

Documentation of Peer Evaluation System:

User Manual for Peer Evaluation System:

Models:

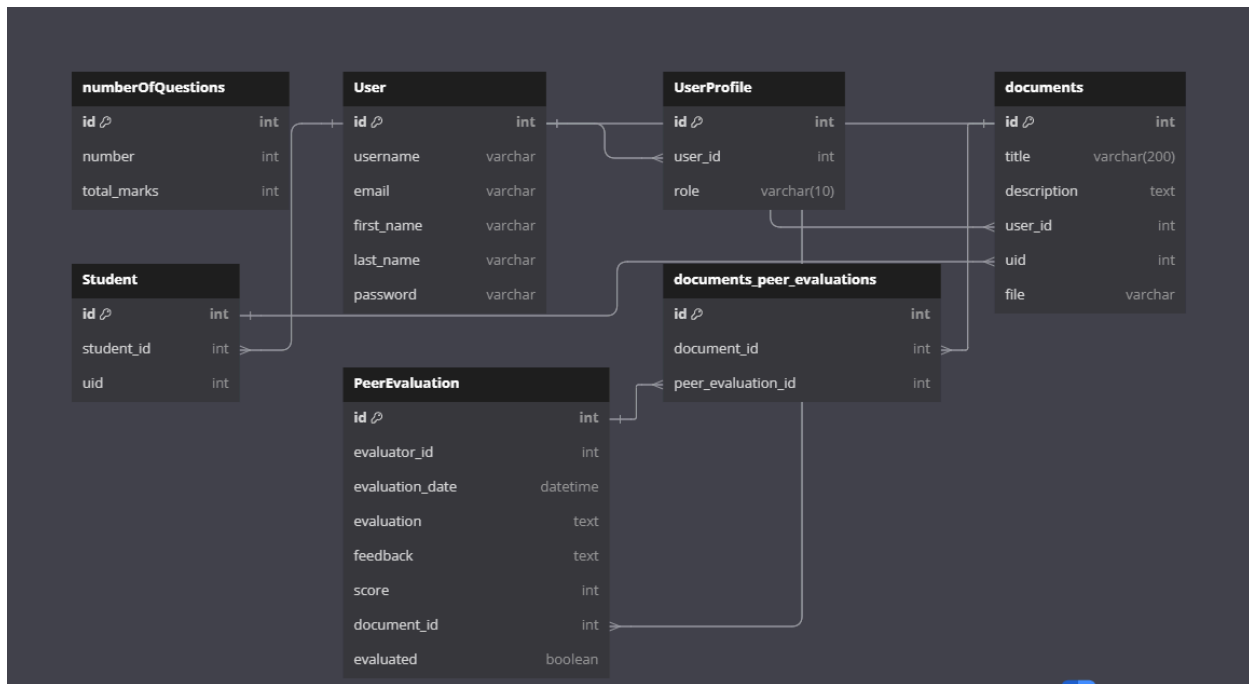


Fig 1: Database ER diagram.

1. Models Overview

1.1 numberOfQuestions

- **Purpose:** Represents the number of questions and the total marks for an evaluation.
- **Fields:**
 - **id:** Auto-incrementing primary key.
 - **number:** Number of questions in the evaluation.
 - **total_marks:** Total marks for the evaluation, defaulting to 0.
- **Methods:**
 - **__str__:** Returns the number of questions as a string.

1.2 UserProfile

- **Purpose:** Extends the **User** model to include roles (e.g., Student, Teacher, TA, Admin).

- **Fields:**
 - **user:** One-to-one relationship with the Django **User** model.
 - **role:** Role of the user, chosen from predefined **ROLE_CHOICES**.
 - Options: **TA**, **Student**, **Teacher**, **Admin**.
 - **Methods:**
 - **__str__:** Returns a string representation in the format **<username> - <role>**.
 - **serialize:** Returns a serialized representation of the user profile, including:
 - **id:** Profile ID.
 - **username:** Associated user's username.
 - **role:** User's role.
-

1.3 Documents

- **Purpose:** Stores information about documents submitted by users, including peer evaluations.
 - **Fields:**
 - **id:** Auto-incrementing primary key.
 - **title:** Title of the document (optional).
 - **description:** Description of the document (optional).
 - **user_id:** Foreign key linking to the **User** who owns the document.
 - **peer_evaluations:** Many-to-many relationship with **PeerEvaluation**, linking peer evaluations to this document.
 - **uid:** Foreign key linking to the **Student** model, identifying the student associated with the document.
 - **file:** File upload field, stores files in the **documents/** directory.
 - **Methods:**
 - **__str__:** Returns the title of the document.
-

1.4 Student

- **Purpose:** Represents student-specific data, linking to the **User** model.
 - **Fields:**
 - **id:** Auto-incrementing primary key.
 - **student_id:** Foreign key linking to the Django **User** model, identifying the user associated with the student.
 - **uid:** Unique identifier for the student.
 - **Note:** The **related_name='students'** for **student_id** allows reverse access from **User** to its associated students.
-

1.5 PeerEvaluation

- **Purpose:** Stores information about peer evaluations performed on documents.
- **Fields:**
 - **evaluator_id:** ID of the user who performed the evaluation.
 - **evaluation_date:** Timestamp of when the evaluation was created (auto-set on creation).
 - **evaluation:** Text field containing the evaluation content.
 - **feedback:** Feedback provided during the evaluation.
 - **score:** Numerical score given during the evaluation.
 - **document:** Foreign key linking to the **documents** model, identifying the document being evaluated.
 - **evaluated:** Boolean field indicating whether the document has been evaluated (default: **False**).

