes, and assignments with the specific lesson.

Once the lesson is saved, mentors can proceed to add **videos**, **quizzes**, and **assignments** for the lesson.

Videos Section

After saving a lesson, the mentor can begin adding videos to it:

- **Add More Videos**:
  - Click the **Add More Videos** button to start uploading videos for the lesson.
- **Choose File**:
  - The mentor uses the **choose file** option to select a video file for upload.
- **Video Description**:
  - After selecting a video, the mentor enters a **description** for the video, providing context or additional details about the video content.
- **Upload Video**:
  - After entering the details and selecting the file, the mentor clicks **Upload** to save the video.
- **Add Additional Videos**:
  - If more videos are required, the mentor can click **Add More Videos** again and repeat the upload process.

Assignment Section

To create assignments for the lesson, the mentor uses the following fields:

- **Title**:
  - Enter the **title** of the assignment.
- **Topic**:
  - Specify the **topic** or subject of the assignment.
- **Start Date**:
  - Choose the **start date** for when the assignment will be made available to students.
- **End Date**:
  - Set the **end date** or deadline for completing the assignment.
- **Review Meet Date**:
  - Enter the date for a **review meet**, where the mentor can discuss the assignment with students (optional).
- **Choose File**:
  - Upload any relevant assignment documents using the **choose file** option.
- **Upload Assignment**:
  - Click **Upload** to save the assignment. The assignment will be linked to the lesson using the lesson ID.

Quiz Section

For creating quizzes related to the lesson:

- **Add Quiz Creator**:
  - Click **Add Quiz Creator**, which opens a popup with the following fields:
  - **Quiz Name**:

        - Enter the **name** of the quiz.
    - **Start Date and Time**:
        - Set the **start date and time** for when the quiz will become available.
    - **End Date and Time**:
        - Specify the **end date and time** for completing the quiz.
- **Save Quiz**:
    - After filling in the quiz details, click **Save Quiz** to generate a quiz ID for the lesson.

Once the quiz ID is created, the mentor can add questions:

- **Question Creation**:
    - A popup appears, allowing the mentor to enter a **question** and four possible **choices**. One of the choices should be marked as the **correct answer**.
- **Add Question**:
    - After entering the question and choices, the mentor clicks **Add Question** to save it. This can be repeated to add multiple questions to the quiz.
- **Create and Save Quiz**:
    - Once all questions have been added, the mentor clicks **Create and Save** to finalize the quiz.

Summary of the Short Course Format

In the **short course format**, the mentor directly creates lessons without organizing them into modules. Each lesson has the same structure and options as in the long course format:

1. **Add lessons** with details like title, content, description, and duration.
2. **Upload videos** for each lesson.
3. **Create assignments** with associated files and deadlines.
4. **Develop quizzes** with multiple-choice questions.

Each lesson is managed independently, allowing the mentor to focus on adding and managing lessons in a streamlined manner.

Step 3.2: Scheduling a Class for an Online Course

When we click the **Add Content** button, three cards appear, allowing the user to:

1. **Add Course**
2. **Schedule Class**
3. **Add Mock Interview**

This step focuses on **scheduling a class** for a course that has already been created. The **Schedule Class** option provides a shortcut to set up a scheduled class for an existing **online course**.

Scheduling a Class for an Online Course

The **Schedule Class** tab is specifically designed for setting up online classes for courses that were created in the **online mode**. It provides a streamlined way to manage the schedule for these courses and link students to their respective classes.

When the user clicks on the **Schedule Class** card, they will be directed to a page where they can see a table of courses that are already created in the system.

Table Overview

The table displays a list of online courses and their details, including:

- **Course Title**: The title of the course.
- **Language**: The language in which the course is conducted.
- **Level**: The level of the course (Beginner, Intermediate, Advanced), which corresponds to the options chosen during course creation (as seen in the **Course Landing** section).
- **Schedule a Class**: This column contains an icon/button that allows the user to schedule a class for that specific course.

