

Juhan Hong GE03 Django Portfolio MVT Database

Purpose:

- Communicate effectively in a variety of professional contexts within a team, doing oral or written presentations, and creating technical documents.
- Function effectively as a member/leader of a team engaged in scrums while participating in different roles
- Apply various tools and agile principles utilizing concepts (scrum, behavior-driven development, pair programming, version control)
- Apply computer science theory such as utilizing design patterns for software architecture, higher-order functions, metaprogramming, to improve the maintainability, modularity and reusability to build a SaaS application

Effort: Collaborative Assignment see [CS3300 Academic Integrity](#)

Points: 40 (**see rubric in canvas**)

Deliverables: Each person submits their own word/pdf document of the GE with their own answers. You will also submit a separate document for your professional communication of your learnings. DO NOT SUBMIT A ZIP FILE.

Due Date: See Canvas

[1 Team Information](#)

[2 Set Up Models, DataBase and Admin Panel](#)

[Set Up Database and Migrations](#)

[Create SuperUser and Login to Admin Panel](#)

[Create Student Model](#)

[Models and Relationships](#)

[3 Apply Technologies \(Individual\)](#)

[4 Professional Communication of Technical Content \(Individual\)](#)

Description: Explore setting up your models and database to store student portfolio information by reading documentation, troubleshooting, collaborating in your team and asking detailed questions when you need help outside of your team. Start building your understanding of object relational mapping pattern, active record pattern, django models and database tables.

Feel free to work with others at the same time while setting up your models and database. If an issue arises, spend at least 20 minutes trying to troubleshoot and documenting what you did. Then collaborate with your team if you aren't already. If no one on the team can help then you can reach out to the appropriate discord channel. Think about how to provide details to ask a question to help others be efficient with helping you. If that doesn't help, come to my office hours.

1 Team Information

Read your assignment for the GE02 Team Roles in Canvas before you begin.

What is your role?

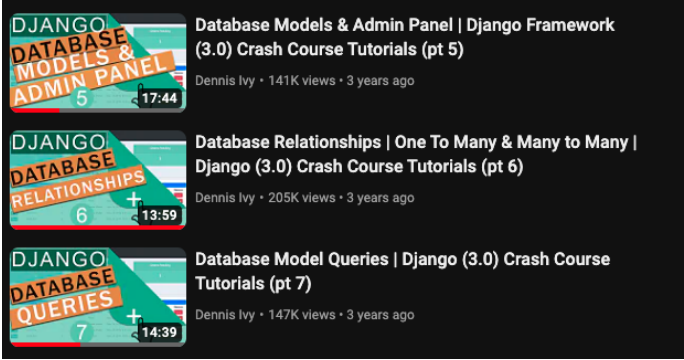
For GHE 03, I am a scrum leader.

What is your responsibility for this GE03 Sprint's Documentation?

My responsibilities is the models and relationships

2 Set Up Models, DataBase and Admin Panel

Here are some general resources for you that I found helpful in learning how to work with the models and database in Django.

Location	Summary
Django Tutorial Part 3: Using models Django Tutorial Part 4: Django admin site	This helps explain the components in setting up django models, admin site, and working with a database. You were given the "Setting up a Django development environment in the last guide exploration.
Django (3.0) Crash Course Tutorials Customer Management App - YouTube	This resource contains videos to explain the 
DJ101 Django Database ORM Mastery Course - YouTube	

Django documentation	<p>Django documentation is another good resource. Found this to be helpful after understanding the underlying concepts before using the technical documentation.</p> <div data-bbox="584 722 1367 1123"> <h3>The model layer</h3> <p>Django provides an abstraction layer (the "models") for structuring and manipulating the data of your application. Below are some links to help you get started:</p> <ul style="list-style-type: none"> • Models: Introduction to models Field types Indexes Meta options Model class • QuerySets: Making queries QuerySet method reference Lookup expressions • Model instances: Instance methods Accessing related objects • Migrations: Introduction to Migrations Operations reference SchemaEditor Writing migrations • Advanced: Managers Raw SQL Transactions Aggregation Search Custom fields Multiple databases Conditional Expressions Database Functions • Other: Supported databases Legacy databases Providing initial data Optimize database access </div> <div data-bbox="584 1131 1105 1407"> <h3>The admin</h3> <p>Find all you need to know about the automated admin interface:</p> <ul style="list-style-type: none"> • Admin site • Admin actions • Admin documentation generator </div>

