Statement - " Front end UI needed to be created using python, Needed Modules to be installed, which should be integratable with any other python code coming to UI it should have user textbox on left and response textbox on right side with Title of 'Edu Assist: AI - Powered Personalized Learning Companion'" Provide solution to above statement using Django Framework

**ChatGPT said:**

ChatGPT

Creating a front-end UI using Python with the Django framework involves several steps. Here’s a solution to implement the requested UI:

**Solution Overview**

We will:

1. Create a Django project and app.
2. Set up the necessary URL routes.
3. Design an HTML template with the required UI structure.
4. Add logic to handle user input and provide responses.

**Step 1: Install Required Modules**

Install Django if not already installed:

bash

Copy code

pip install django

**Step 2: Create a Django Project and App**

Run the following commands to create a Django project and an app:

bash

Copy code

django-admin startproject edu\_assist

cd edu\_assist

python manage.py startapp assistant

**Step 3: Configure the Project**

Add the app (assistant) to INSTALLED\_APPS in edu\_assist/settings.py:

python

Copy code

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'assistant',

]

**Step 4: Define URL Patterns**

In edu\_assist/urls.py, include the app's URLs:

python

Copy code

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('assistant.urls')),

]

Create assistant/urls.py:

python

Copy code

from django.urls import path

from . import views

urlpatterns = [

path('', views.index, name='index'),

]

**Step 5: Create the View Logic**

In assistant/views.py, define the index view:

python

Copy code

from django.shortcuts import render

def index(request):

user\_input = ''

ai\_response = ''

if request.method == 'POST':

user\_input = request.POST.get('user\_textbox')

# Simulate an AI response (replace with real logic as needed)

ai\_response = f"AI Response to: {user\_input}"

return render(request, 'assistant/index.html', {

'user\_input': user\_input,

'ai\_response': ai\_response

})

**Step 6: Create the HTML Template**

In assistant/templates/assistant/index.html, create the UI:

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edu Assist: AI-Powered Personalized Learning Companion</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.container {

display: flex;

flex-direction: column;

align-items: center;

}

.textbox-container {

display: flex;

justify-content: space-between;

width: 80%;

}

.textbox {

width: 45%;

height: 200px;

font-size: 16px;

}

button {

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

}

</style>

</head>

<body>

<div class="container">

<h1>Edu Assist: AI-Powered Personalized Learning Companion</h1>

<form method="POST">

{% csrf\_token %}

<div class="textbox-container">

<textarea name="user\_textbox" class="textbox" placeholder="Enter your query here...">{{ user\_input }}</textarea>

<textarea readonly class="textbox" placeholder="AI's response will appear here...">{{ ai\_response }}</textarea>

</div>

<button type="submit">Submit</button>

</form>

</div>

</body>

</html>

**Step 7: Run the Application**

Run the development server:

bash

Copy code

python manage.py runserver

Visit http://127.0.0.1:8000/ to see the UI.

**Summary**

* The UI has a title, a user textbox on the left, and a response textbox on the right.
* It uses Django to handle form submissions and process user input.
* The response logic can be integrated with any AI model or API. Replace the placeholder logic in views.py with your AI backend code.

4o

**You said:**

Explain the structure to be create Django project.

**ChatGPT said:**

ChatGPT

When you create a Django project, it follows a specific directory and file structure. Below is an explanation of the structure you would create when setting up your Django project edu\_assist (as in the previous example):

**Directory Structure of a Django Project**

plaintext

Copy code

edu\_assist/ # Root project folder

├── edu\_assist/ # Main project configuration directory

│ ├── \_\_init\_\_.py # Makes this directory a Python package

│ ├── settings.py # Global project settings (database, installed apps, etc.)

│ ├── urls.py # Main URL configuration (routing)

│ ├── asgi.py # ASGI application entry point (for asynchronous servers)

│ ├── wsgi.py # WSGI application entry point (for production servers)

├── assistant/ # Application directory (your custom app)

│ ├── migrations/ # Database migrations (auto-generated files)

│ │ ├── \_\_init\_\_.py

│ ├── \_\_init\_\_.py # Makes this directory a Python package

│ ├── admin.py # Admin site configuration

│ ├── apps.py # App configuration file

│ ├── models.py # Define data models (tables in the database)

│ ├── tests.py # Unit tests for the app

│ ├── views.py # View functions to handle requests and responses

│ ├── urls.py # URL configuration specific to the app

│ └── templates/ # HTML templates (for rendering views)

│ └── assistant/

│ └── index.html

├── manage.py # Command-line utility for Django tasks (runserver, migrations, etc.)

