

**WEB APPLICATION FRAMEWORKS**

**Assessment 1**

**Django Website Creation (School Management)**

Unit Code : COMP6006

Name : Syed Muhammad Ahmed Zaidi

Student ID: 20972008

# Information About the Project

**Project Name :** schoolmanagement  
**App Name :** courseapp

**Models :**

* Course
* Student

**Method :**

* total\_students
* \_\_str\_\_

**View Functions :**

* courses
* students
* singlecourse

## Admin Information

* **Username :** 20972008
* **Password :** 20972008

## Important URLs:

[*http://127.0.0.1:8000/schoolmanagement/courses*](http://127.0.0.1:8000/schoolmanagement/courses)

* To view the complete list of courses offered by the school

[*http://127.0.0.1:8000/schoolmanagement/students*](http://127.0.0.1:8000/schoolmanagement/students)

* To view the complete list of students enrolled in multiple Units

[*http://127.0.0.1:8000/schoolmanagement/courses*](http://127.0.0.1:8000/schoolmanagement/courses)*/1*

* To view the complete data of a single course.
* Changing the number will redirect towards complete data of other courses being offered.
* A hyperlink is also set up with the name of every course.

**Question 3 :**  In a Word document, describe why you chose the attributes and data types that you chose and why you implemented the relationship(s) between the two models in the way that you did. You may wish to contrast your choices with alternatives and consider the functionality enabled by your choices. (3)

**Answer :**

Considering the first model, which is the course model, the first attribute that I chose was named ‘name’. This is relatively easy to ease understanding of the user. It is as essential attribute as multiple courses are being offered and hence having this would allow to differentiate them. The datatype used for this was Charfield as it will be representing a textual data. A max length of 255 characters was set as the name of a person will always be of a limited character. An alternative approach would have been to use the ‘TextField’ however this would have used more storage space and may have adversely impacted the performance of database reading. The next attribute for this model was for the name of the Coordinator. For this, I used coordinator as it was easy and simple to understand. Alternatively, I could have used name\_coordinator. However, a longer name would have led to more chances of syntax errors. For the data type, the same Charfield was used due to the reasons mentioned above. The last attribute added in this model was the size. This was to show the maximum capacity of students that can be added in a particular course. For this the attribute of size was used. The data type used for this was ‘PositiveIntegerField’ this is because the number of students can only be an integer and would always be a positive number if not 0. An alternative approach could be to use ‘SmallIntegerField’ but as we are unsure about the school size, the course capacity may increase and due to which this data type may not be appropriate as it only has a capacity of small size limit.

For the second model called Student, there was a total of 5 attributes created. The first one being the name of the student which was represented by ‘name’. The data type is similar to what was used for the name of the course due to the same reasons mentioned above. The second is the email id for the students for which the attribute being used is ‘email’. Alternatively, ‘email\_id’ or ‘student\_email’ could also be used. However, I wanted to keep things short and simple and just used ‘email’. The data type used for this is the ‘EmailField’ which could be replaced with a simple charfield or textfield. However, ‘EmailField’ is unique in a way that it automatically validates email addresses and shows an error if the structure is not correct. Next is the student ID, for which the attribute is named student\_id. The data type being used is the ‘CharField’, as many of the students may have ids that starts with a character and not just integers. Next is the date of birth, for which the attribute is named a simple ‘date\_of\_birth’. An alternative approach could have been to use just the ‘DOB’; however, personally, I think using such short names may lead to writing errors that may be hard to detect. For this the datatype used was the ‘DateField’ this is to store the dates for the student’s respective births. The alternative was the ‘DateTimeField’; however, for this particular attribute, we do not require the time. Last is the information about the course the student is enrolled in for which a simple attribute of ‘course’ was used.

For the relationship between the two models, a many-to-one relationship was established where each student can enrol in only one course. The relationship was implemented using a ‘ForeignKey’ field in the student model which allows a reference towards the course model. A constraint was enforced on this that any student can only and exactly enrol in 1 course at a time. In addition to this, the on\_delete=models.CASCADE parameter was also included so that if any course is deleted, all students that are enrolled in that course should also get deleted. This was done to maintain consistency in the data. Overall, the functionality enabled by using such attributes, data types and relationships was to ensure there is efficient storage use and also improvement in the integrity and consistency of data so that the chances of errors can be kept to a minimum.

**Question 9:** In the Word document used for question three, detail which CRUD operations you would restrict to the course coordinator, students and website visitors respectively and justify your decisions. (3)

**Answer :** For Course Coordinators in particular, the operation of Create should be accessible because they are the ones who will oversee the complete curriculum, and hence, the ability to create data within the models is highly applicable. The operation of Read should also be accessible as they will be making decisions based on the course offerings and student enrolled; hence, they should be able to assess the complete data. They should also be allowed to Update information in the database, for example, a change in the course size. They should also have the authority to delete data for example any course that no longer being offered should be deleted to keep the database clean and efficient.

For Students, the operation of creating a record in the student model should be allowed as a new student would need to add their information in the system about their name, email, DOB, etc. They should also be able to read up information about the course and their own individual information as students. A restriction can be made to not allow reading information about other students. The Update category for courses should not be accessible; however, for their own individual information, it should open if incase any student makes an error and wants to correct personal information. The delete category should not be allowed as this could lead to inadvertent data loss or unauthorised modifications.

Website visitors may include general public reading about the school. Hence, considering the operation of Create, they should not be allowed to create any further information within the two models specified as it could lead to unauthorised additions which may compromise the integrity of the data. They should be able to read up about the course information; however, personal information about the students should not be accessible. Updating data should also not be allowed for any visitors as this may lead to security vulnerabilities. Lastly, the delete function should also not be permitted to any of the visitors as any accidental or malicious removal of course or student information may cause issues with the overall data. Restricting delete access helps maintain data consistency and protects against data loss.

**Question 10:** discuss two of the benefits of implementing this App within Django, with respect to either security, performance or ease of development. You may provide two benefits from the same category. (1)

**Answer :**

**Ease Of Development:**

Django is a fairly simple platform where the organization of code is easily understandable. The way it separates the components like models, views and templates into individual Python files makes it very easy to understand, keeping the entire project disciplined and organized. As a newcomer, this allowed easy learning and grasping of knowledge as things were never jumbled up. The built-in Admin interface was also something that added to its ease, as there was no requirement of writing complex scripts to add data into the database. The CRUD commands are easy to understand and use, especially due to the very user-friendly interface.

**Security:**

Django proves an easy customisable authentification system that allows user authentication and permissions to be set up. It enables user registration, login, and password management, which can easily be implemented using secure access controls. All of this is easy for a new user as there is not much requirement for security principal knowledge while implementing such controls. For security, input sanitization, CSRF protection, and secure password storage are also some of the features that come with Django due to which a new user does not need any extensive knowledge as they are readily available to protect from common vulnerabilities.