Methods:

- **__str__:** Returns a string representation of the peer evaluation, including the associated document's title.

2. Relationships Between Models

UserProfile

- Extends the **User** model with a **role** field.

Student

- Represents a user in the **Student** role, linking to **User** with a one-to-one relationship.

Documents

- Links to a **User** (who submitted the document) and a **Student** (associated with the document).
- Maintains many-to-many relationships with **PeerEvaluation** for feedback and scoring.

PeerEvaluation

- Links to a **documents** instance and records feedback, evaluation content, and scores.
-

3. Usage Notes

Database Schema

- The relationships and fields ensure that the data structure supports:
 - User role management.
 - Document ownership and evaluation tracking.
 - Peer evaluation functionality.

File Storage

- Uploaded files for documents are stored in the **documents/** directory.

Model Queries

- Access evaluations for a document: **document.peer_evaluations.all()**.
- Access documents owned by a user: **user.documents_set.all()** (reverse relation).

Project:

The project involves managing users, documents, and peer evaluations within a system with various user roles (Admin, Teacher, TA, Student). Below are the inferred logics and workflows implemented in the project:

1. Authentication and User Management

1.1 Login and Registration

- **Login Logic:**
 - Authenticate users based on username and password.
 - Redirect users to the appropriate home page based on their role (e.g., [/AdminHome/](#), [/StudentHome/](#)).
 - Display error messages for invalid usernames or passwords.
- **Registration Logic:**
 - Validate password strength using a regex pattern.
 - Ensure the uniqueness of usernames.
 - Create a new **User** object and associate it with a **UserProfile** having a default role of "Student."

1.2 Logout

- Log out the user and redirect to the login page.

1.3 Forgot Password

- Generate a random password and send it to the user via email.
 - Update the user's password in the database.
-

2. Role Management

- Admins, Teachers, and TAs can modify the roles of other users.
 - Validate role changes to ensure only authorised users can update roles.
-

3. File and Data Management

3.1 File Upload

- Admins, Teachers, and TAs can upload documents (CSV or PDF files).
- Extract data from uploaded files, process them, and save them in the database.

3.2 File Deletion

- Delete all associated records and physical files when instructed.
 - Ensure proper error handling during file system operations.
-

4. Document and Evaluation Management

4.1 Document Upload

- Documents can be uploaded by Admins, TAs, and Teachers.
- Associate uploaded documents with specific students using unique identifiers (UIDs).
- Store the uploaded files and maintain references in the database.

4.2 Peer Evaluation Assignment

- Assign peer evaluations randomly to students.
- Ensure:
 - Students are not assigned their own documents.
 - Each document is evaluated by a specific number of peers ($\sqrt{\text{number_of_documents}}$).
- Send peer evaluation emails with evaluation links to assigned students.

4.3 Evaluation Submission

- Students can evaluate assigned documents.
- Record:
 - Scores for each question.
 - Feedback for each question.
 - Total marks for the evaluation.
- Mark evaluations as "evaluated" upon submission.

4.4 View Evaluations

- Display evaluations for both assigned documents and the student's own submissions.
- Provide an aggregate score for the student's submitted documents.

5. Analytics and Dashboard Features

5.1 Admin Dashboard

- Manage document uploads and initiate peer evaluation assignments in bulk.
- Use background threads for asynchronous operations.

5.2 Teacher Dashboard

- Display:
 - Distribution of average marks for top-performing documents.
 - Total documents submitted.
 - Total, evaluated, and pending peer evaluations.

5.3 Student Dashboard

- Display:
 - Evaluation files assigned to the student.
 - Evaluation status (evaluated or not).
 - Peer reviews and aggregate marks for the student's submitted documents.

6. Email Notifications

6.1 Peer Evaluation Assignment

- Send evaluation links to students when they are assigned a document for review.

6.2 Reminder Emails

- Send reminders for pending evaluations to students with incomplete tasks.

6.3 Forgot Password

- Send randomly generated passwords via email to users who request a password reset.
-

7. Security

7.1 Authorization

- Ensure that users can only access features relevant to their roles:
 - Admins can manage all resources.
 - Teachers and TAs can upload and manage documents.
 - Students can view and evaluate assigned documents.

7.2 Validation

- Validate inputs for:
 - Password strength.
 - Correct file formats for uploads.
 - Proper role-based permissions.

7.3 CSRF Protection

- All forms include CSRF tokens to protect against cross-site request forgery attacks.
-

8. Data Integrity

8.1 Database Management

- Use Django models to define relationships:
 - **UserProfile** links users to their roles.
 - **documents** associates files with users and students.
 - **PeerEvaluation** tracks evaluations for each document.
- Delete records and associated files when necessary (e.g., bulk deletions).