1 Version Control

Set up a branch for GE03 sprint to do your development work. You will need to complete the work on this branch and then at the end when it is working you will merge it with your main branch but do not delete your other branches.

```
Python manage.py makemigration
Python manage.py migrate
```

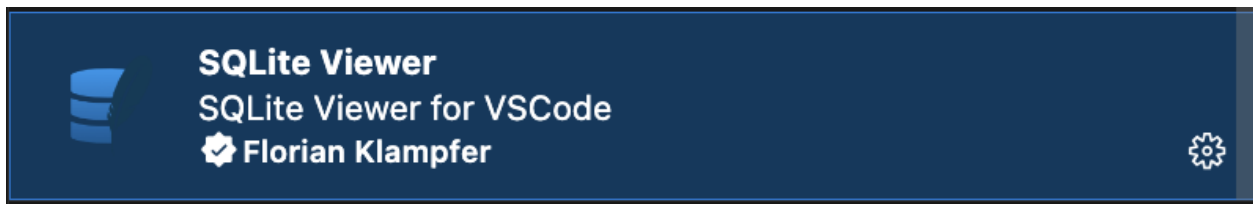
2 Set Up Database and Admin

In this section you will set up the admin and initial database tables that come with Django.

When putting screenshots please resize and crop appropriately.

Set Up Database and Migrations

If you didn't migrate the database yet for your portfolio application in the last GE do that now to set up the database. We will use the default SQLite for the portfolio app. You can use a different database for your own app. There are different extensions to view the database. I am using this SQLite VS code Extension to view the SQLite database <https://sqliteviewer.app/>.



Create SuperUser and Login to Admin Panel

In this section you are going to create a superuser to use the django admin application available to you. **You will not be creating your models through the django admin application.**

Python manage.py createsuperuser

Create Student Model

1. Create a student model in your portfolio_app by adding the following to the models.py. Remember to apply indenting when you copy and paste. Then you should update the database.

```
from django.db import models

# Create your models here.

class Student(models.Model):

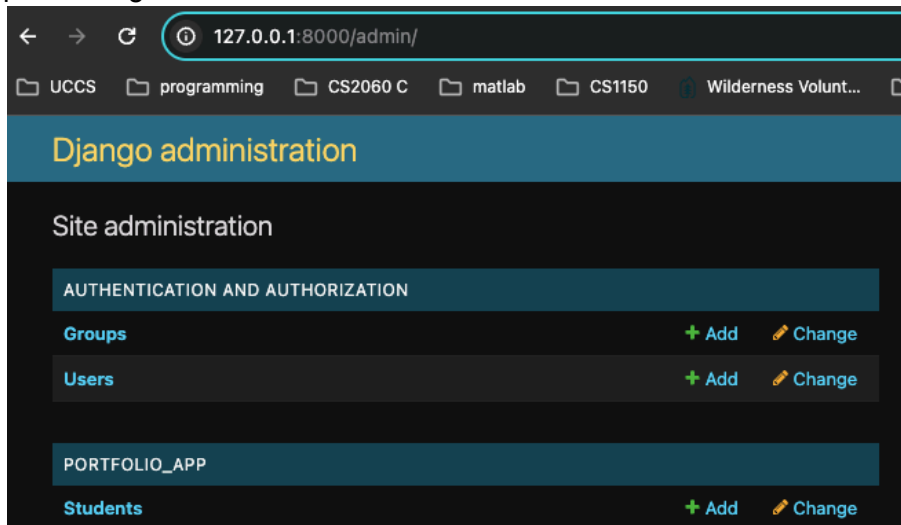
    #List of choices for major value in database, human readable
    name
    MAJOR = (
        ('CSCI-BS', 'BS in Computer Science'),
        ('CPEN-BS', 'BS in Computer Engineering'),
        ('BIGD-BI', 'BI in Game Design and Development'),
        ('BICS-BI', 'BI in Computer Science'),
        ('BISC-BI', 'BI in Computer Security'),
        ('CSCI-BA', 'BA in Computer Science'),
        ('DASE-BS', 'BS in Data Analytics and Systems Engineering')
```

```

)
name = models.CharField(max_length=200)
email = models.CharField("UCCS Email", max_length=200)
major = models.CharField(max_length=200, choices=MAJOR, blank =
True)