Scheduling a Class

When the user clicks the **Schedule a Class** icon/button in the table, a popup window appears, allowing them to enter the details for scheduling the class. The popup includes the following fields:

- **Room Name**: A unique name for the virtual classroom where the session will take place.
- **Timing**: The scheduled time for the class, where the user specifies the start time for the class.

After filling in the details, the user clicks the **Schedule** button to confirm the class schedule. Upon successful scheduling, the class will be added to the system with the specified time.

Viewing Scheduled Classes

Once a class is scheduled, the system provides additional functionalities in the table under a new column called **Training Rooms**.

In this column, users can:

- **View Scheduled Class Timings**: Each row displays the time at which the class is scheduled to take place.
- **Go Button**: A **Go** button is available in the **Training Rooms** column, which allows the user to navigate to the virtual classroom. However, this button will only be enabled once the scheduled time arrives, ensuring users can only enter the classroom when the class is active.

Step 3.3: Add Mock Interview

When clicking on the **Add Content** button, users are presented with three cards:

1. **Add Course**
2. **Schedule Class**
3. **Add Mock Interview**

This step focuses on the **Add Mock Interview** card, which provides functionality to create a mock interview process for a particular course.

Mock Interview Creation

Upon selecting the **Add Mock Interview** card, the system opens a popup that allows the user to create the details for a mock interview. The popup includes the following fields:

1. **Upload Image**: The user can upload an image representing the mock interview (such as a company logo or related image).
2. **Title**: The title of the mock interview (e.g., "Technical Round Mock Interview").
3. **Description**: A brief description of the mock interview (e.g., "This interview simulates a technical round for software engineering roles").
4. **Course Name**: The course for which the mock interview is being set up (dropdown list of available courses).
5. **Fee**: The fee associated with the mock interview if it's a paid service (input field for currency and amount).
6. **Free Attempts**: The number of free attempts allowed for the interview (input field for the number of attempts).
7. **Test Type**: The type of mock interview test (e.g., "Technical Test", "Behavioral Test", "Aptitude Test"), which can be selected from a dropdown menu.

Save Mock Interview Details

Once the user has filled in all the fields, they can click the **Save Mock Details** button. Upon clicking this, the system will create the mock interview record and store the details for future use.

Creating Slots for the Mock Interview

After successfully creating the mock interview, the user can proceed to the next step: **Creating Slots** for the interview assessment rounds.

The user will now see the option to define available slots for the mock interview process, which allows candidates to book time slots for their assessments. The slot creation involves the following:

1. **Slot Date**: The date on which the slot is available for candidates.
2. **Slot Timing**: The start and end times of the interview slot (e.g., 10:00 AM to 11:00 AM).

3. **Max Candidates per Slot**: The maximum number of candidates who can book this particular slot (e.g., 5 candidates per slot).

Once the user creates a slot for the mock interview, candidates will be able to view and book available slots for their assessments.

Summary of the Add Mock Interview Process

1. **Click Add Mock Interview**: The user selects the **Add Mock Interview** card to create a mock interview.
2. **Fill Mock Interview Details**: A popup appears where the user can fill in details such as **Upload Image**, **Title**, **Description**, **Course Name**, **Fee**, **Free Attempts**, and **Test Type**.
3. **Save Mock Interview**: After completing the form, the user clicks the **Save Mock Details** button to create the mock interview.
4. **Create Slots for Assessment**: After the mock interview is created, the user can define **interview slots** (date, time, and max candidates).

This process allows users to create and manage mock interviews efficiently, along with setting up assessment slots for candidates to practice. Let me know if you'd like to expand on any specific part of the flow!

Step 3.4: Dashboard - In Progress and Completed Courses

In this step, the **Dashboard Tab** serves as a progress tracker for the courses created by the mentor. The mentor can view the status of their courses, edit or delete them, and monitor their completion progress. The courses are categorized under two sections: **In Progress** and **Completed**.

