Frontend Documentation for Academix Portal

Introduction

This documentation provides an overview and guidelines for the frontend part of the Academix Portal. It aims to assist developers, designers, and readers in understanding and working with the frontend codebase.

Codebase Overview

Directory Structure :
Templates: Contains HTML templates for the frontend.
Static: Stores static assets like CSS, JavaScript, and images.

Frontend Architecture:

1. Layout and Structure:

Base Template: A core layout (like base.html) is often used as a foundation, ensuring consistency across different pages.
Navigation: A common navigation bar (e.g., navbar.html) is used for seamless navigation across the portal.
Specific Templates: Templates for specific functionalities (like student_list.html, student_profile.html) extend the base layout, adding content and features relevant to each page.

2. Styling and Responsiveness:

CSS Integration: Use of CSS files for styling, with specific stylesheets for different pages and a common stylesheet (stylish.css) for overall design consistency.
Responsive Design: Bootstrap or similar frameworks ensure that the portal is accessible and usable across a variety of devices and screen sizes.

3. User Interaction and Dynamic Content:

Forms and Inputs: Pages like login_page_admin.html and reply_query.html contain forms that capture user input, which is then sent to the backend.
JavaScript: Although not directly mentioned, JavaScript or similar scripting languages are used for dynamic content manipulation, client-side validation, and enhancing user interaction.

4. Template Rendering:

Django Templating: The use of Django template language has dynamic rendering of content based on data passed from the backend.

Interaction with the Backend:

1. Data Handling:

APIs and Endpoints: The frontend interacts with the backend through API calls. The backend exposes various endpoints for data retrieval and manipulation (like fetching student lists, submitting queries, etc.).
Form Submission: Actions like logging in, registering, or submitting assignments involve sending data from frontend forms to backend servers for processing.

2. Authentication and Authorization:

Secure Sessions: Login mechanisms (admin and student) involve creating secure sessions. The backend handles authentication and maintains session integrity.
Access Control: The backend also manages access control, determining what data and functionalities are available to different user types (students, faculty, admins).

3. Server-Side Processing:

Data Storage and Retrieval: All data modifications and retrievals initiated from the frontend are processed by the backend, which interacts with the AWS database
Business Logic: The backend handles the business logic of the application, ensuring that all data processing aligns with the portal's rules and logic.

UI COMPONENTS

1. Navigation Bar
Found in files like navbar.html.
Typically includes links to main sections of the portal, like courses, profile, and logout.

2. Forms
Seen in files such as login_page_admin.html, login_page_student.html, register.html.
Used for capturing user input for login, registration, feedback submission, etc.
Includes text fields, password fields, submission buttons, and potentially validation messages.

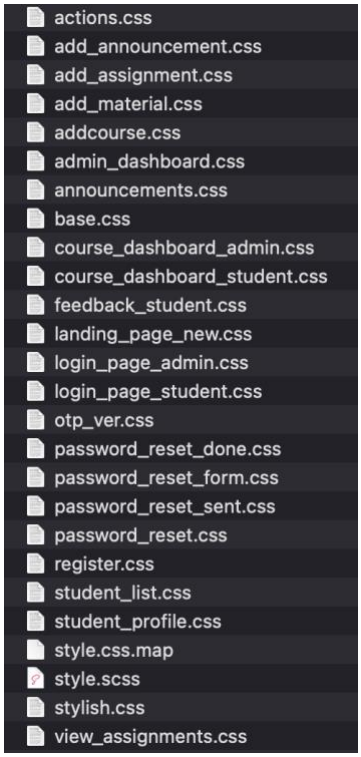
3. Lists and Tables
In files like student_list.html, view_assignments.html.
Used to display lists of students, courses, or assignments.

4. Cards
Used in student_profile.html, view_other_student_profile.html.
Displays information like student details or course summaries in a concise and visually appealing format.

5. Modals and Pop-Ups
Used for showing additional information or forms without leaving the current page.
Often used for confirmations, warnings, or additional data entry.

Static Files

CSS



Images
Templates
Structure and usage of HTML templates.

Coding Standards
HTML: Best practices and standards for writing HTML.
CSS: Standards for CSS, including responsive design principles.

Key Files

admin.py Documentation - Academix Portal

Overview
The admin.py file in Django is used for configuring the Django admin interface. This interface is a powerful and auto-generated tool that Django provides for managing database records.

In the Academix Portal, this file is used to make various models accessible and manageable through the Django admin panel.

