

# **Student Management System Using Django**

## **1. Introduction**

The Student Management System is a web-based application developed using the Django framework. The system provides basic CRUD (Create, Read, Update, and Delete) functionalities to manage student records. Each student record includes personal and academic information such as name, sex, address, course, and result. The system is designed to be simple, user-friendly, and suitable for academic institutions.

## **2. System Requirements**

### **2.1 Hardware Requirements**

- ✓ Computer with 4 GB RAM
- ✓ At least 500 GB free disk space

### **2.2 System Requirements and Functionality Environment**

The Student Management System was developed and tested under the following environment and software specifications to ensure smooth operation and compatibility:

#### **2.2.1. Operating System**

**Windows 10:** The system was deployed and tested on Windows 10, providing compatibility with the Python runtime and Django framework.

#### **2.2.2. Python**

**Python Version 3.6.2:** The system requires Python 3.6.2, which supports all Django 3.2.25 features and libraries used in this project, including virtual environments and package management through pip.

#### **2.2.3. Django**

**Django Version 3.2.25:** This version was used to implement the web-based Student Management System. It provides the MVT (Model–View–Template) architecture for managing models, views, templates, and URL routing.

#### **2.2.4. Web Browser**

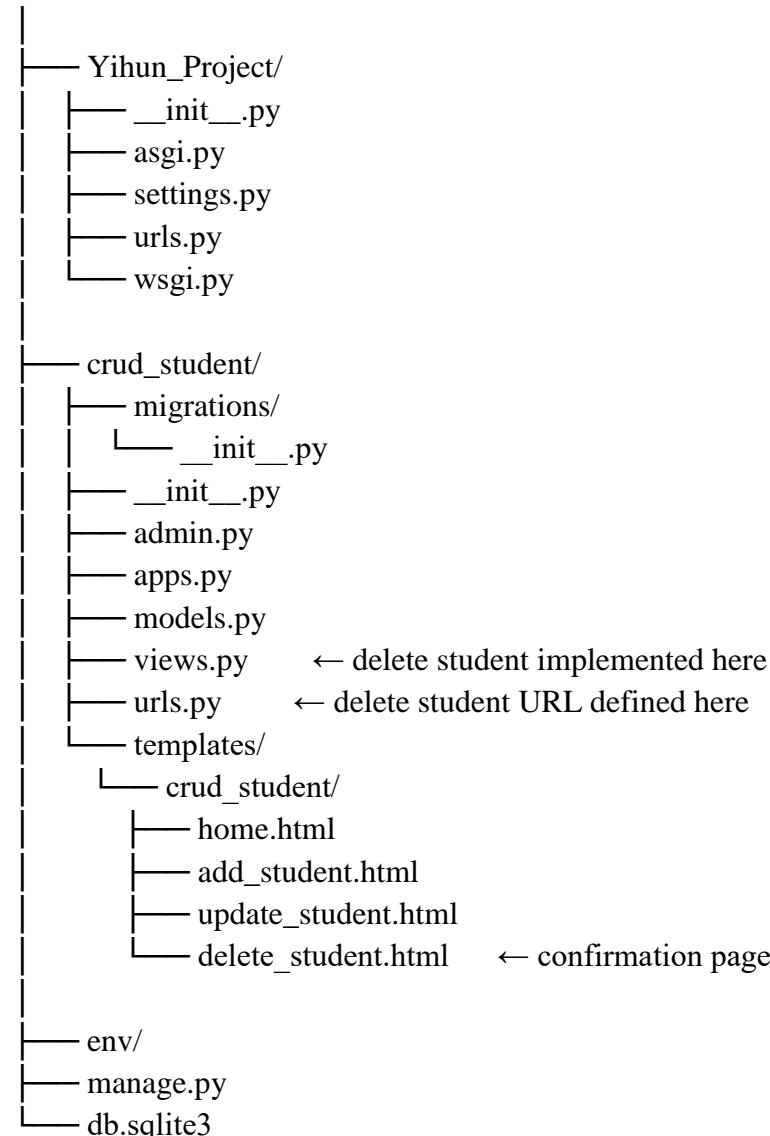
**Google Chrome or Equivalent:** The system is tested on modern web browsers that support HTML5, CSS3, and JavaScript for the front-end rendering of templates and interaction with the web application.