```

2. Perform migrations in django to add class model to database
3. Start the server if it isn't running and go to the admin panel. Notice the student table is not showing. The table must be registered in admin.py to access it from the admin panel. Register the table and then refresh to see the Student model.



4. Try adding a student without filling in any fields. What is required? What is not? Go back to your class Student model code. Update the code and database to make the major required. Now add a student by filling in the required fields.

To make the major required, simply delete the blank = "true" part

5. Click on the Student Table and notice it is called Student object(1). Not very informative. Remember overriding - Example of [overriding in java](#). Go back to the models.py and define the default string function to return the name in the Student class and the get_absolute_url to view the model's record editing screen. You must import python urls reverse.

```

from django.db import models
from django.urls import reverse

```

Then below major add the following.

```

major = models.CharField(max_length=200, choices=MAJOR)

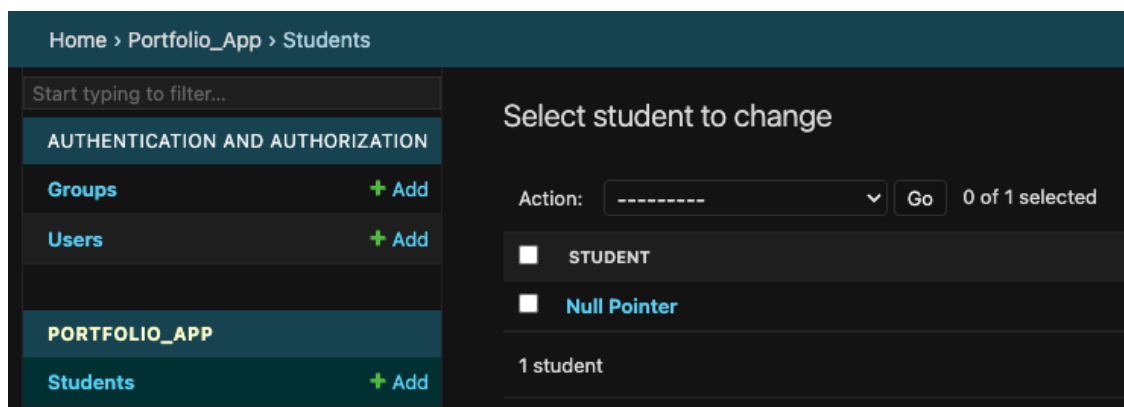
#Define default String to return the name for representing the Model
object."

```

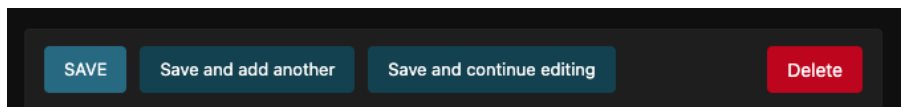
```
def __str__(self):
    return self.name

#Returns the URL to access a particular instance of MyModelName.
#if you define this method then Django will automatically
# add a "View on Site" button to the model's record editing screens
in the Admin site
def get_absolute_url(self):
    return reverse('student-detail', args=[str(self.id)])
```

Refresh the admin panel to see the name now displayed. If you don't see the change, go back and make sure you indented the methods correctly to be defined in the Student class.

