Building a Robust Teacher Portal with Django, HTML & JavaScript

Overview:

You are tasked with creating a robust teacher portal with Django. The portal will include a login screen and a home screen for teachers, which will also serve as the student listing screen. Additionally, you will need to implement functionality to manage student listings and add new students.

Requirements:

1. Login Functionality:

- Create a login screen where teachers can input their credentials.
- Upon successful login, authenticate the user by verifying with a database.
- Handle authentication errors and provide appropriate feedback to the user.

2. Teacher Portal Home & Student Listing Screen:

- After successful login, redirect the user to the home screen of the teacher portal, which also serves as the student listing screen.
- Display a list of students with their Names, Subject Names, and Marks.
- Include options to edit and delete student details.
- Implement functionality to edit student details inline and update the state accordingly.

3. New Student Entry:

- Provide a feature for adding details of a new student using a popup/modal.
- When adding a new student:
- Check if a student with the same name and subject combination already exists in the database.
- If a matching record is found, update the marks for that student by adding the new marks to the existing ones.





If no matching record is found, create a new student record.

Technology Requirements:

Front-end (HTML & JavaScript):

- Use basic HTML and CSS for the views.
- Use vanilla JavaScript for front-end logics.

Back-End (Django):

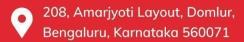
• Use Django for DB connection, controllers and helper functions.

Submission Guidelines:

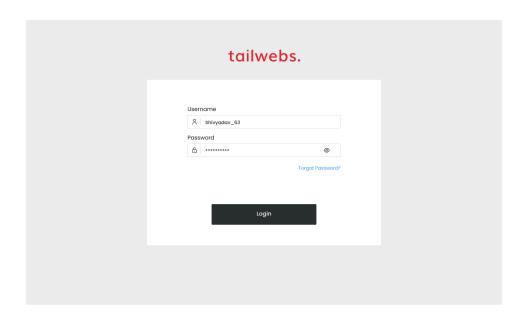
- Create a separate repository for submitting the project.
- Share the link to the repository once you have completed the task.
- Include clear instructions on how to run the Django project locally.
- Ensure the code is well-structured, documented, and follows best practices for both front-end and back-end development.
- Bonus points for implementing additional features or improvements beyond the basic requirements.

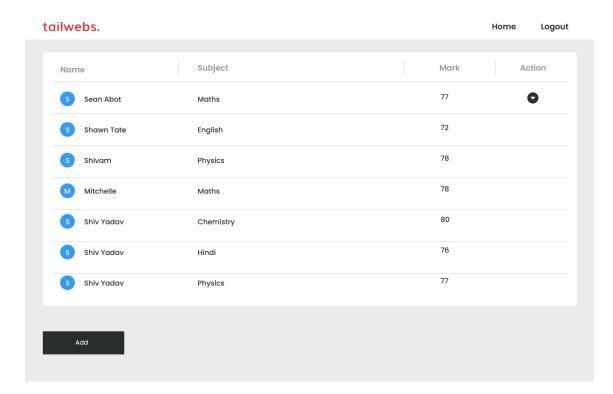
Additional Information

- Aim for a highly scalable and maintainable architecture that can accommodate future enhancements and changes.
- Pay attention to security best practices, including data encryption, input validation, and protection against common security vulnerabilities.
- Perform thorough testing, including unit tests, integration tests, and end-to-end tests, to ensure the reliability of the application.



Screenshots for reference





- www.tailwebs.com
- hello@tailwebs.com
- **(** +91-9590708339 / 7044074445