## 2.2.5. Code Editor

**Visual Studio Code (Recommended):** VS Code was used for writing, debugging, and managing project files. It offers integration with Python, Django extensions, and terminal commands to run the development server efficiently.

## 3. Project Structure

Yihun\_Project/



## 4. Navigate to Our Workspace

Open a command prompt and move to the directory where you want to create the project:

- ✓ cd C:\Users\A\Desktop\Yihun\ Yihun\_Project

This ensures all project files are organized in a dedicated folder. Keeping projects organized improves maintainability and reduces confusion when managing multiple projects.

## 5. Create a Virtual Environment

A virtual environment was created to isolate the project dependencies from the global Python installation. This approach ensures version consistency, prevents dependency conflicts, and improves project portability.

```
python -m venv env  
.env\Scripts\activate.bat
```

### Explanation:

- ✓ python -m venv env creates a virtual environment named env specifically for this project.
- ✓ .env\Scripts\activate.bat activates the environment on Windows.
- ✓ Using a virtual environment ensures project dependencies do not conflict with globally installed packages and allows version control of packages.

## 6. Install Django

After activating the virtual environment, the Django web framework was installed within the virtual environment:

```
pip install django==3.2.25
```

- ✓ Django version **3.2.25** was selected due to its stability and long-term support.
- ✓ The framework provides built-in features such as URL routing, ORM, template engine, and security mechanisms, which simplify web application development.

## 7. Creating the Django Project

This section describes the step-by-step process followed to initialize and prepare the Django project environment for the development of the Student Management System. Initialize the main project:

```
django-admin startproject Yihun_Project  
cd Yihun_Project
```

### Explanation:

- ✓ startproject generates the main project structure containing:
- ✓ The project includes essential configuration files such as:
  - settings.py → Global project settings configuration (database, installed apps, templates, etc.)

- urls.py → Root URL routing configuration
  - wsgi.py → Web Server Gateway deployment Interface for deployment
- ✓ The directory was entered using cd Yihun\_Project to continue development tasks.

## 8. Create the Application Module

A Django application was created to encapsulate all student management functionalities.

```
python manage.py startapp crud_student
```

### Explanation:

- ✓ The **crud\_student** app handles student data processing and presentation.
- ✓ This modular approach improves maintainability and supports system scalability.

## 9. Configure the Project

Project configuration ensures that Django recognizes the created application and properly manages static files, templates, and database interactions.

To enable the application within the project, it was registered in the INSTALLED\_APPS section of settings.py.

```
INSTALLED_APPS = [
```

```
    ...
    'crud_student',
]
```

Registering the app allows Django to detect models, templates, and migrations associated with the application.

## 10. Design the Database Model

The database layer was designed using Django's Object-Relational Mapping (ORM) system, which simplifies database interaction without writing raw SQL queries. Define the Student model in crud\_student/models.py:

## 11. Apply Database Migrations

After defining the model, database migrations were performed to translate the model structure into actual database tables. Generate and apply database migrations:

```
python manage.py makemigrations
```

```
python manage.py migrate
```

### Explanation:

- ✓ makemigrations generates migration files based on model definitions.
- ✓ migrate applies these changes to the database.
- ✓ This process ensures synchronization between the model and the database schema.

## 12: Configure URLs

URL configuration defines how HTTP requests are mapped to application views.

### 12.1 Application URL Configuration

The app-level URL routing was defined in crud\_student/urls.py.

```
from django.urls import path
from . import views
urlpatterns = [
    path("", views.home, name='home'),
    path('add/', views.add_student, name='add_student'),
    path('update/<int:id>', views.update_student, name='update_student'),
    path('delete/<int:id>', views.delete_student, name='delete_student'),
]
```

#### **Explanation:**

- ✓ Maps HTTP requests to specific view functions.
- ✓ <int:id> allows operations on specific student records.
- ✓ Named URLs (name='...') make referencing in templates and redirects easier.