Configured Models
The admin.py file registers several models with the Django admin. Here's a breakdown:

student_profile: This model likely contains information about student profiles. By registering it, you can view and manage student profiles through the admin interface.

faculty_profile: Similar to student_profile, this model is for faculty profiles, allowing admin users to manage faculty-related information.

Course: Represents courses available or conducted in the portal. This includes course details, instructors, schedules, etc.

Assignment: Manages assignments within the portal. This includes assignment descriptions, deadlines, and submissions.

Submission: Related to the Assignment model, this tracks students' submission of assignments.

Announcements: Used for managing announcement for any courses , faculty sided use..

feedback: Manages feedback from student to a faculty in a particular course

query: Handles queries or questions raised by users of the portal.
Material: Pertains to educational materials or resources associated with courses.
Models , the Django admin site will allow administrators to perform CRUD (Create, Read, Update, Delete) operations on these models directly from the web interface. This is highly useful for managing the data without needing to interact directly with the database or create separate views for these operations.

Accessing the Admin Panel
The admin panel can be accessed at the /admin URL path in the Django project localhost address.

apps.py Documentation - Academix Portal
Overview
The apps.py file in Django is used for application configuration — specifically, it defines settings and configurations for a Django app within a project. In the Academix Portal, this file contains configuration for an app named Academix_Portal.

Configuration Details
AcademixPortalConfig Class
Purpose: This class, AcademixPortalConfig, extends AppConfig from Django's app framework. It is used to set various configurations related to the Academix_Portal app.

default_auto_field:
Set to 'django.db.models.BigAutoField'.
This setting specifies the type of primary key to use for models in this app if no explicit primary key is defined.

name: The name of the app is set as 'Academix_Portal'.
This is the formal name used by Django to identify the app within the project.

Usage
The AcademixPortalConfig class is primarily used by Django's internal mechanisms to setup app-specific configurations. It is referenced in the project's settings, typically in the INSTALLED_APPS list, to include the app in the project

Documentation for Key Python Files in Academix Portal

1. models.py
Overview
Defines the data models for the Academix Portal. These models are used to structure the database schema and handle data manipulation.

Key Models
student_profile: Represents student profiles with fields like name, batch, and branch.
faculty_profile: Similar to student profiles, but for faculty members.
Course, Assignment, Submission, Announcements, Material, feedback, query: Other models representing different functionalities within the portal, like course management, assignments, feedback, etc.

Usage
These models are integral for CRUD (Create, Read, Update, Delete) operations within the portal. They are used in views for data processing and are likely referenced in templates for data display.

2. urls.py
Overview
Manages URL routing for the Academix Portal application.

Key URL Patterns
Routes for home, login, registration, admin dashboard, and course-related pages.
Each path function maps a URL pattern to a corresponding view function.

Usage
This file is essential for navigating through the portal, as it defines how URLs correspond to different views and functionalities.

3. utils.py
Overview
Contains utility functions that are used across the application.

Key Functions
send_email_to_client: A function to send emails, used for features like email verification via use of OTP.
Usage
Such utility functions are called in views or other parts of the application to perform common tasks, ensuring code reusability and separation of concerns.

4. views.py
Overview
Contains the business logic and handles requests and responses for the Academix Portal.

Key Views
Functions to render pages like home, login, registration, admin dashboard, etc.
Functions interacts with the models to fetch, process, and send data to templates.
Usage
Views are crucial for processing user requests and returning appropriate responses. They form the backbone of the application's functionality.

Detailed Frontend CSS Documentation for Academix Portal:

1. register.css

Key Styles

.whole:
This class applies a background image to the entire registration page.
The background-size: cover; ensures that the background image covers the entire area of the element, adjusting the image size as needed.
.card:
The opacity: 0.75; style makes elements with this class slightly transparent, creating a modern, stylish effect. This is applied to containers or cards on the registration page to overlay the background image slightly.

2. student_list.css

Key Styles

body:

The body style uses a linear gradient as the background. This gradient is to transition from white to white, creating a subtle effect.
This styling sets the tone for the student list page, aiming for a clean and minimalistic look.

3. student_profile.css

Key Styles

body:
Sets a very light (almost white) background color, creating a neutral canvas for the content with some lining patterns over it.

.padding:
Adds significant padding (3rem) to elements, enhancing spacing and readability.

.user-card-full, .card:
These classes are used for styling profile cards, including shadows (box-shadow) for depth and rounded borders (border-radius).This is a modern, card-based layout for displaying student profiles.