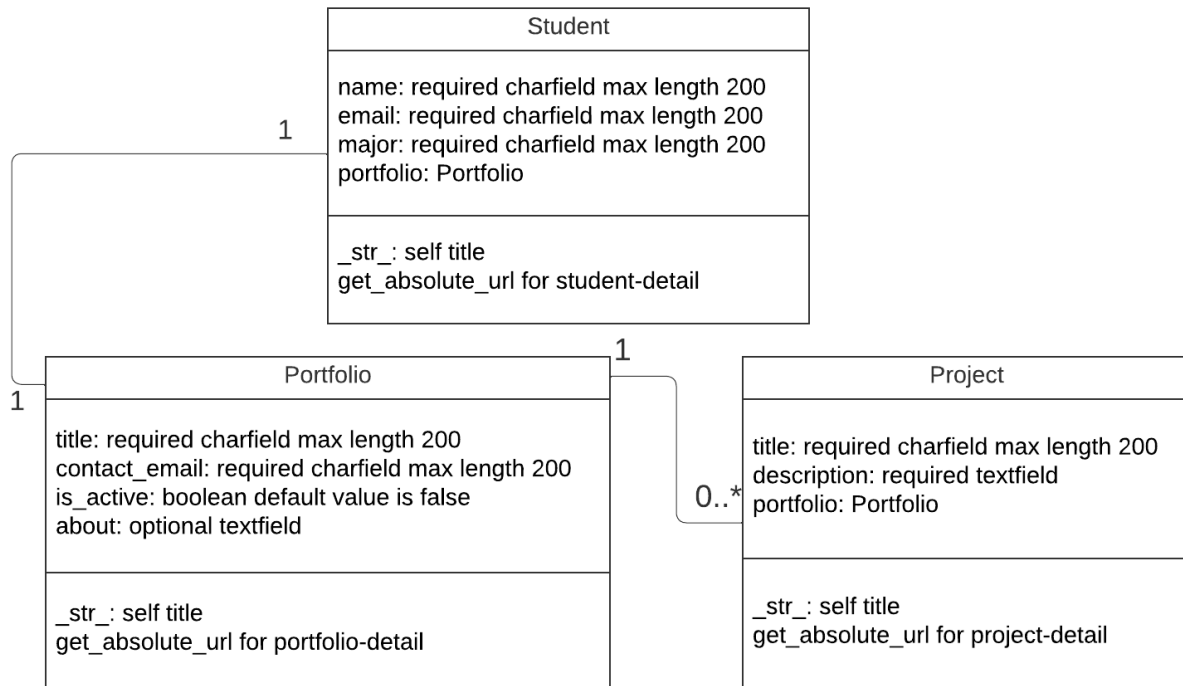


- Use the admin panel to delete the student from the table so the table is empty.



Models and Relationships

UML diagram for the required and optional fields, default fields and the field types.



Created using <https://www.lucidchart.com/>

1. Update the student model and create a portfolio and project model so that the fields are set up as shown above
 - a. What is required? What type of field?

I need to create a Portfolio and Project model. Student and Portfolio will have One to One field, Portfolio and Project will have One to Many field.

- b. What field do you add to the Student class to create the relationship for a Student model to have one to one portfolio.

OnetoOne field.

```
portfolio = models.OneToOneField('Portfolio', on_delete=models.CASCADE,
null=True, blank=True)
```

- c. What field do you add to the Project class to create the relationship for a portfolio to be able to have multiple projects?

One to many field

- d. What order must the classes must be put in the models.py?

The class that does not have any relationships with others must go first, then the ones with relationships. In this case, it would be project, portfolio, then student?

e. How do you define the `str` and `get_absolute_url` methods for each model?

For Project:

```
def __str__(self):
    return self.title

def get_absolute_url(self):
    return reverse('project-detail', args=[str(self.id)])
```

For Portfolio:

```
def __str__(self):
    return self.title

def get_absolute_url(self):
    return reverse('portfolio-detail', args=[str(self.id)])
```

For student:

```
def __str__(self):
    return self.name

def get_absolute_url(self):
    return reverse('student-detail', args=[str(self.id)])
```

When creating your classes and relationships your database can get messed up. Here are some things I found helpful. Explore these as you add the relationships below so practice this before you create relationships in your customer app project.