Dashboard Interface

The **Dashboard** features two radio buttons:

- **In Progress**: Displays courses that are **less than 90% filled** by the mentor.
- **Completed**: Displays courses that are **90% or more filled**.

Each course has a progress percentage based on how much information has been filled out during the course creation process (as detailed in Step 3.1).

Course Display Cards

In both the **In Progress** and **Completed** sections, each course is represented by a **course card**. Each card contains the following details:

1. **Course Image**: Displays the cover image of the course.

2. **Course Progress**: A visual representation (e.g., progress bar or percentage) showing how much of the course information has been filled by the mentor (e.g., 85%, 95%).
3. **Last Updated Timeline**: The date and time when the course was last updated by the mentor.
4. **Edit Button**: Clicking this button allows the mentor to make edits to the course. The mentor will be redirected to the course creation steps (from **Step 3.1**) where they can modify or update details such as course landing, trailer, planning, curriculum, project, and pricing.
5. **Delete Button**: Clicking this button will allow the mentor to delete the course. A confirmation prompt will ensure the mentor confirms the deletion before proceeding.

In Progress vs. Completed Courses

- **In Progress Courses**: Courses that are **less than 90% filled** by the mentor. These courses are displayed in the **In Progress** section. The mentor can see the progress percentage and can complete or update the missing details.
- **Completed Courses**: Courses that are **90% or more filled** by the mentor. These courses appear in the **Completed** section, indicating that they are nearly ready for publication or distribution.

Edit Course Functionality

When the **Edit** button is clicked on any course card, the mentor will be redirected to the course creation flow, where they can edit the previously filled data. The mentor will be able to edit all the steps from **Step 3.1**, including:

1. **Course Landing Page**: Where the title, description, and other introductory information can be edited.
2. **Course Trailer**: Where the course's promotional video can be changed or updated.
3. **Course Planning**: Where learning objectives, prerequisites, and other plans for the course can be modified.
4. **Curriculum**: Where modules (for long courses) or lessons (for short courses) can be updated. Videos, quizzes, and assignments related to the lessons can also be edited.
5. **Course Project**: Where project details like the title, description, deadline, and project document can be changed.
6. **Pricing and Promotions**: Where the course fee, discount, and promotional details can be edited.

The course is updated dynamically, and changes are saved once the mentor completes the editing process.

Delete Course Functionality

If the **Delete** button is clicked, the mentor will be prompted with a confirmation dialog before permanently deleting the course. Once deleted, the course will no longer appear in either the **In Progress** or **Completed** sections.

Step 3.5: Dashboard - Search and Filter Courses

In the final stage of the **Dashboard Tab** (Step 3.5), mentors are provided with a **search input box** that enables them to filter through the list of courses they have created. This feature simplifies the process of locating specific courses from potentially large lists by applying search and filter criteria.

Search Input Box Functionality

The **Search Input Box** allows mentors to quickly search for courses they have created. It includes the following features:

- **Search by Course Title**: The mentor can type part or all of the course title to search for specific courses. The search will display only the courses that match the entered title or partial title.
- **Real-time Filtering**: As the mentor types in the search box, the list of displayed courses is filtered in real-time. Courses that do not match the search term are hidden, while those that match are shown dynamically.

Search and Filter Process

1. **Default Display**: When the mentor first accesses the dashboard, all courses (both **In Progress** and **Completed**) are displayed.
2. **Entering Search Terms**: As the mentor begins typing in the **search box**, the system starts filtering the courses that match the search term based on their titles.
3. **Results Display**:
    - If courses matching the search term are found, they are displayed as **course cards**.
    - If no courses match the search term, a **"No Courses Found"** message is displayed, indicating that no courses fit the search criteria.

Step 4: Overview of Mentor Online Component

4.1 Frontend Component Initialization

- The `MentoronlineComponent` is initialized when the user navigates to the relevant route.
- During `ngOnInit`, the `userId` is fetched from localStorage. If available, the component triggers a service call to fetch the mentor's online courses.