8.2 Unique Identifiers

- Use UUIDs to associate students with their submissions and evaluations.
-

9. Error Handling

- Catch and log errors during operations like:
 - File processing.
 - Peer evaluation assignments.

- Role updates and database transactions.
 - Display user-friendly error messages in the UI.
-

10. Utility Functions

10.1 Password Generation

- Generate strong, random passwords using a mix of uppercase, lowercase, digits, and symbols.

10.2 Email Templates

- Use HTML templates for formatted email notifications.

10.3 Background Threads

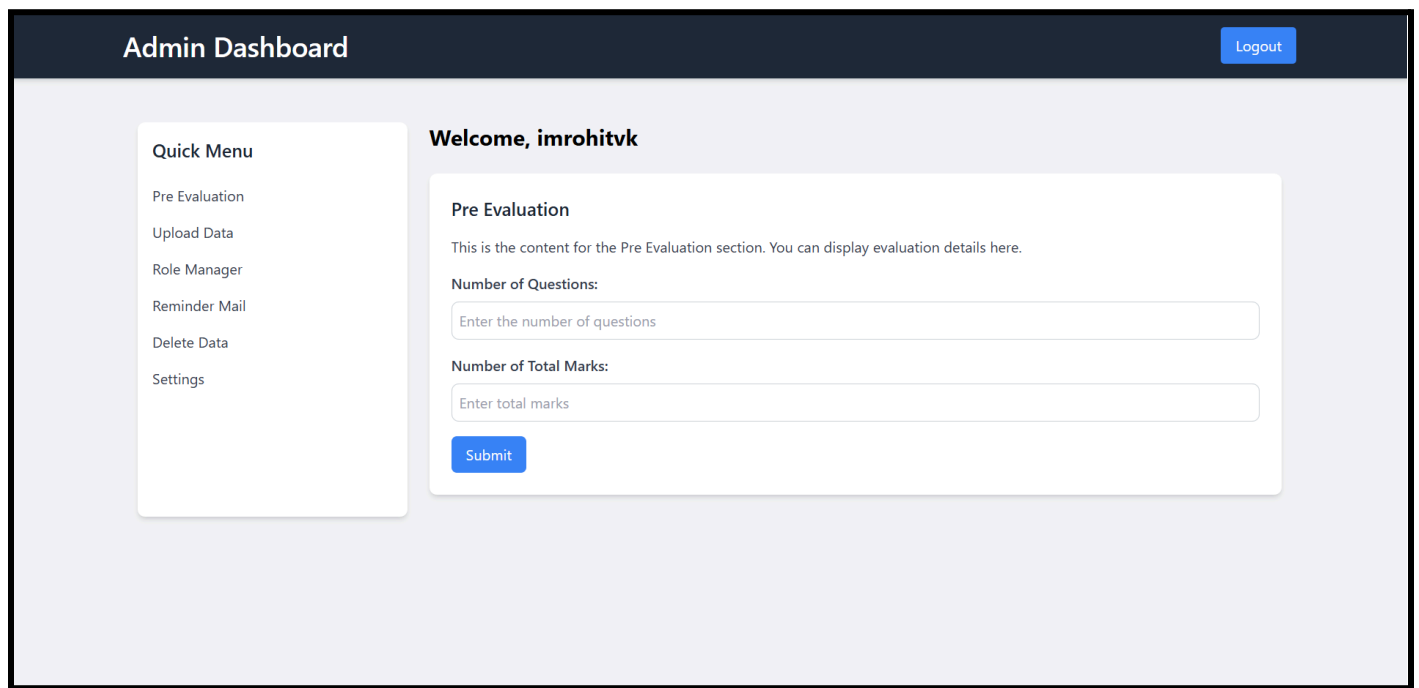
- Use threading for time-intensive tasks like peer evaluation assignment.
-

11. Routing and Navigation

- Role-based redirection:
 - Admins: [/AdminHome/](#)
 - Teachers: [/TeacherHome/](#)
 - TAs: [/TAHome/](#)
 - Students: [/StudentHome/](#)
- Common login page for all users.

Dashboards:

Admin Dashboard



The image shows a mockup of an Admin Dashboard. At the top, there is a dark blue header bar with the text "Admin Dashboard" on the left and a "Logout" button on the right. Below the header, the main content area has a light gray background. On the left side of the main area, there is a white sidebar titled "Quick Menu" containing a list of links: "Pre Evaluation", "Upload Data", "Role Manager", "Reminder Mail", "Delete Data", and "Settings". To the right of the sidebar, the main content area features a "Welcome, imrohitvk" message. Below this, there is a "Pre Evaluation" section. This section contains a paragraph: "This is the content for the Pre Evaluation section. You can display evaluation details here." followed by two input fields. The first is labeled "Number of Questions:" and the second is labeled "Number of Total Marks:". Both fields have placeholder text: "Enter the number of questions" and "Enter total marks" respectively. A blue "Submit" button is located below the second input field.