**Explanation of Each Component**

**1. Root Project Folder (edu\_assist/)**

This is the top-level directory of your project. It contains:

* The edu\_assist/ directory with project configuration files.
* The manage.py file for command-line interactions.

**2. Project Configuration Directory (edu\_assist/)**

This contains files that configure the project as a whole:

* **\_\_init\_\_.py:** An empty file indicating that this directory is a Python package.
* **settings.py:** The main settings file for the project, where you configure:
  + Installed apps.
  + Middleware.
  + Database settings.
  + Static and media files.
* **urls.py:** The main URL routing configuration. Routes URLs to specific app views.
* **asgi.py and wsgi.py:** Entry points for ASGI and WSGI servers. Required for deploying your project.

**3. App Directory (assistant/)**

This is the directory for the custom app you create (in this case, assistant). It contains:

* **models.py:** Define database tables and their structure using Django ORM.
* **views.py:** Define view functions or class-based views to handle HTTP requests and responses.
* **urls.py:** Define app-specific URL routes, typically included in the project's urls.py.
* **admin.py:** Configure the Django admin interface for models in this app.
* **apps.py:** Metadata for the app (e.g., name, configurations).
* **templates/:** A directory for HTML templates used by this app. The convention is to name the subfolder after the app (assistant/ in this case) to avoid name conflicts.

**4. Migrations Directory (assistant/migrations/)**

This folder contains migration files that Django generates to keep your database schema in sync with your models. Each migration represents a change in your database schema.

**5. manage.py**

A command-line utility script for managing the Django project. Common commands include:

* python manage.py runserver: Start the development server.
* python manage.py makemigrations: Create migration files.
* python manage.py migrate: Apply migrations to the database.
* python manage.py createsuperuser: Create an admin user.

**How the Structure Supports Development**

1. **Modular Design:** Apps like assistant allow for reusable and independent components within the project.
2. **Separation of Concerns:** The structure separates logic into models (data), views (business logic), and templates (presentation).
3. **Scalability:** You can add more apps as your project grows (e.g., auth, dashboard).
4. **Maintainability:** Organized files and directories make it easy to locate and modify specific parts of the code.

**Key Commands to Create and Navigate This Structure**

1. **Create a Project:**

bash

Copy code

django-admin startproject edu\_assist

1. **Navigate to Project Directory:**

bash

Copy code

cd edu\_assist

1. **Create an App:**

bash

Copy code

python manage.py startapp assistant

1. **Run the Development Server:**

bash

Copy code

python manage.py runserver

1. **Generate and Apply Migrations:**

bash

Copy code

python manage.py makemigrations

python manage.py migrate

This structure ensures Django projects are well-organized, easy to scale, and follow best practices.

4o

**You said:**

TemplateDoesNotExist at / assistant/index.html Request Method: GET Request URL: http://127.0.0.1:8000/ Django Version: 5.0.1 Exception Type: TemplateDoesNotExist Exception Value: assistant/index.html Exception Location: C:\Users\HI\AppData\Local\Programs\Python\Python312\Lib\site-packages\django\template\loader.py, line 19, in get\_template Raised during: assistant.views.index Python Executable: C:\Users\HI\AppData\Local\Programs\Python\Python312\python.exe Python Version: 3.12.1 Python Path: ['C:\\Users\\HI\\OneDrive\\Desktop\\Django version\\edu\_assist', 'C:\\Users\\HI\\AppData\\Local\\Programs\\Python\\Python312\\python312.zip', 'C:\\Users\\HI\\AppData\\Local\\Programs\\Python\\Python312\\DLLs', 'C:\\Users\\HI\\AppData\\Local\\Programs\\Python\\Python312\\Lib', 'C:\\Users\\HI\\AppData\\Local\\Programs\\Python\\Python312', 'C:\\Users\\HI\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages'] Server time: Mon, 25 Nov 2024 17:57:48 +0000

**ChatGPT said:**

ChatGPT

The error TemplateDoesNotExist indicates that Django cannot find the specified template (assistant/index.html). Here are the steps to resolve this issue:

**1. Ensure Correct Template Directory Structure**

Make sure your template file exists in the correct location. For the app assistant, the template file should be located at:

bash

Copy code

assistant/templates/assistant/index.html

This means:

* **assistant/** is your app directory.
* Inside assistant/, there should be a **templates/** directory.
* Inside templates/, there should be a subdirectory named **assistant/** (matching your app's name).
* Inside this subdirectory, the file **index.html** should exist.