4.2 Fetching Courses (Frontend to Backend)

- The `getCourses()` method in the component calls a service method to fetch online courses using a GET request to `/course/online/byMentor/{userId}`.
- The backend controller retrieves the mentor's courses and sends them back to the frontend.

- Once received, the component maps each course to include a toggle property (`showRooms`) for managing training rooms visibility and stores them for display.

4.3 Displaying Courses

- Courses are displayed in a table format in the HTML, including details like course title, language, and level.
- Each course has a button for scheduling a class and a toggle button to show/hide associated training rooms.

4.4 Training Room Toggle

- Clicking the toggle button calls `toggleRooms()` to expand/collapse the training room section.
- If training rooms haven't been fetched, `getTrainingRooms()` is triggered to fetch them from the backend.

4.5 Fetching Training Rooms

- The `getTrainingRooms()` method sends a GET request to `/api/training-rooms/by-course/{courseId}/{teacherId}`.
- The backend returns a list of training rooms for the specific course and teacher.
- The component updates the course with the fetched rooms and displays them in a nested table.

4.6 Displaying Training Rooms

- Training rooms are displayed beneath the course row and include room name, scheduled date/time, and a conference URL.
- The conference link is clickable if the scheduled time hasn't passed; otherwise, it's disabled and visually indicated.

4.7 Join Conference Functionality

- The **Join** button allows users to open the conference link in a new tab.
- If the scheduled time has passed, the button is disabled using the `isScheduledTimeCompleted()` method.

Backend Process

4.8 Fetching Courses by Mentor

- The Spring Boot controller exposes `/online/byMentor/{userId}` to fetch online courses for a specific mentor.
- The corresponding service retrieves courses from the database based on the `userId` and course type ("online").

4.9 Fetching Training Rooms by Course

- The controller at `/by-course/{courseId}/{teacherId}` fetches training rooms for a given course and teacher.
- The service queries the database for matching training rooms and sends them back to the frontend.

Step 9: Analytics Component Overview

9.1 Component Initialization (`ngOnInit`)

- When the component initializes, it triggers the `loadCourses()` method to fetch courses for the current user.
- The method `FusionService.getCoursesByUserId()` is called, and the fetched courses are stored in the `courses` array.
- After loading courses, enrollment data for each course is fetched to populate the **Course Enrollment** chart.

9.2 Course Enrollment Data

- Once courses are loaded, the system makes individual calls to fetch enrollment data for each course.
- The **Course Enrollment Pie Chart** is updated by calculating the number of users enrolled in each course.
- The `updateCourseEnrollmentChart()` method dynamically updates the chart with the enrollment data for each course.

9.3 Handling User Clicks on the Pie Chart

- When a user clicks on a course in the pie chart, the course ID is retrieved, triggering the `updateChartsForCourse()` method.
- This method refreshes both the **Student Progress** and **Assessment Completion** charts, displaying data specific to the selected course.

9.4 Fetching Student Progress Data

- After a course is selected, the system fetches student progress data using `FusionService.getEnrolledUsersProgress()`.
- The student names and progress percentages are extracted and used to update the **Student Progress Bar Chart**.
- The chart displays the progress percentage for each student enrolled in the selected course, dynamically rendering the fetched data.

9.5 Fetching Assessment Completion Data

- The system simultaneously fetches **Assessment Completion** statistics for the selected course.
- Monthly statistics of total enrollers and submitted assessments are fetched from the backend and displayed on the **Assessment Completion Line Chart**.
- The chart shows how the number of enrollers and submitted assessments changes over time.

9.6 Chart Customization and Responsiveness

- Each chart (Pie, Bar, Line) is configured with options for responsiveness, legends, titles, and click events.
- The Pie Chart includes an `onClick` event listener that allows user interactions, triggering updates in other charts.

9.7 User Data from Local Storage

- The component retrieves the user ID from localStorage during initialization to load data specific to the logged-in user.
- The `userId` is passed into service calls that fetch user-specific course and progress data.