4. style.scss

Key Styles

Font Imports:
Imports specific Google Fonts ('Oswald' and 'Open Sans') in various weights, indicating a tailored typographic approach.
Color Gradients ($gr-1, $gr-2, $gr-3):

Defines three distinct linear gradients, used as background elements or highlights across the site.

Utility Classes (.h-100 & .align-middle):
.h-100 ensures an element takes up 100% of the viewport height.
.align-middle vertically centers elements, useful for layout and positioning.

5. stylish.css

Key Styles

Background Styles (.gr-2, .gr-3):
.gr-2 sets a background image, possibly for a specific section or element.
.gr-3 applies a linear gradient, similar to one defined in style.scss, suggesting thematic consistency.
Utility Classes:
Similar to style.scss, these classes provide quick layout and alignment tools.

6. view_assignments.css

Key Styles

body:
Uses a gradient background, giving the page a distinct visual style.

7. course_dashboard_student.css

body:

Implements a subtle, white gradient background. This choice suggests an plan for a clean and minimalistic interface, The gradient, although subtle, adds depth to the page, enhancing the user experience.
.backcolor:
Likely used for specific sections within the dashboard, this class adds a layer of visual segmentation, aiding in content organization.

.nav:
Consistent with base.css, this reinforces the uniformity in navigation styling across the portal.

a:
White color for hyperlinks enhances visibility and contrasts well with the blue navigation background, ensuring user-friendly navigation.

.coursediv:
This class, with white text color, is used for textual content within the course dashboard, maintaining readability against darker backgrounds.

.navbar-toggler:

Detailed styling for the navbar toggler enhances user interaction, especially on mobile devices. The color and border styling are pragmatic choices for visibility.

8. feedback_student.css

body:
The dark gradient background sets a serious and focused tone for the feedback page, which could be intentional to emphasize the importance of feedback.
The gradient from a darker to a lighter shade can be designed to guide the user's focus towards specific elements or actions on the page.
Responsive Styles:

The use of media queries demonstrates a responsive design approach. Adjusting the max-width and min-width of form elements like textareas ensures that the feedback form is usable and accessible on devices of varying screen sizes.

9. login_page_admin.css
body:

The gradient background blends from light to medium shades, creating a welcoming yet professional atmosphere for the admin login page.
The choice of colors here is crucial as it sets the first impression for the administrative interface of the portal.

.container:
Utilizing flexbox to center content vertically and horizontally shows a modern CSS approach, ensuring the login form is the focal point of the page.
.card:
The card's styling with semi-transparent white background, padding, rounded corners, and subtle shadow gives it a modern, floating appearance. This design choice is intended to make the login process visually appealing and user-friendly.

10. login_page_student.css

The styling is very similar to login_page_admin.css, suggesting a design philosophy that values consistency across different user roles while maintaining distinct experiences for students and administrators. Any minor differences would be pivotal in subtly differentiating the student login experience from the admin's.

11. otp_ver.css

body:
The darker gradient background for the OTP verification page could be an intentional design choice to convey a sense of security and focus, as OTP verification is a critical security step. The gradient effect can also create a visual cue for the user, directing attention towards the verification form.
.container and .card:
Similar to the login pages, these styles provide a consistent user experience. The card's distinct background color (#757575) and styling indicate a special focus on the OTP verification process.

12. add_announcement.css

Detailed Analysis
form:

Specifies minimum and maximum widths, padding, margin, and background, indicating a focus on form layout and aesthetics.
The semi-transparent white background (#ffffff59) with box-shadow creates a floating effect, enhancing the visual appeal and focus on the form.
form input:

Inputs have a subtle border, padding, and color styling, ensuring usability and aesthetic consistency.
form button:
Button styling with a light background and border indicates a minimalist and clean design approach. The color and background changes on hover improve user interaction.'

13. add_assignment.css

having similar styling to add_announcement.css, suggesting a consistent design language across different forms within the portal. This consistency in form styling enhances the user experience by providing a familiar interface across different functionalities.

14. add_material.css
Also shares styling with add_announcement.css, reinforcing the uniform design approach across various administrative tasks such as adding announcements, assignments, and materials.

15. addcourse.css
form:
The form has distinct width, padding, and background settings, tailored for the course addition interface.
The choice of a lighter background (rgba(223, 223, 223, 0.838)) differentiates it from other forms, signifying the importance or frequency of use.

form input and button:
Similar styling as in other forms maintains design consistency. The hover effect on the button is a thoughtful addition for enhancing user interaction.