- Explore the relationships after migrating class updates in the django admin app. See what is required and how the relationships work between models.

Add student

Please correct the errors below.

Name: This field is required.

UCCS Email: This field is required.

Major: This field is required.

Portfolio: This field is required.

student1

Name: This field is required.

UCCS Email:

Major:

Portfolio:

- Delete the instances in the database in the django admin app if you need to update the class model due to a mistake before migrating again.

Select student to change

Action: Go 2 of 2 selected

☒ student 2

☒ student1

Action: Go

☒ PORTFOLIO

☒ student2

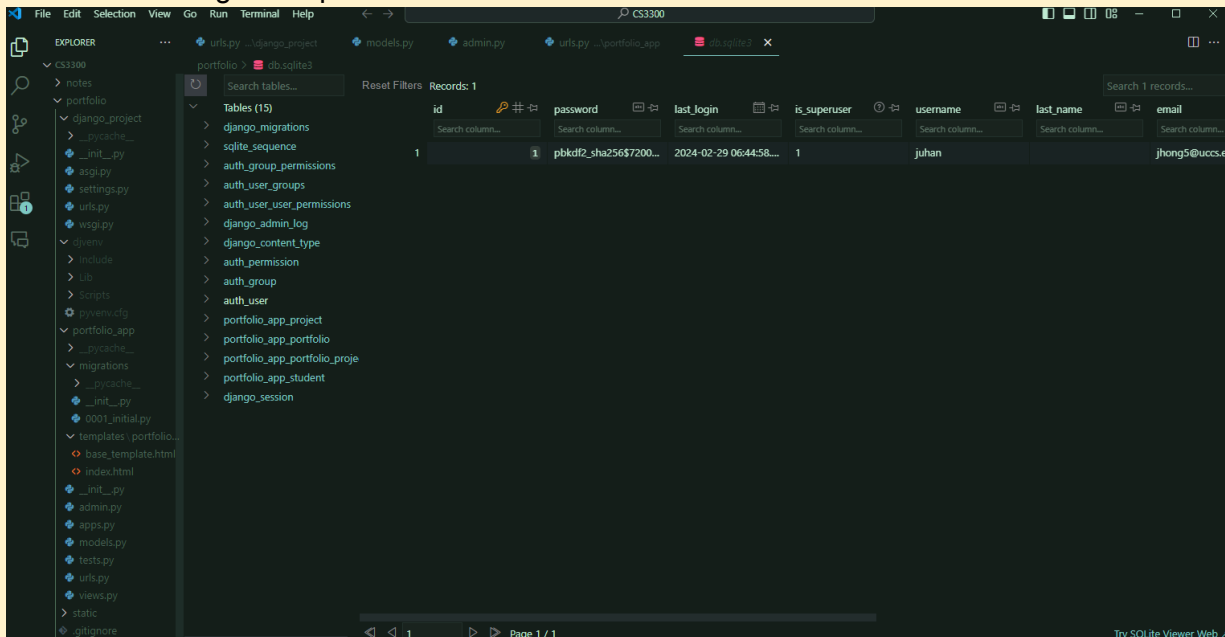
☒ Student 1 Portfolio

- Here is how to [Reset Database in Django](#) if you can't do the deletions above due to database issues. Depending on the data you delete from the tables you may need to restart the server and create a new superuser. Don't worry if the database gets messed up as this happens when developing an app and working with a database. That is why you have a local database associated with your own development environment.
- Read [Django One-To-Many Relationship](#). If you come across the following when migrating you may need to do option 1 to fix the situation.
It is impossible to add a non-nullable field 'portfolio' to a student without specifying a default. This is because the database needs something to populate existing rows.
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit and manually define a default value in models.py.