Step 10: Mock Interview Tab

10.1 Mock Interview Slot Creation

- **`saveInterviewSlots()`**: This method loops through the interview slots and creates a payload for each slot. It sends the data to the backend using `mockService.saveSlot()`.

10.2 Mock Test Creation

- **`saveMockInterviewDetails()`**: This method handles the submission of mock test interview form details, sending them to the backend via `mockService.createMockTestInterview()`. It also processes file uploads (such as an image for the mock test) using `FormData`.

10.3 Assessment and Project Submission

- **`submitAssessmentForm()`**: This method is triggered when the user submits an assessment form.
- **`submitProjectForm()`**: This method processes the submission of a project, including handling file uploads for project documents.

10.4 Interview Navigation

- **`viewDetails()`**: This method handles navigation based on the form type (assignment, project, or interview). It redirects the user to the appropriate feedback page.

10.5 Form Validation

- Use Angular's **`FormGroup`** and **`FormControl`** to enable efficient validation in reactive forms, ensuring that forms are fully validated before submission.

10.6 Handling Form Data

- In the **`saveMockInterviewDetails()`** method, form data is manually appended to a `FormData` object. Consider centralizing the validation and processing of form data to ensure consistency across different methods.

10.7 File Upload Handling

- Ensure proper validation of file types and sizes in the **`onFileChange()`** method before appending files to the `FormData`.

10.8 UI Feedback

- Provide user feedback when forms are successfully submitted or if errors occur (e.g., using Angular Material **Snackbar** or alerts). This enhances the user experience.

Step 5: Courses Tab

5.1 Courses Section

- Displays a list of courses with details such as:
  - **Course Title**
  - **Enrolled Students**
  - **Course Type** (e.g., Online, Offline)
- Allows mentors to perform actions like navigating to create assignments, projects, or quizzes for each course.
- Provides a way to view enrolled student data for each course by dynamically updating the **enrolledStudents** field.

5.2 Assignments Section

- Displays assignments related to the courses in a table format.
- Includes functionality for:
  - **Editing Assignments**: Users can click to edit assignment details.
  - **Deleting Assignments**: Provides a delete option for removing assignments.

- **Loading Assignments**: Automatically fetches and loads assignments for the mentor based on the course data.

5.3 Projects Section

- Displays project details such as:
  - **Project Title**
  - **Course Title**
  - **Course Type**
- The system fetches project data associated with the teacher and binds it to the projects table.
- Implements functionality for:
  - **Editing Projects**: Provides the ability to modify project information.
  - **Deleting Projects**: Allows the removal of projects.

5.4 Quizzes Section

- Lists quizzes related to the courses with details such as:
  - **Quiz Name**
  - **Course Title**
  - **Course Type**
- Provides options to create, edit, and delete quizzes as needed.

5.5 Additional Features

- **Loading Data**: In the `ngOnInit()` lifecycle hook, you load data for courses, assignments, projects, and quizzes by calling appropriate services. The data fetched from the backend is based on the **user ID** or **teacher ID** retrieved from localStorage.
- **Course Enrollments**: For each course, enrollment data is fetched and dynamically updates the **enrolledStudents** field. This allows mentors to keep track of how many students are enrolled in their courses.
- **Dialogs**: There is a dialog (`CourseDialogComponent`) for creating or updating course details. This modal-style dialog allows for smooth interaction when adding or editing course information.

5.6 Potential Enhancements/Notes

- **Error Handling**: You have integrated error logging mechanisms, particularly when there is an issue with the **userId** or **teacherId** in localStorage or when API calls fail. This ensures errors are captured and displayed appropriately.
- **Navigations**: You utilize Angular's **Router** to navigate between routes, allowing users to seamlessly move between creating and updating assignments, projects, or quizzes.
- **UI Components**: The use of **Angular Material** components such as buttons, icons, and tables enhances the overall user experience. The design ensures that the courses section is user-friendly and intuitive.

