**Report on the Student Management System Project**

1. **Introduction**

The **Student Management System** is a web-based application developed using the Django framework. It allows administrators to manage student information, assign grades, and handle other academic data efficiently.

This report outlines the setup process, challenges encountered during development, and how each feature was implemented. Key code snippets and screenshots are included to illustrate important aspects of the project.

1. **Setup Process**

2.1 Clone the Repository

git clone https://github.com/Mingyang096/StudentManagementSystem.git

cd StudentManagementSystem

* 1. Create a Virtual Environment

python -m venv student\_env

source student\_env/bin/activate

* 1. Install Dependencies

pip install -r requirements.txt

* 1. Apply Migrations

python manage.py migrate

* 1. Create a Superuser

python3 manage.py createsuperuser

* 1. Run the Development Server

python manage.py runserver

* 1. Access the Application

Open your web browser and navigate to:

http://127.0.0.1:8000/admin/

Use your admin credentials to log in. Then you can access the Django admin panel to manage students and other models.

1. **Challenges Encountered**

The biggest challenge I met was the input validation for grades. According to the requirement, the grade should be between 1 and 12. However, after I developed the code for grade input validation in student\_management/students/forms.py, it didn’t work. Users could still input numbers larger than 12 and save the forms.

To deal with this, I added a lot of prints to see what happens. Then I found out there was no output in my console. Then I added log settings to save the log to a log file. I get the conclusion: my grade input validation did not work.

文本

描述已自动生成

Then I searched for the internet to learn how to solve this problem. Till I found the MinValueValidator and MaxValueValidator classes. I added them to the students/models.py file so when the Grade field was set; it was restricted between 1 and 12. Finally, it worked.

文本, 应用程序, 聊天或短信

描述已自动生成

1. **Features Implementation**

**4.1 Student Management**

Student Management implemented Create, Read, Update, and Delete functionalities for managing student records.

**Models**: Defined a Student model in models.py with fields such as first\_name, last\_name, grade, etc.

文本, 聊天或短信

描述已自动生成

**Views**: Created views to handle each CRUD operation.

图形用户界面, 文本, 聊天或短信

描述已自动生成

**Templates**: Designed HTML templates for listing students, displaying details, and forms for adding/editing students.

* 1. **Admin Interface**

Utilized Django’s built-in admin interface to manage student data efficiently. The admin panel provides a user-friendly interface for administrators to perform management tasks without requiring custom front-end development for these functions.

**Admin Registration**: Registered the Student model in admin.py to make it accessible through the admin interface.

文本

描述已自动生成

* 1. **Forms**

Implemented forms to handle user input for creating and updating student records. Forms ensure that data is validated before being processed and saved to the database.

**Forms Module**: Created a StudentForm in forms.py based on the Student model.

**Validation**: Leveraged model validators defined in the Student model to ensure data integrity.

文本

描述已自动生成

**Templates**: Used Django’s template language to render forms in HTML templates.

* 1. **Authentication**

Secured the application by implementing user authentication, ensuring that only authorized users (administrators) can access certain views and perform management tasks.

**Django Authentication**: Used Django’s built-in authentication system.

**Login Required Decorator**: Applied @login\_required decorators to views that require authentication.

文本, 聊天或短信

描述已自动生成

* 1. **Search Functionality**

Added the ability for users to search for students based on certain criteria, such as first name, last name, or grade.

**Search Form**: Included a search form in the template.

截图里有图片

描述已自动生成

* 1. **Pagination**

Implemented pagination to manage the display of large numbers of student records by dividing them into pages.

**Paginator**: Used Django’s Paginator class in the view.

电脑屏幕的手机截图

描述已自动生成

* 1. **Error Handling and Validation**

Ensured that the application gracefully handles errors and validates user input to maintain data integrity and provide a better user experience.

**Model Validators**: Added validators to the Student model fields.

**Form Validation**: Relied on model validators in forms to automatically enforce validation rules.

**Error Messages in Templates**: Displayed validation errors in the templates.

图形用户界面, 文本

描述已自动生成