**2. Verify Template Settings in settings.py**

Ensure the TEMPLATES setting in your settings.py is configured to look for templates in app directories. It should look like this:

python

Copy code

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [], # Can add additional template directories here if needed

'APP\_DIRS': True, # Ensure this is set to True

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

The key here is:

* APP\_DIRS: True tells Django to search for templates inside the templates directory of each app.

**3. Check the File Location**

Navigate to the directory where your index.html file is stored and verify its path. If your project is in C:\Users\HI\OneDrive\Desktop\Django version\edu\_assist, the full path to index.html should be:

makefile

Copy code

C:\Users\HI\OneDrive\Desktop\Django version\edu\_assist\assistant\templates\assistant\index.html

**4. Debugging Steps**

* **Check File Extension:** Ensure the file is named index.html (not index.html.txt).
* **Check File Contents:** Open the file and confirm it has valid HTML content.

**5. Run the Server Again**

After confirming the structure and settings:

bash

Copy code

python manage.py runserver

**Example Directory Structure**

plaintext

Copy code

edu\_assist/

├── edu\_assist/

│ ├── settings.py

│ ├── urls.py

│ ├── ...

├── assistant/

│ ├── templates/

│ │ ├── assistant/

│ │ │ ├── index.html

│ ├── views.py

│ ├── urls.py

│ ├── ...

After verifying the structure, the template should render correctly. If the issue persists, let me know!

4o

**You said:**

Can you change the style the page with any colorful template apt to the chatbot and decrease the text box

**ChatGPT said:**

ChatGPT

Here's an updated version of the template for your chatbot UI with a colorful and modern design. This template uses a **vibrant color palette** and smaller textboxes to make the interface more visually appealing.

**Updated index.html Template**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edu Assist: AI-Powered Personalized Learning Companion</title>

<style>

/\* General Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Arial', sans-serif;

background: linear-gradient(to right, #4facfe, #00f2fe);

color: #333;

padding: 20px;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.chatbot-container {

width: 400px;

background-color: #ffffff;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

border-radius: 10px;

overflow: hidden;

padding: 20px;

}

.chatbot-header {

background-color: #00c9a7;

color: #fff;

padding: 10px;

text-align: center;

font-size: 20px;

font-weight: bold;

border-radius: 10px 10px 0 0;

}

.textbox-container {

margin-top: 20px;

display: flex;

flex-direction: column;

gap: 10px;

}

.textbox {

width: 100%;

padding: 10px;

font-size: 14px;

border: 1px solid #ddd;

border-radius: 5px;

resize: none;

height: 80px;

}

.textbox:focus {

border-color: #00c9a7;

outline: none;

box-shadow: 0 0 5px rgba(0, 201, 167, 0.5);

}

button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 5px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #00a986;

}

</style>

</head>

<body>

<div class="chatbot-container">

<div class="chatbot-header">

Edu Assist: AI-Powered Personalized Learning Companion

</div>

<form method="POST">

{% csrf\_token %}

<div class="textbox-container">

<!-- User Input -->

<textarea name="user\_textbox" class="textbox" placeholder="Type your query here...">{{ user\_input }}</textarea>

<!-- AI Response -->

<textarea readonly class="textbox" placeholder="AI's response will appear here...">{{ ai\_response }}</textarea>

</div>

<button type="submit">Ask AI</button>

</form>

</div>

</body>

</html>

**Key Features of the Updated Design**

1. **Colorful Background:**
   * A gradient background (#4facfe to #00f2fe) gives a vibrant and modern look.
2. **Compact Textboxes:**
   * Reduced height of textboxes to 80px for a more compact design.
   * Disabled resizing to maintain consistent UI.
3. **Themed Header:**
   * A header with a bold green shade (#00c9a7) to grab attention.
4. **Interactive Button:**
   * A submit button with hover effects for better user interaction.
5. **Responsive Layout:**
   * The design adapts to different screen sizes with padding and alignment.

**How It Looks**

The page will have:

* A **centered chatbot UI** on a colorful gradient background.
* A **minimalist card design** for the chatbot container.
* Small, functional textboxes for user queries and AI responses.

You can paste this template into your index.html file under assistant/templates/assistant/. Then restart the server and refresh the page to see the updated UI.

Let me know if you need further customizations!