1. Structure Overview

HTML File Structure

- **<head> Section:**
 - Includes Tailwind CSS for styling.
 - Inline JavaScript for interactive components.
 - **<body> Section:**
 - **Header:** Contains the navigation menu and logout functionality.
 - **Sidebar:** Provides a quick access menu.
 - **Main Content:** Dynamic sections for various administrative tasks.
-

2. Functional Components

2.1 Header

- Displays the title: "Admin Dashboard."
 - **Navigation Menu:**
 - Links to core sections: Pre Evaluation, Upload Data, Role Manager, etc.
 - Styled using Tailwind utility classes.
 - **Logout Button:**
 - Redirects to `/logout/`.
-

2.2 Sidebar

- Visible only on larger screens.
 - Provides a quick menu for navigation.
 - Dynamically renders Django messages (`{% for message in messages %}`) to display feedback.
-

2.3 Main Content

The content area dynamically updates based on the selected menu option. Each section is implemented as a hidden `<div>` that becomes visible when selected.

3. Content Sections

3.1 Upload Data

- **Purpose:** Enables administrators to upload CSV and PDF files.
- **Features:**
 - CSV Upload: Posts to `/uploadCSV/`.
 - Multiple PDF Upload: Posts to `/AdminHome/`.
 - File validation (CSV and PDF).
- **Forms:**
 - Include `{% csrf_token %}` for security.

3.2 Role Manager

- **Purpose:** Allows administrators to manage user roles.
- **Features:**
 - Form fields:
 - Username (text input).
 - Role (dropdown with options: Admin, Teacher, TA, Student).
 - Posts to [/change_role/](#) for backend processing.

3.3 Pre Evaluation

- **Purpose:** Set up pre-evaluation parameters.
- **Features:**
 - Inputs for the number of questions and total marks.
 - Posts data to [/questionNumbers/](#).

3.4 Reminder Mail

- **Purpose:** Send reminder emails to users.
- **Features:**
 - Posts to [/send_email/](#) via a simple form.
 - Includes a prominently styled button.

3.5 Delete Data

- **Purpose:** Delete uploaded data from the system.
 - **Features:**
 - Posts to [/DeleteDocs/](#).
 - Includes a red, visually distinct delete button.
-

3.6 Settings

- **Profile Section:**
 - Displays user information:
 - Username ({{ users.username }}).
 - Role ({{ users.role }}).
 - **Reset Password:**
 - Password validation:
 - Minimum 8 characters.
 - Must include one letter, one number, and one special character.
 - Password confirmation field.
 - Posts to [/changePassword/](#).
 - Includes JavaScript validation for:
 - Matching passwords.
 - Enabling/disabling the submit button.
-

4. Interactivity

Menu Navigation

- **Purpose:** Switch between sections dynamically.
- **Implementation:**
 - JavaScript toggles the visibility of content sections using [data-content](#) attributes.

Accordion Toggle

- **Purpose:** Expand/collapse settings subsections.
 - **Implementation:**
 - Toggles [max-height](#) for smooth transitions.
 - Adjusts icon rotation to provide visual feedback.
-

5. Integration with Django

Dynamic Rendering

- The dashboard uses Django template variables to inject dynamic data:
 - [{{ users.username }}](#): Displays the logged-in admin's username.

- `{{ users.role }}`: Displays the logged-in admin's role.
- `{% for message in messages %}`: Renders feedback messages.

CSRF Protection

- All forms include `{% csrf_token %}` to ensure security.
-

6. External Libraries

Tailwind CSS

- Utility-first CSS framework for layout and styling.
 - Ensures responsive design across devices.
-