3 Apply Technologies (Individual)

These are individual responses and not team responses.

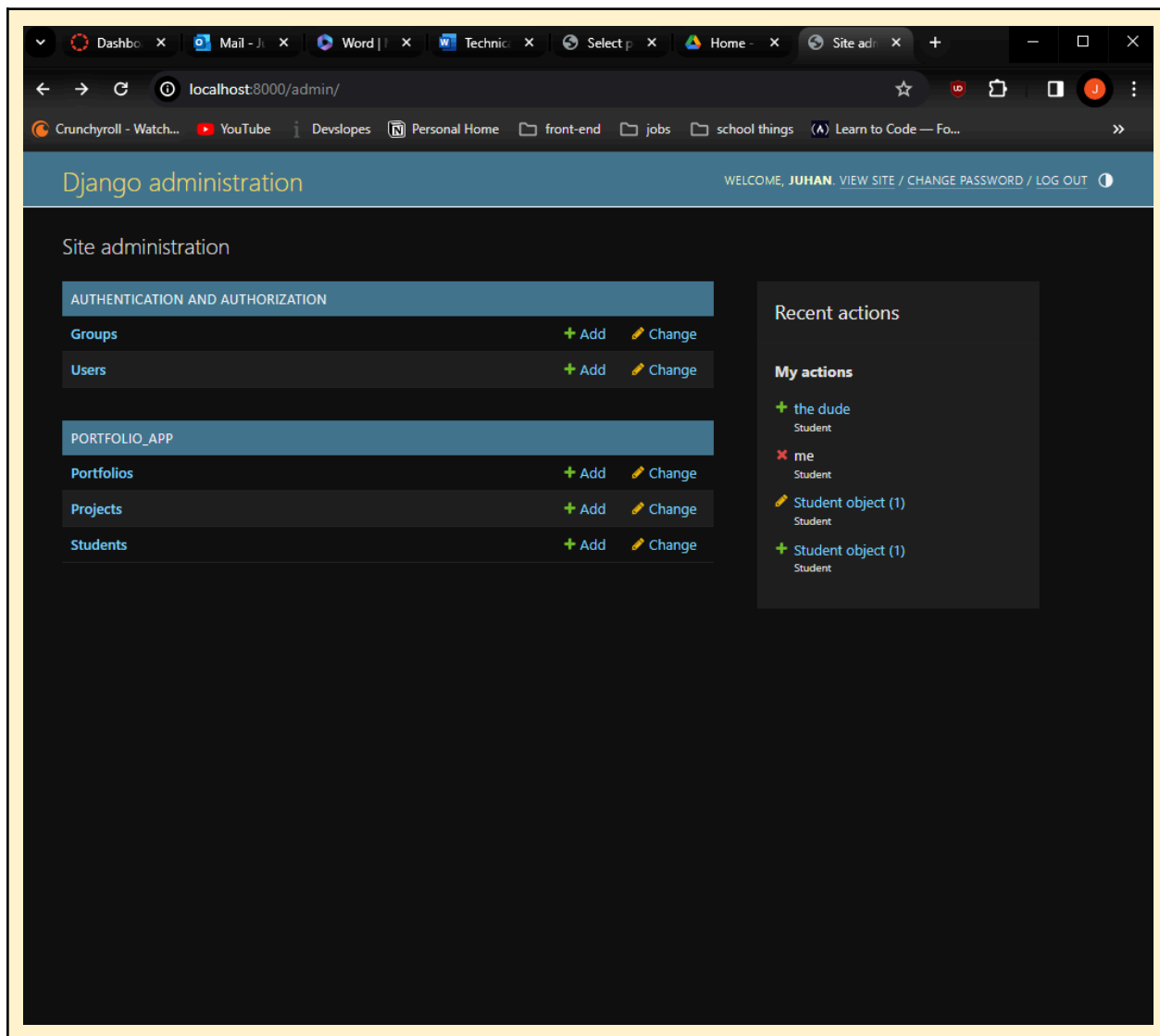
1. Take a screenshot of your database set up in your IDE showing the auth_user table containing the superuser record.