16. admin_dashboard.css
body:
Sets the text color, alignment, and background color for the admin dashboard. The chosen colors suggest a professional and clean interface.
.card:
Card elements have a defined style with border-radius and box-shadow, suggesting a modern, modular design approach for displaying content.

17. announcements.css
.nav:
Consistent with other CSS files, it sets the navigation background color, maintaining the thematic consistency across the portal.
form:
Specific styling for announcement forms with different background color (#efefef) and border settings. This slight variation in styling could be to visually distinguish the announcement form from others.
form input and button:
The hover effect on the button adds a dynamic interaction element for the user.

Best Practices and Recommendations

HTML

1. add_announcement.html

Structure:
Contains a form for creating announcements, ncluding input fields for title, description, and perhaps date and having dynamic content rendering.
Design:
Styling from both announcements-specific CSS and a general stylish.css file combination of customized and standard styles.
Bootstrap's inclusion means a grid layout and responsive design elements like buttons and input fields.

2. add_assignment.html

Structure:
Similar to the announcement addition page, this includes fields for assignment title, description, deadline, and file upload options.
The form action and method would be set to handle assignment data submission.
Design:
The consistent use of a specific CSS file for announcements and assignments suggests a unified aesthetic across these functionalities.

3. add_feedback.html

Structure:
A feedback form, possibly with text areas for detailed feedback, and a submission button.
May include hidden fields or dynamically populated data for context-specific feedback (e.g., course or faculty feedback).
Design:
The use of a separate CSS file for adding announcements suggests customized styling for feedback forms.

4. add_material.html

Structure:
A form tailored for adding educational materials, which might include file uploads or links to resources.
Could include categorization options like course association, material type (video, document, etc.).
Design:
Follows the portal's design language, ensuring that the material upload interface is intuitive and aligned with other forms.

5. add_query.html

Structure:
Designed for submitting queries, with input fields for the query subject and details.
also include user identification fields, either visible or hidden, for personalized query handling.
Design:
The form's design is likely straightforward and user-centric, focusing on ease of query submission.

6. add_submission.html

Structure:
Contains fields for students to submit their work, possibly through file uploads.
Specific form actions to direct the data to the appropriate server-side handlers.
Design:
The layout and design would prioritize clarity and simplicity, as assignment submission is a critical functionality.

7. addcourse.html

Structure:
A detailed form for course creation, including fields for course name, description, instructor details.
Integration with server-side logic for course data processing.
Design:
Given its significance, this page likely has a more comprehensive layout with clear sectioning and input validation cues.

8. announcements.html

Structure:
Extending from a navbar template, this file is structured to list announcements, possibly in a chronological or priority-based order.
includes dynamic content loading for continuous update of announcements.
Design:
The in-file style definitions suggest custom styling for specific announcement elements, such as highlighting new or important announcements.

9. assignments.html

Structure:
Designed to list assignments, with each item potentially displaying the title, due date, and status (completed, pending).
Interactive elements like buttons or links to view or submit assignments.
Design:

The use of course dashboard-specific CSS indicates tailored styling, possibly to facilitate easy navigation and status checking of assignments.

10. base.html

Purpose: Serves as the base template for other pages in the Academix Portal, providing a consistent layout and style.
Key Features:
Includes a meta viewport tag for responsive design.
Links to CSS files (stylish.css and Bootstrap) for consistent styling across the portal.
Can contain common elements like header, footer, and navigation bar.

11. course_dashboard_student.html

Purpose: Displays the course dashboard for students.
Structure:
Extends base.html, inheriting its base structure.
Specific links to CSS files (my_course_student.css) for tailored styling of the dashboard.
Mostly contains a list or grid of courses, along with links to view and unenroll course..

12. edit_assignment.html

Purpose: Provides an interface for editing assignments.
Key Elements:
Form elements for editing various aspects of an assignment, such as title, description, due date, etc.
Utilizes Bootstrap for layout and styling, ensuring a consistent and responsive user interface.

13. editcourse.html

Purpose: Used for editing course details.
Structure:
Similar to edit_assignment.html, but tailored for course-specific details.
Includes fields for course name, description, instructor information, and scheduling.

14. feedback_faculty.html

Purpose: Enables faculty members to view and manage feedback.
Layout:
Included is a list or table displaying feedback entries, possibly with options to filter, sort, or respond to feedback.
Utilizes Bootstrap for a responsive design and may include custom CSS for specific styling needs.

15. feedback_student.html

Purpose: Allows students to submit feedback.
Key Features:
Form for submitting feedback, including text fields for detailed responses.
Inherits styling from navbar.html, indicating a consistent top navigation across student-facing pages.