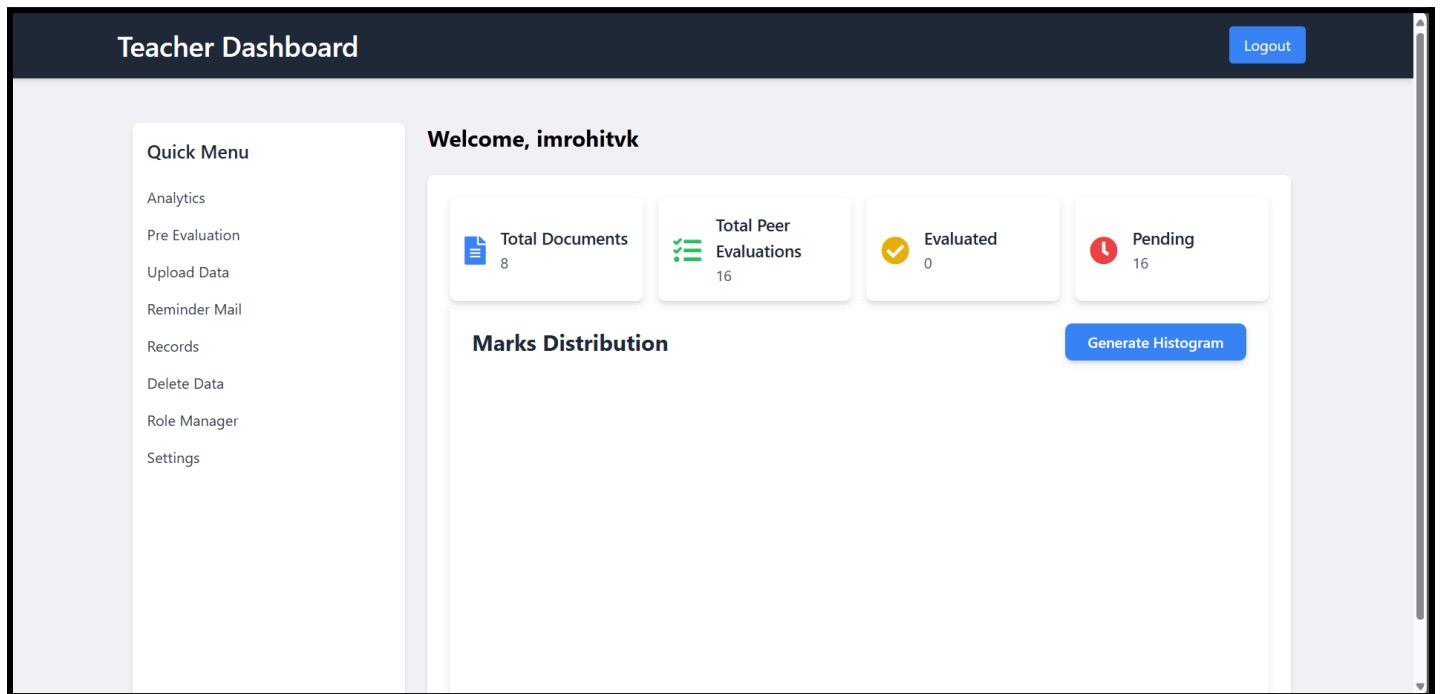
7. Deployment Notes

- **Backend Dependencies:**
 - Ensure corresponding backend routes (`/uploadCSV/`, `/change_role/`, etc.) are implemented and secured.
 - **Email Configuration:**
 - Reminder mail functionality requires SMTP settings.
 - **File Upload:**
 - Verify server permissions for uploading and handling files.
-

8. Maintenance Tips

- **Code Updates:**
 - Regularly test JavaScript interactivity and CSS styling.
 - **Security:**
 - Validate input data on both frontend and backend.
 - **Feedback:**
 - Use Django's messaging framework for improved UX.
-

Teacher Dashboard



1. Structure Overview

HTML File Structure

- **<head> Section:** Includes external CSS and JS libraries:
 - **Tailwind CSS:** Used for styling.
 - **Font Awesome:** Provides icons.
 - **Chart.js:** Renders data visualizations.
- **<body> Section:** Main structure of the dashboard.

2. Functional Components

Header

- Contains the title ("Teacher Dashboard") and navigation menu.
- Navigation buttons switch between sections (Pre Evaluation, Upload Data, Role Manager, etc.).
- Includes a logout button.

Sidebar

- Displays a quick navigation menu (visible only on larger screens).
 - Lists shortcuts to major features like Analytics, Pre Evaluation, etc.
 - Displays messages using Django's [messages](#) framework.
-

3. Content Sections

3.1 Upload Data

- Allows uploading:
 - **CSV Files:** Form posts to [/uploadCSV/](#).
 - **PDF Files:** Form posts to [/TeacherHome/](#).
 - Includes:
 - Form validation.
 - File type restrictions (CSV and PDF only).
-

3.2 Analytics

- Displays key statistics:
 - Total documents.
 - Total peer evaluations.
 - Evaluated and pending evaluations.
 - **Histogram Generator:**
 - Users input maximum marks.
 - Divides scores into quartiles.
 - Visualizes the distribution of scores using a bar chart.
-

3.3 Role Manager

- Allows admins to assign roles (Teacher, TA, Student) to users.
 - Posts data to [/change_role/](#) for processing.
 - Includes input validation for username and role.
-

3.4 Pre Evaluation

- Facilitates setting up pre-evaluation tasks:
 - Inputs for the number of questions and total marks.
 - Posts data to </questionNumbers/>.
-

3.5 Reminder Mail

- Sends reminder emails via a form posting to /send_email/.
-

3.6 Delete Data

- Provides an interface to delete uploaded data.
 - Posts data to </DeleteDocs/>.
-

3.7 Settings

- **Profile Section:**
 - Displays the username and role.
 - **Password Reset:**
 - Includes form validation:
 - Minimum 8 characters.
 - At least one letter, one number, and one special character.
 - Password confirmation.
-

4. Interactivity and Scripting

JavaScript Features

- **Menu Navigation:**
 - Hides/shows content sections when a menu item is clicked.
- **Accordion Toggle:**
 - Expands or collapses sections in the settings area.
- **Histogram Generator:**
 - Dynamically calculates quartiles based on input.
 - Uses Chart.js to render bar charts.