2/2

4o

**You said:**

Can you make the title outside the Query Box

**ChatGPT said:**

ChatGPT

Here’s the updated design where the **title** (Edu Assist: AI-Powered Personalized Learning Companion) is placed **outside the chatbot container** for better separation. The title now sits prominently at the top of the page, while the query box remains below in a visually distinct section.

**Updated index.html Template**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edu Assist: AI-Powered Personalized Learning Companion</title>

<style>

/\* General Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Arial', sans-serif;

background: linear-gradient(to right, #4facfe, #00f2fe);

color: #333;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

}

.title {

font-size: 24px;

font-weight: bold;

color: #ffffff;

margin-bottom: 20px;

text-align: center;

}

.chatbot-container {

width: 400px;

background-color: #ffffff;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

border-radius: 10px;

overflow: hidden;

padding: 20px;

}

.textbox-container {

margin-top: 10px;

display: flex;

flex-direction: column;

gap: 10px;

}

.textbox {

width: 100%;

padding: 10px;

font-size: 14px;

border: 1px solid #ddd;

border-radius: 5px;

resize: none;

height: 80px;

}

.textbox:focus {

border-color: #00c9a7;

outline: none;

box-shadow: 0 0 5px rgba(0, 201, 167, 0.5);

}

button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 5px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #00a986;

}

</style>

</head>

<body>

<!-- Title Outside the Chatbot Box -->

<div class="title">Edu Assist: AI-Powered Personalized Learning Companion</div>

<!-- Chatbot UI -->

<div class="chatbot-container">

<form method="POST">

{% csrf\_token %}

<div class="textbox-container">

<!-- User Input -->

<textarea name="user\_textbox" class="textbox" placeholder="Type your query here...">{{ user\_input }}</textarea>

<!-- AI Response -->

<textarea readonly class="textbox" placeholder="AI's response will appear here...">{{ ai\_response }}</textarea>

</div>

<button type="submit">Ask AI</button>

</form>

</div>

</body>

</html>

**Key Changes**

1. **Title Placement:**
   * The title (Edu Assist: AI-Powered Personalized Learning Companion) is now outside the chatbot container and styled separately.
2. **Enhanced Title Styling:**
   * A bold, white font with larger size ensures it stands out against the gradient background.
3. **Chatbot Box Starts Below:**
   * The query box is distinct and positioned below the title.
4. **Consistent Layout:**
   * The title and chatbot UI remain aligned in the center of the page.

**How It Looks**

* **Top Section:** The title is prominently displayed at the top.
* **Bottom Section:** The chatbot UI (query box, response box, and button) is clearly separated and neatly designed.

Place this updated code in your index.html file, restart your server, and refresh the page to see the changes. Let me know if you'd like further tweaks!

4o

**You said:**

Can you add some more buttons where functionality can be added later buttons needed to be added are 1. Chatbot 2. Quizzes 3. Alarm all the buttons Should be on left side of the Screen and buttons to be in Rectangle with rounded edges. On clicking the Chatbot it should be show now present screen when I click Quizzes it should show Welcome to Quizzes and Choose the difficulty when I click Alarm it should show Alarm as title and Show current time on and option to set alarm

**ChatGPT said:**

ChatGPT

Here's the updated code incorporating your requirements. The screen now has **buttons on the left side** for navigation (Chatbot, Quizzes, Alarm), each of which dynamically updates the main content. The buttons are styled as rectangular with rounded edges.

**Updated index.html**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edu Assist</title>

<style>

/\* General Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Arial', sans-serif;

background: linear-gradient(to right, #4facfe, #00f2fe);

color: #333;

height: 100vh;

display: flex;

}

.sidebar {

width: 200px;

background-color: #ffffff;

box-shadow: 2px 0 5px rgba(0, 0, 0, 0.2);

padding: 20px;

display: flex;

flex-direction: column;

gap: 20px;

}

.sidebar button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 10px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

}

.sidebar button:hover {

background-color: #00a986;

}

.content {

flex: 1;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

}

.content h1 {

font-size: 24px;

color: #ffffff;

margin-bottom: 20px;

}

.chatbot-container {

width: 400px;

background-color: #ffffff;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

border-radius: 10px;

padding: 20px;

}

.textbox-container {

margin-top: 10px;

display: flex;

flex-direction: column;

gap: 10px;

}

.textbox {

width: 100%;

padding: 10px;

font-size: 14px;

border: 1px solid #ddd;

border-radius: 5px;

resize: none;

height: 80px;

}

.textbox:focus {

border-color: #00c9a7;

outline: none;

box-shadow: 0 0 5px rgba(0, 201, 167, 0.5);