This detailed breakdown ensures that the mentor can effectively manage their courses, assignments, projects, and quizzes within the application.

Step 8: Progress Tab

8.1 Course Selection

- The progress tab begins by presenting the user with a **"Course Management"** heading and a dropdown menu to select a course.
- **Available Courses**: The available courses are populated in the dropdown.
- When a course is selected, it triggers the **`onActivityCourseChange`** function, which loads the necessary modules or lessons based on the course type (short-term or long-term).

8.2 Module/Lesson Selection

- **For Long-Term Courses**:
  - A **Module Selection Dropdown** appears if the selected course is a long-term course.
  - When a module is selected, it triggers the **`onActivityModuleChange`** function, which loads the lessons for the selected module.
  - After selecting a module, a **Lesson Dropdown** appears with the available lessons.
  - Selecting a lesson triggers the **`onLessonChange`** function, loading relevant tasks like quizzes or assessments.
- **For Short-Term Courses**:
  - A **Lesson Selection Dropdown** appears directly, as modules are not present in short-term courses.
  - When a lesson is selected, it triggers the **`onActivityLessonChange`** function.

8.3 Task Selection

- After selecting a lesson, a **Task Selection Dropdown** appears, allowing users to choose between **Quiz** or **Assessment**.
- The selection triggers the **`onTaskChange`** function, dynamically loading either quizzes or assessments based on the user's choice.

8.4 Quiz/Assessment Selection

- **For Quizzes**:
  - If **Quiz** is selected as the task, a dropdown appears for selecting a specific quiz from the available quizzes for that lesson.
  - Selecting a quiz triggers the **`onActivityQuizChange`** function, which loads the enrollers (students) who attempted the quiz.
- **For Assessments**:
  - If **Assessment** is selected, a dropdown appears for selecting a specific assessment related to the selected lesson.
  - Selecting an assessment triggers the **`onActivityAssessmentChange`** function, loading enrollers who submitted the assignment.

8.5 Enroller Selection

- After selecting a quiz or assessment, a **table** is displayed listing the enrollers (students) who participated in the quiz or submitted the assessment.
- The table includes the following fields:
  - **Serial Number**
  - **Name**
  - **Email**
  - **View Button** to open enroller details.

8.6 Viewing Enroller Details

- When the **"View"** button is clicked for an enroller, a **modal window** opens displaying details based on the selected task (quiz or assessment).
- **For Quizzes**:
  - The modal shows:
    - Quiz Questions
    - The Enroller's Answers
    - Correct Answers
    - The Score obtained by the enroller.
- **For Assessments**:
  - The modal displays:
    - Assignment content submitted by the enroller.
    - Options to **Download** or **Preview** the assignment file.

8.7 Providing Feedback

- The modal includes a feedback section where the instructor can:
  - **View the Quiz Score** (for quizzes).
  - **Enter a Grade** (for assessments).
  - **Write Comments** to provide feedback to the student.
- A **"Submit"** button is available for the instructor to submit the feedback after reviewing the enroller's performance.

8.8 Document Viewer (for Assessments)

- For assignments, the progress tab includes a **Document Viewer** feature, allowing the instructor to preview the submitted assignment directly within the interface.

8.9 Key Functions

- **`onActivityCourseChange`**: Handles course selection and triggers further dropdowns for modules or lessons based on course type.
- **`onActivityModuleChange`**: Handles module selection for long-term courses and loads related lessons.
- **`onLessonChange`** / **`onActivityLessonChange`**: Handles lesson selection based on the course and module.