Validation

- Includes basic client-side validation for file uploads and password resets.
-

5. Django Integration

Template Features

- **Dynamic Data Rendering:**
 - Uses Django's template syntax (`{{ }}`) to populate data.
 - Example:
 - User info: `{{ users.username }}`, `{{ users.role }}`.
 - Analytics data: `{{ analytics_data.total_documents }}`.
 - **CSRF Protection:**
 - All forms include `{% csrf_token %}`.
-

6. External Libraries

Dependencies

- **Tailwind CSS:** Provides utility-first CSS classes.
 - **Font Awesome:** Adds icons for improved UX.
 - **Chart.js:** Creates charts for visual data representation.
-

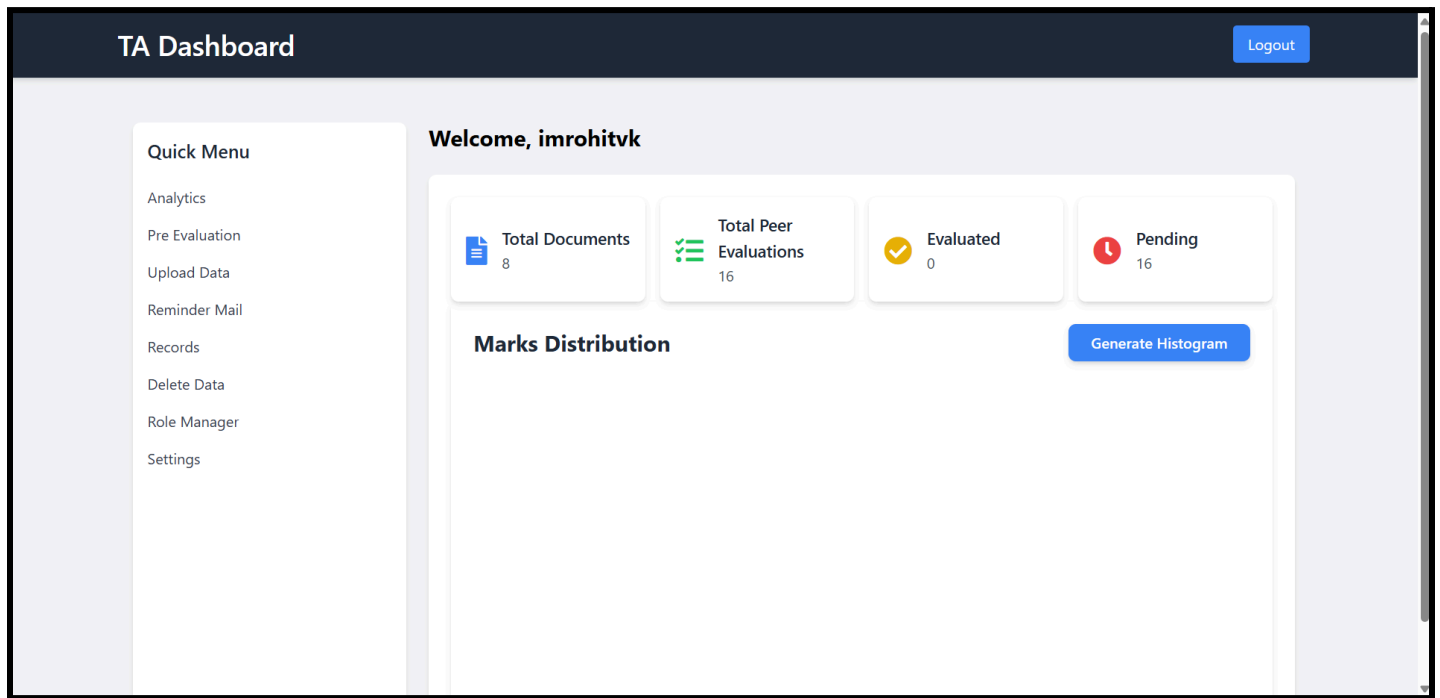
7. Deployment Notes

- **Ensure File Upload Permissions:**
 - Verify that the server allows file uploads (e.g., PDFs, CSVs).
 - **Database Integration:**
 - The analytics and role manager sections depend on backend data.
 - **Email Configuration:**
 - Reminder mail functionality requires proper SMTP settings.
-

8. Maintenance Tips

- Keep external libraries up to date.
 - Regularly test all interactive features (file uploads, analytics, etc.).
 - Ensure consistent styling across devices with responsive design.
-

TA Dashboard



1. Structure Overview

HTML File Structure

- **<head> Section:** Includes external libraries:
 - **Tailwind CSS:** Provides styling.
 - Inline scripts for custom interactivity.
- **<body> Section:** Main structure, comprising:
 - **Header:** Contains the navigation bar and logout button.
 - **Sidebar:** Optional menu visible on larger screens.
 - **Main Content:** Dynamic sections for various dashboard functionalities.

2. Functional Components

Header

- **Title:** Displays "TA Dashboard."

- **Navigation Menu:** Links to key sections (e.g., Pre Evaluation, Upload Data, etc.).
 - **Logout Button:** Redirects to the logout page.
-

Sidebar

- Available only on larger screens.
 - Provides quick navigation links.
 - Displays messages passed via Django's `messages` framework.
-

Main Content

The main content area displays dynamically based on user interaction with the navigation menu.