16. feedback.html

Purpose: A general feedback view, possibly used for both students and faculty.
Structure:
Could be a combination of feedback_student.html and feedback_faculty.html, providing a unified interface for viewing and submitting feedback.
Inclusion of Bootstrap and custom CSS files for responsive and specific styling.

17. home.html

Purpose: The homepage of the Academix Portal.
Content:
Likely contains a welcome message, links to main sections of the portal, and possibly a newsfeed or announcements section.
The form element suggests functionality for course enrollment or login, with a conditional display based on user authentication status.

18. land.html

Purpose: Serves as a landing or home page for the Academix Portal.
Structure:
Includes responsive design elements (meta viewport tag).
Utilizes Bootstrap and a custom stylesheet for styling.
Contains introductory content, navigation links, or a login prompt.

19. login_page_admin.html

Purpose: Admin login interface for the Academix Portal.
Key Elements:
Meta tags for character set and viewport configuration.
Includes Bootstrap and a specific CSS file for login page styling.
Contains a form for admin login credentials (username/password).

20. login_page_student.html

Purpose: Student login interface.
Structure:
Similar to the admin login page, ensuring consistency in user experience.

Contains fields for student authentication, possibly using student-specific credentials.

21. materials.html

Purpose: Displays educational materials or resources.
Key Features:
Inherits styling from navbar.html, indicating a consistent navigation experience.

22. my_course_student.html

Purpose: Student's personal course dashboard.
Layout:
Extends the base layout, ensuring uniformity across the portal.
Likely organized to display courses enrolled , materials and assignments.

23. navbar.html

Purpose: Provides a common navigation bar across multiple pages.
Structure:
Contains meta tags and links to global stylesheets.
The navigation bar includes links to key sections of the portal, user profile access, and logout functionality.

24. otp_ver.html

Purpose: One-Time Password (OTP) verification page.
Elements:
Utilizes specific CSS for OTP verification styling.
Includes Bootstrap for responsive design.
Contains form elements for OTP entry, ensuring secure access.

25. register.html

Purpose: Registration page for new users.
Design:
Incorporates Bootstrap and a specific CSS file for registration form styling.
The form includes fields for personal information necessary for user registration.

26. reply_query.html

Purpose: Interface for responding to student queries.
Key Elements:
Inherits common navigation and styling from navbar.html.
Specific styling for the query response interface, possibly including input areas for detailed replies.

27. student_list.html

Purpose: Displays a list of students, potentially for administrative or faculty use.
Layout and Design:
Extends navbar.html for consistent navigation.
Utilizes student_list.css for specific styling, organizing student data in a table or list format.

28. student_profile.html

Purpose: Displays individual student profiles.
Structure:
Inherits base layout from base.html.
Includes sections for personal information and other relevant details.
Custom CSS for card-like design elements, indicating a visually appealing presentation of profile data.

29. view_assignments_faculty.html

Purpose: Allows faculty to view and manage assignments.
Content and Functionality:
Inherits common elements from navbar.html.
Features a comprehensive list of assignments, possibly with options to edit, grade, or view submissions.
Custom styling to facilitate easy navigation and interaction with assignment data.

30. view_assignments.html

Purpose: For students to view assignments.
User Interface:
Similar in structure to view_assignments_faculty.html but tailored for student use.
Includes features like submission status, due dates, and links to assignment details.
Ensures a user-friendly and informative layout for students to track their assignments.
31. view_other_student_profile.html

Purpose: Enables viewing profiles of other students for faculty
Design:
Similar to student_profile.html but adjusted for viewing by peers.
Might restrict certain personal information for privacy.

32. view_query.html

Purpose: Interface for viewing user queries.
Features:
Extends navbar.html for consistent navigation.
Customized layout to display queries in an organized manner.
Can include quick response features or links to detailed response interfaces.

33.. view_students_submission.html

Purpose: Allows faculty or faculty or admin to view students' assignment submissions.
Layout and Interactivity:
Inherits styling from navbar.html.
Organizes submissions in a list or table, with features to view, grade, or provide feedback.
Custom styling might highlight key information like submission dates, grades, or completion status.

Each of these HTML files plays an important role in the user interface and experience of the Academix Portal. They are designed to provide a seamless and intuitive interaction for users, whether it's for accessing course materials, managing personal information, or performing administrative tasks. Consistent use of Bootstrap across these templates ensures a responsive and cohesive design, while the use of Django template language enables dynamic content rendering, contributing to an efficient and user-friendly educational portal.