- **`onTaskChange`**: Triggers loading of tasks (quiz or assessment) after lesson selection.
- **`onActivityQuizChange`**: Handles quiz selection and loads enroller data for the selected quiz.
- **`onActivityAssessmentChange`**: Handles assessment selection and loads enroller data for the selected assessment.
- **`openModal`**: Opens the modal window displaying quiz or assessment details.
- **`closeModal`**: Closes the modal window.
- **`downloadAssignment`**: Allows the instructor to download the enroller's submitted assignment file.
- **`previewAssignment`**: Opens the document viewer to preview the submitted assignment.
- **`submitFeedback`**: Submits the instructor's feedback for the enroller.
- **`closeViewer`**: Closes the document viewer after previewing the assignment.

This detailed step ensures a streamlined experience for instructors, allowing them to manage course-related progress effectively, from selecting a course to providing feedback on quizzes and assessments.

Step 6: Online Course Tab

6.1 Frontend Component Initialization

- **Component**: The `MentorOnlineComponent` is initialized when the user navigates to the appropriate route for managing online courses.
- **User ID Fetch**: During initialization (`ngOnInit`), the component fetches the `userId` from `localStorage`.
    - If the `userId` is available, a service call is triggered to fetch the online courses associated with the specific mentor by calling `getCourses()`.

6.2 Fetching Courses (Frontend to Backend Communication)

- The `getCourses()` method in the component calls a service that sends a **GET request** to the backend, targeting the endpoint `/course/online/byMentor/{userId}`.
- **Backend Controller**: The controller at `/online/byMentor/{userId}` handles this request, retrieves the courses for the specified mentor (based on `userId` and course type "online"), and returns them in the response.
- **Courses Array**: Once the data is received on the frontend, the component processes the course data, adding a property (`showRooms`) for toggling the visibility of associated training rooms.
    - The courses are then stored in the `courses` array and displayed in the interface.

6.3 Displaying Courses

- In the **HTML**, courses are rendered in a table format with columns displaying:
    - **Course Title**
    - **Course Language**
    - **Course Level**
- **Additional Actions**:
    - A button or icon for navigating to schedule an online class for each course.

- A **Toggle Button** to expand and view the training rooms for each course.

6.4 Training Room Toggle

- When a user clicks the **Toggle Button** for a course, the `toggleRooms()` method is called. This method:
  - Toggles the `showRooms` flag, which either expands or collapses the section displaying the training rooms for that course.
  - If the training rooms for the selected course have not yet been fetched (i.e., `trainingRooms` is undefined), the `getTrainingRooms()` method is triggered to fetch the rooms.

6.5 Fetching Training Rooms

- The component's `getTrainingRooms()` method sends a **GET request** to the backend at `/api/training-rooms/by-course/{courseId}/{teacherId}` to retrieve training rooms associated with a specific course and teacher.
- **Backend Controller**: The controller at `/by-course/{courseId}/{teacherId}` handles the request, retrieves the relevant training rooms from the database, and sends them back to the frontend.
  - If no rooms are found, the controller returns an empty response, which prompts the frontend to display a message indicating that no training rooms were found.

6.6 Displaying Training Rooms

- Once the training rooms are fetched, they are displayed in a **nested table** beneath the respective course row.
  - The training rooms table includes:
    - **Room Name**
    - **Scheduled Date & Time**
    - **Conference URL**
- **Conference Link**:
  - The **Conference URL** is clickable, allowing users to join the training room session.
  - If the scheduled time for the training room has already passed, the **Join button** is disabled and its color changes to indicate the link is no longer available.

6.7 Join Conference Functionality

- The **Join button** allows users to open the conference link in a new tab.
  - However, if the scheduled time for the training room has passed, the button is dynamically disabled using the `isScheduledTimeCompleted()` method.

Backend Process

6.8 Fetching Courses by Mentor

- The **Spring Boot controller** exposes an endpoint `/online/byMentor/{userId}` to retrieve online courses for a given mentor.
- **Service Layer**: The corresponding service method fetches courses from the database by filtering based on the `userId` and course type ("online").
    - The courses are then returned to the frontend component.