3. Content Sections

3.1 Upload Data

- **Purpose:** Allows TAs to upload data files.
 - **Features:**
 - Upload CSV files (form posts to `/uploadCSV/`).
 - Upload multiple PDF files (form posts to `/TAHome/`).
 - File validations:
 - CSV: Only single file upload.
 - PDF: Multiple file uploads allowed.
 - User prompts with guidelines for file types and sizes.
-

3.2 Role Manager

- **Purpose:** Manage roles of other users.
- **Features:**
 - Assign roles (TA, Student).
 - Form posts to `/change_role/`.
 - Input fields:
 - Username: Required.
 - Role: Dropdown menu with predefined options.

3.3 Pre Evaluation

- **Purpose:** Set up evaluation parameters.
 - **Features:**
 - Input total questions and total marks.
 - Form posts to [/questionNumbers/](#).
-

3.4 Reminder Mail

- **Purpose:** Send reminder emails to users.
 - **Features:**
 - Posts to [/send_email/](#).
 - Includes a single button for triggering reminders.
-

3.5 Delete Data

- **Purpose:** Allows TAs to delete uploaded data.
 - **Features:**
 - Posts to [/DeleteDocs/](#).
 - Includes a prominent delete button styled in red.
-

3.6 Settings

- **Purpose:** Manage profile and password settings.
 - **Features:**
 - **Profile Section:**
 - Displays username and role using Django template variables (`{{ users.username }}` and `{{ users.role }}`).
 - **Reset Password:**
 - **Validations:**
 - Minimum 8 characters.
 - Must include one letter, one number, and one special character.
 - Confirmation field ensures matching passwords.
 - Posts to [/changePassword/](#).
-

4. Interactivity

Menu Navigation

- **Purpose:** Dynamically show/hide content sections.
- **Implementation:**
 - JavaScript toggles the visibility of sections based on the clicked navigation item.

Accordion

- **Purpose:** Expand or collapse settings subsections (e.g., Profile, Reset Password).
 - **Implementation:**
 - Toggles max-height of the content area for smooth transitions.
 - Adds/removes a rotation class to icons for visual feedback.
-

5. Integration with Django

Template Variables

- The dashboard dynamically renders user-specific content using Django template variables:
 - **User Information:** `{{ users.username }}`, `{{ users.role }}`.
 - **Messages:** Rendered using `{% for message in messages %}`.

CSRF Protection

- All forms include `{% csrf_token %}` for security.
-

6. External Libraries

Tailwind CSS

- Utilized for styling and responsiveness.

Font Awesome (Optional)