}

button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 5px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #00a986;

}

.hidden {

display: none;

}

</style>

<script>

function showScreen(screen) {

// Hide all sections

document.querySelectorAll('.section').forEach(section => section.classList.add('hidden'));

// Show selected section

document.getElementById(screen).classList.remove('hidden');

}

function updateCurrentTime() {

const now = new Date();

document.getElementById('current-time').textContent = now.toLocaleTimeString();

}

setInterval(updateCurrentTime, 1000); // Update time every second

</script>

</head>

<body>

<!-- Sidebar -->

<div class="sidebar">

<button onclick="showScreen('chatbot')">Chatbot</button>

<button onclick="showScreen('quizzes')">Quizzes</button>

<button onclick="showScreen('alarm')">Alarm</button>

</div>

<!-- Main Content -->

<div class="content">

<!-- Chatbot Screen -->

<div id="chatbot" class="section">

<h1>Edu Assist: AI-Powered Personalized Learning Companion</h1>

<div class="chatbot-container">

<form method="POST">

{% csrf\_token %}

<div class="textbox-container">

<!-- User Input -->

<textarea name="user\_textbox" class="textbox" placeholder="Type your query here...">{{ user\_input }}</textarea>

<!-- AI Response -->

<textarea readonly class="textbox" placeholder="AI's response will appear here...">{{ ai\_response }}</textarea>

</div>

<button type="submit">Ask AI</button>

</form>

</div>

</div>

<!-- Quizzes Screen -->

<div id="quizzes" class="section hidden">

<h1>Welcome to Quizzes</h1>

<p>Choose the difficulty:</p>

<button>Easy</button>

<button>Medium</button>

<button>Hard</button>

</div>

<!-- Alarm Screen -->

<div id="alarm" class="section hidden">

<h1>Alarm</h1>

<p>Current Time: <span id="current-time"></span></p>

<label for="alarm-time">Set Alarm Time:</label>

<input type="time" id="alarm-time">

<button onclick="alert('Alarm set successfully!')">Set Alarm</button>

</div>

</div>

</body>

</html>

**Key Features**

1. **Sidebar Navigation:**
   * Buttons for Chatbot, Quizzes, and Alarm on the left side.
   * Styled with a rectangular shape and rounded edges.
2. **Dynamic Content Display:**
   * Clicking a button shows the corresponding section (Chatbot, Quizzes, or Alarm) while hiding the others using JavaScript.
3. **Individual Screen Designs:**
   * **Chatbot:** Existing chatbot interface.
   * **Quizzes:** Simple placeholder with buttons for difficulty selection (Easy, Medium, Hard).
   * **Alarm:** Displays the current time (auto-updating) and provides an input for setting the alarm.
4. **Responsive Layout:**
   * Sidebar is fixed on the left.
   * Main content dynamically updates based on user interaction.

**How It Works**

1. Clicking a **button in the sidebar** triggers the showScreen function.
2. The function hides all sections and displays the one corresponding to the button.
3. The **current time** on the Alarm screen updates every second using JavaScript.

Let me know if you’d like to add specific functionality to the Quizzes or Alarm sections!

4o

**You said:**

Add the following changes to UI 1.Title( Edu Assist: AI-Powered Personalized Learning Companion ) Should be Common for every screen. 2.For Chatbot screen it should Show title as Chatbot. 3.Add spacing between textbox of AI Response and Ask AI in Chatbot Screen, In Quizzes Screen add spacing between text and buttons, in between buttons as well, In Alarm Screen Add spaing between Current time and Set Alarm, and add spacing between alarm and alarm set button. 4. Adjust the Individual Title at Center for every Screen

**ChatGPT said:**

ChatGPT

Here’s the updated code with the requested changes:

**Changes Made**

1. **Common Title**:
   * The main title, Edu Assist: AI-Powered Personalized Learning Companion, is now common for all screens.
2. **Screen-Specific Titles**:
   * Each screen displays its own title (Chatbot, Quizzes, or Alarm) below the common title, centered.
3. **Spacing Adjustments**:
   * Added consistent spacing between elements in all screens:
     + Chatbot: AI Response and Ask AI button.
     + Quizzes: Text and buttons, and between buttons.
     + Alarm: Current time, Set Alarm input, and button.
4. **Centered Screen Titles**:
   * Individual titles are now centered within their respective content areas.