6.9 Fetching Training Rooms by Course

- When the frontend requests training rooms for a specific course and mentor, the backend controller at `/by-course/{courseId}/{teacherId}` processes the request.
- **Service Layer**: The service queries the database for training rooms associated with the given `courseId` and `teacherId`.
    - The rooms are returned to the frontend for display.
    - If no rooms are found, an empty response is sent, and the frontend shows a message indicating no available rooms.

Students Tab in mentor-dashboard

Overview

This document outlines the key features and functionalities of the Student Assessment Management System built using Angular. It aims to guide the testing team through the system's user interface, highlighting critical functionalities, testing areas, and expected outcomes.

Purpose

The Student Assessment Management System allows educators to manage student assessments, projects, quizzes, and online classes efficiently. The application leverages Angular's reactive forms and template-driven forms to capture user inputs, ensuring a dynamic and user-friendly interface.

Key Features

1. **Action Buttons**
    a. **Add Assessment**: Opens an overlay form to input assessment details.
    b. **Add Project**: Similar functionality to the assessment button, facilitating project creation.
    c. **Add Quiz**: Initiates quiz creation, allowing users to enter quiz details and multiple-choice questions.
    d. **Schedule Online Class**: Provides a form to schedule online classes for selected courses.
2. **Overlay Forms**
    a. Each button displays a corresponding overlay for user input.
    b. Overlays are conditionally rendered based on component state variables (e.g., `showAssessmentOverlay`, `showOverlay`, etc.).
3. **Form Controls**
    a. Utilizes Angular's reactive forms and template-driven forms for input binding.
    b. Supports various input fields: text boxes, select dropdowns, datetime inputs, and file uploads.

    c. Includes a search feature to filter enrolled users dynamically.
4. **Dynamic User Enrollments**
    a. Displays a checkbox list of users enrolled in a selected course.
    b. Allows filtering based on user input.
5. **File Upload**
    a. Enables users to upload files associated with assessments and projects.
    b. Visual confirmation of selected files is provided.
6. **Quiz Management**
    a. Supports adding multiple-choice questions and true/false questions.
    b. After quiz creation, the form resets, and previous questions are cleared.
7. **Validation and Submission**
    a. Forms include required fields for data integrity.
    b. Submission triggers methods to handle data processing.
8. **User Interaction**
    a. Interactive elements (buttons, overlays, modals) facilitate user engagement and ensure a responsive experience.
9. **Table Structure for Courses**
    a. Displays essential course details, including title, enrolled students count, and assignment/project/quiz counts.
    b. Supports row expansion to show detailed views for assignments, projects, and quizzes.

Testing Scope

The testing team should focus on the following areas to ensure comprehensive coverage of the application:

1. **Functional Testing**
    a. Verify the functionality of each action button (Add Assessment, Add Project, etc.).
    b. Ensure that overlays open correctly and capture user inputs as expected.
    c. Validate that file uploads function properly and provide visual feedback.
    d. Test the dynamic filtering of enrolled users.
2. **Form Validation**
    a. Check that required fields are enforced and appropriate error messages are displayed.
    b. Test the behavior of forms upon successful submission and ensure data integrity.

3. **User Interface Testing**
    a. Confirm that the UI is responsive and that overlays/modal windows are correctly styled.
    b. Validate that the table structure displays data correctly and that row expansion works as intended.
4. **Integration Testing**
    a. Test the integration of the frontend with the backend APIs for submitting assessments, projects, quizzes, and scheduling online classes.
    b. Ensure data retrieved from APIs populates the forms correctly.
5. **Usability Testing**
    a. Evaluate the overall user experience, focusing on the ease of navigating through the system and completing tasks.
    b. Gather feedback on labels, placeholders, and overall interaction flow.

Expected Outcomes

- **Successful Interactions**: Users should be able to add assessments, projects, quizzes, and schedule online classes without errors.
- **Data Integrity**: All user inputs should be captured accurately and stored in the backend.
- **Responsive UI**: The application should function seamlessly across different devices and screen sizes.
- **Error Management**: Any errors encountered should provide clear feedback to the user, enabling corrective actions.