- Can be added for icons.
-

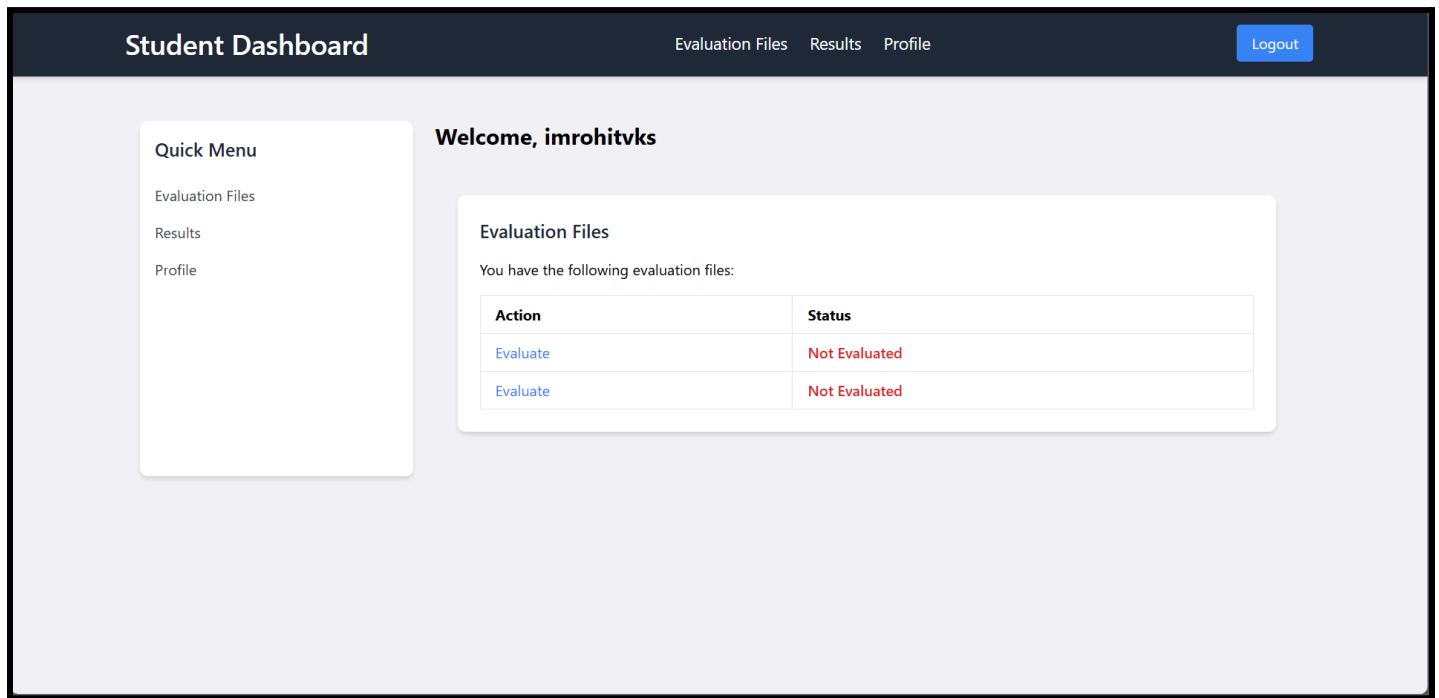
7. Deployment Notes

- **File Permissions:** Ensure the server allows file uploads.
 - **Email Configuration:** Reminder mail functionality requires proper SMTP settings.
 - **Backend Dependencies:** Verify Django routes ([/uploadCSV/](#), [/change_role/](#), etc.) are implemented.
-

8. Maintenance Tips

- Test navigation and form submissions regularly.
- Validate file uploads for compliance with accepted formats.
- Update Tailwind CSS as needed to leverage new features.

Student Dashboard



1. Structure Overview

HTML File Structure

- **<head> Section:**
 - Tailwind CSS for responsive and modern styling.
 - Chart.js for creating visual data representations.
- **<body> Section:**
 - Contains:
 - Header for navigation and logout functionality.
 - Sidebar for quick navigation.
 - Main content sections, which dynamically render based on user interaction.

2. Functional Components

2.1 Header

- **Title:** Displays "Student Dashboard."
 - **Navigation Menu:**
 - Links to [Evaluation Files](#), [Results](#), and [Profile](#) sections.
 - Styled with Tailwind classes for hover effects.
 - **Logout Button:**
 - Redirects to </logout/>.
-

2.2 Sidebar

- **Purpose:**
 - Provides quick navigation for larger screens.
 - Mirrors the navigation links in the header.
 - **Messages Section:**
 - Uses Django's `{% for message in messages %}` loop to display system messages.
 - Includes functionality to dismiss messages with a close button.
-

2.3 Main Content

Content dynamically switches based on user interaction with the navigation menu.

3. Content Sections

3.1 Evaluation Files

- **Purpose:**
 - Displays a list of evaluation files assigned to the student.
 - **Features:**
 - Table with:
 - **Action:** Link to evaluate the file.
 - **Status:** Indicates whether the file has been evaluated.
 - If no files are assigned, displays a fallback message: "No evaluation files assigned yet."
-

3.2 Results

- **Purpose:**
 - Shows the student's evaluation results.
 - **Features:**
 - **Download Answer Sheet:** Button to download the student's answer sheet (PDF).
 - **Summary Table:**
 - Displays peer evaluation feedback, scores, and aggregate marks.
 - Highlights aggregate score in bold for better visibility.
 - **Fallback Message:**
 - Displays "No peer reviews available for your documents" if there are no reviews.
-

3.3 Settings

- **Profile Section:**
 - Displays:
 - Username (`{{ request.user.username }}`).
 - Role: Always "Student."
 - **Reset Password:**
 - Allows the student to reset their password with the following validations:
 - Minimum 8 characters.
 - Must include one letter, one number, and one special character.
 - Password confirmation must match.
 - Posts data to `/changePassword/`.
-

4. Interactivity

Menu Navigation

- **Purpose:** Dynamically switches between sections based on user input.
- **Implementation:**
 - JavaScript toggles visibility of content sections using `data-content` attributes.

Accordion Toggle

- **Purpose:** Expands or collapses the Profile and Reset Password sections in the settings.
- **Implementation:**
 - Toggles `max-height` for smooth transitions.
 - Adjusts the rotation of the icons for visual feedback.

Chart.js Integration

- **Purpose:** Provides visual representation of aggregate marks.
 - **Implementation:**
 - A bar chart is rendered using Chart.js, with:
 - Labels for students.
 - Data representing aggregate marks.
 - Styled with a blue color palette.
-

5. Integration with Django

Dynamic Rendering

- The dashboard integrates with Django's template engine:
 - Displays dynamic data using variables like:
 - `{{ request.user.username }}` for the logged-in student.
 - `{% for file in evaluation_files %}` to iterate over assigned files.
 - `{% for document in own_documents %}` to render evaluation results.
 - Conditional rendering ensures proper fallback messages.

CSRF Protection

- All forms include `{% csrf_token %}` for enhanced security.
-

6. External Libraries

Tailwind CSS

- Utility-first CSS framework for modern styling.
- Ensures responsiveness across devices.

Chart.js

- Used for rendering bar charts in the results section.
-

7. Deployment Notes

Backend Dependencies

- Ensure backend endpoints (`/logout/`, `/changePassword/`, etc.) are correctly implemented.
- Use Django views to populate:
 - `evaluation_files` for the Evaluation Files section.
 - `own_documents` and `own_pdf` for the Results section.

File Access

- Verify that students have the necessary permissions to access their files and download resources.
-

8. Maintenance Tips

Code Updates

- Regularly test all JavaScript functionalities (menu toggles, accordion, and charts).
- Keep external libraries (Tailwind CSS, Chart.js) up-to-date.

User Experience

- Continuously improve the dashboard based on student feedback.