**Updated index.html**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edu Assist</title>

<style>

/\* General Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Arial', sans-serif;

background: linear-gradient(to right, #4facfe, #00f2fe);

color: #333;

height: 100vh;

display: flex;

flex-direction: column;

}

.main-title {

font-size: 24px;

font-weight: bold;

color: #ffffff;

text-align: center;

margin: 20px 0;

}

.container {

display: flex;

flex: 1;

}

.sidebar {

width: 200px;

background-color: #ffffff;

box-shadow: 2px 0 5px rgba(0, 0, 0, 0.2);

padding: 20px;

display: flex;

flex-direction: column;

gap: 20px;

}

.sidebar button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 10px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

}

.sidebar button:hover {

background-color: #00a986;

}

.content {

flex: 1;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

justify-content: flex-start;

}

.section-title {

font-size: 20px;

color: #ffffff;

margin-bottom: 20px;

text-align: center;

}

.chatbot-container {

width: 400px;

background-color: #ffffff;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

border-radius: 10px;

padding: 20px;

}

.textbox-container {

margin-top: 10px;

display: flex;

flex-direction: column;

gap: 10px;

}

.textbox {

width: 100%;

padding: 10px;

font-size: 14px;

border: 1px solid #ddd;

border-radius: 5px;

resize: none;

height: 80px;

}

.textbox:focus {

border-color: #00c9a7;

outline: none;

box-shadow: 0 0 5px rgba(0, 201, 167, 0.5);

}

button {

background-color: #00c9a7;

color: white;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 5px;

padding: 10px;

cursor: pointer;

transition: background-color 0.3s ease;

margin-top: 20px; /\* Spacing below buttons \*/

}

.quizzes-container button {

margin: 10px 0; /\* Spacing between buttons \*/

}

.alarm-container p {

margin-bottom: 20px; /\* Spacing below Current Time \*/

}

.alarm-container input,

.alarm-container button {

margin-top: 10px; /\* Spacing between Set Alarm input and button \*/

}

.hidden {

display: none;

}

</style>

<script>

function showScreen(screen) {

// Hide all sections

document.querySelectorAll('.section').forEach(section => section.classList.add('hidden'));

// Show selected section

document.getElementById(screen).classList.remove('hidden');

}

function updateCurrentTime() {

const now = new Date();

document.getElementById('current-time').textContent = now.toLocaleTimeString();

}

setInterval(updateCurrentTime, 1000); // Update time every second

</script>

</head>

<body>

<!-- Main Title -->

<div class="main-title">Edu Assist: AI-Powered Personalized Learning Companion</div>

<!-- Layout -->

<div class="container">

<!-- Sidebar -->

<div class="sidebar">

<button onclick="showScreen('chatbot')">Chatbot</button>

<button onclick="showScreen('quizzes')">Quizzes</button>

<button onclick="showScreen('alarm')">Alarm</button>

</div>

<!-- Main Content -->

<div class="content">

<!-- Chatbot Screen -->

<div id="chatbot" class="section">

<h1 class="section-title">Chatbot</h1>

<div class="chatbot-container">

<form method="POST">

{% csrf\_token %}

<div class="textbox-container">

<!-- User Input -->

<textarea name="user\_textbox" class="textbox" placeholder="Type your query here...">{{ user\_input }}</textarea>

<!-- AI Response -->

<textarea readonly class="textbox" placeholder="AI's response will appear here...">{{ ai\_response }}</textarea>

</div>

<button type="submit">Ask AI</button>

</form>

</div>

</div>

<!-- Quizzes Screen -->

<div id="quizzes" class="section hidden">

<h1 class="section-title">Quizzes</h1>

<div class="quizzes-container">

<p>Choose the difficulty:</p>

<button>Easy</button>

<button>Medium</button>

<button>Hard</button>

</div>

</div>

<!-- Alarm Screen -->

<div id="alarm" class="section hidden">

<h1 class="section-title">Alarm</h1>

<div class="alarm-container">

<p>Current Time: <span id="current-time"></span></p>

<label for="alarm-time">Set Alarm Time:</label>

<input type="time" id="alarm-time">

<button onclick="alert('Alarm set successfully!')">Set Alarm</button>

</div>

</div>

</div>

</div>

</body>

</html>

**Key Updates**

1. **Common Title**:
   * A persistent title (Edu Assist: AI-Powered Personalized Learning Companion) at the top of every screen.
2. **Spacing Adjustments**:
   * Added appropriate margin between elements on all screens for better spacing.
3. **Centered Individual Titles**:
   * Each screen has its title (Chatbot, Quizzes, Alarm) centered for consistency.