### 12.2 Project URLs

Include the app URLs in Yihun\_Project/urls.py:

```
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('crud_student.urls')),
]
```

#### **Explanation:**

- ✓ The project URLs delegate routing to the app's URLs.
- ✓ Centralizes all URL configurations while keeping app-level URLs modular.

## 13. Run the Development Server

Start the Django server:

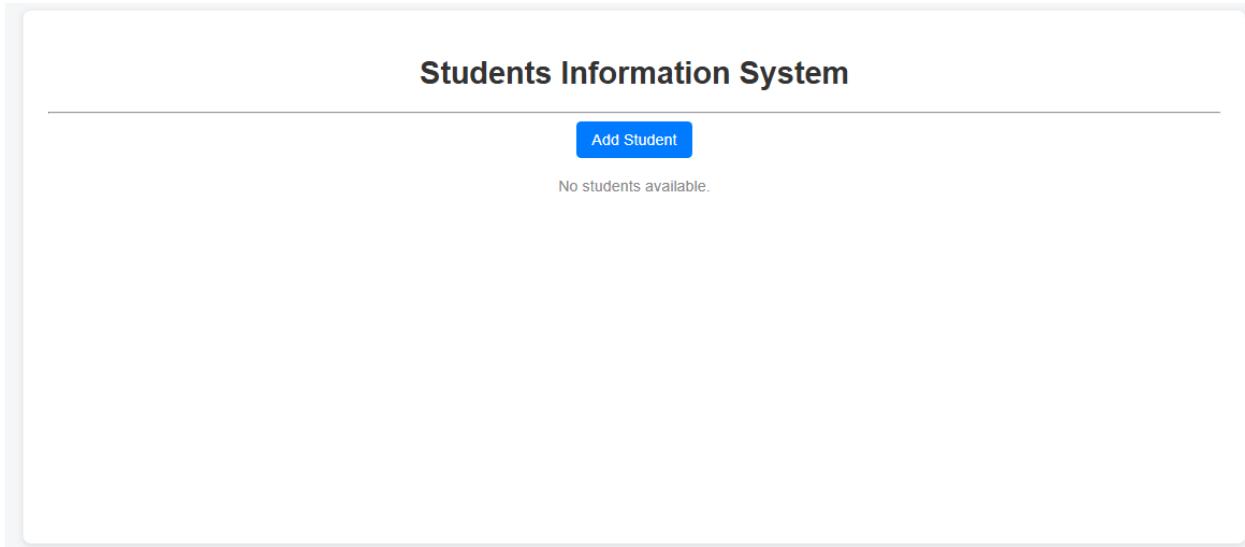
```
python manage.py runserver
```

### Explanation:

- ✓ The server runs locally at <http://127.0.0.1:8000/>.
- ✓ The system can now be tested in a web browser.

## Sample Working Interfaces

### Home Page:



Add Student Information Page:

## Add / Update Student Information

Name:

Sex:

 ▼

Address:

Course:

Result:

 ▼

Date of birth:

Email:

Phone:

[Save](#)

[Back](#)

## Add / Update Student Information

Name:

Yihun Tewachew

Sex:

Male



Address:

Gondar, Ethiopia

Course:

Machine Learning

Result:

A



Date of birth:

1998-12-03

Email:

yihuntewachew12@gmail.com

Phone:

0923419468

[Save](#)

[Back](#)

**View Student Information Page:**

### Students Information System

[Add Student](#)

Yihun Tewachew | M | Gondar, Ethiopia | Machine Learning | A

[Edit](#) | [Delete](#)

Aster Awoke | F | Bahirdar, Ethiopia | Web Based | B

[Edit](#) | [Delete](#)

**Delete Student Information Page:**