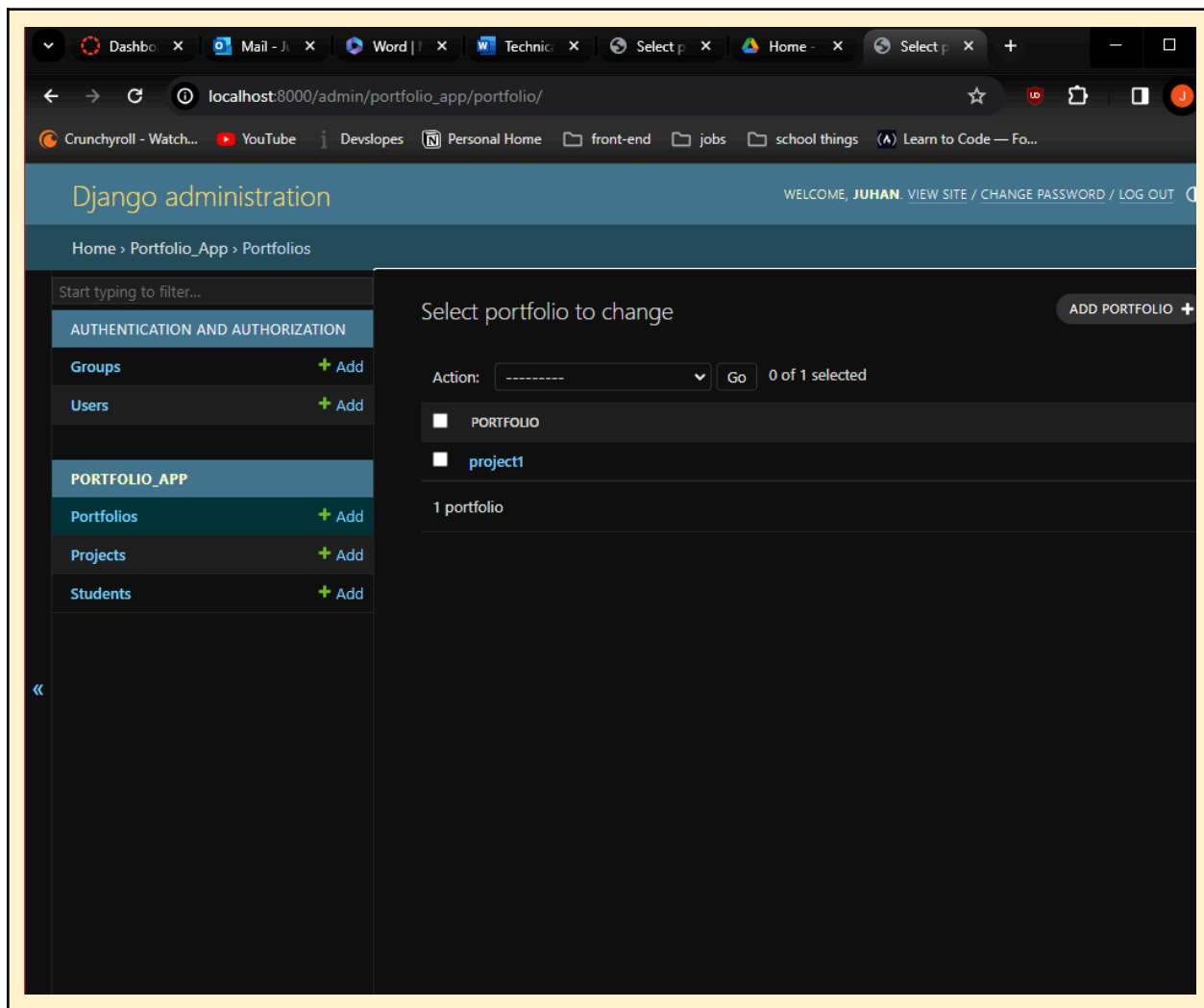


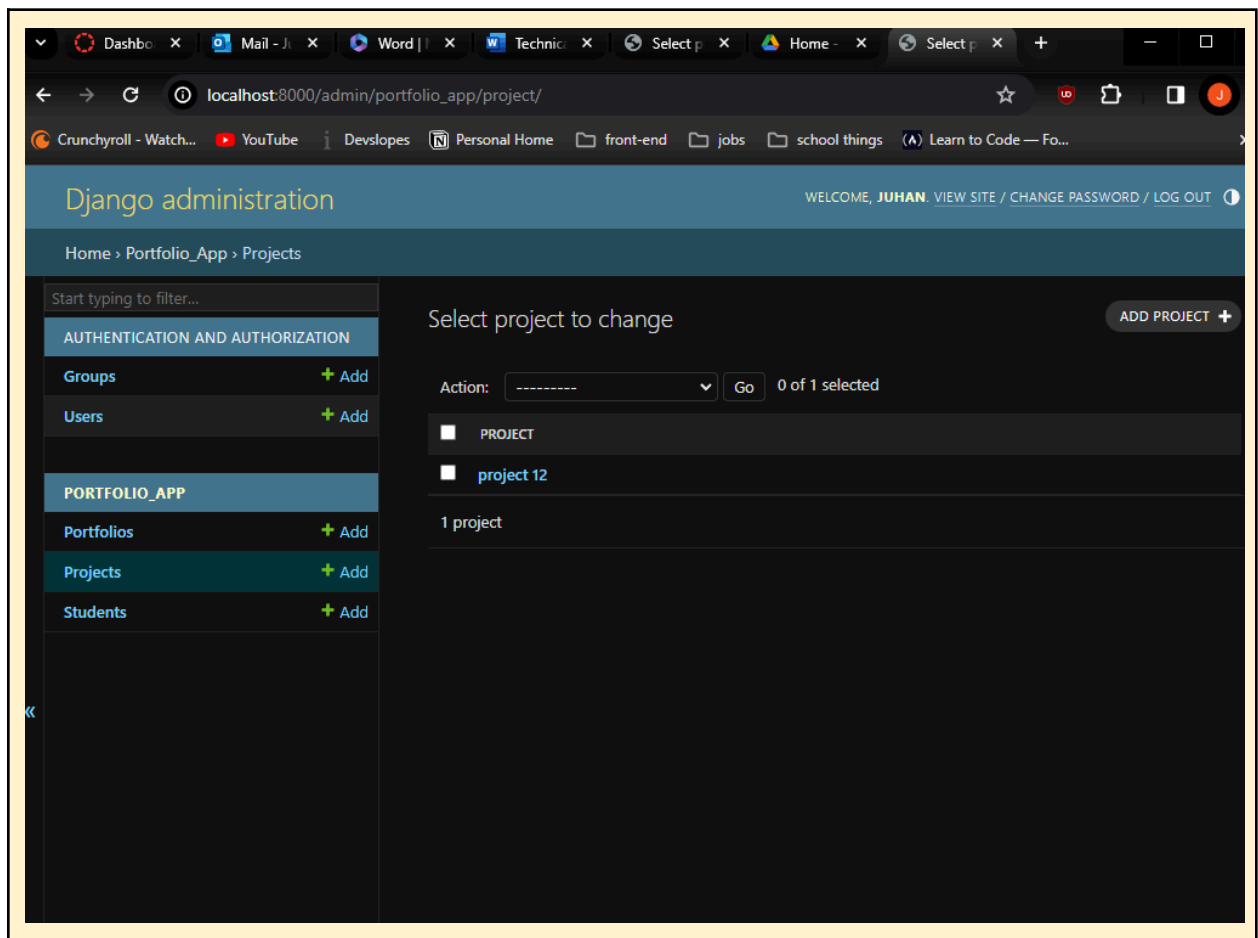
2. Take a screenshot showing your 3 branches - main, sprint 1 ge02 and sprint 2 ge03

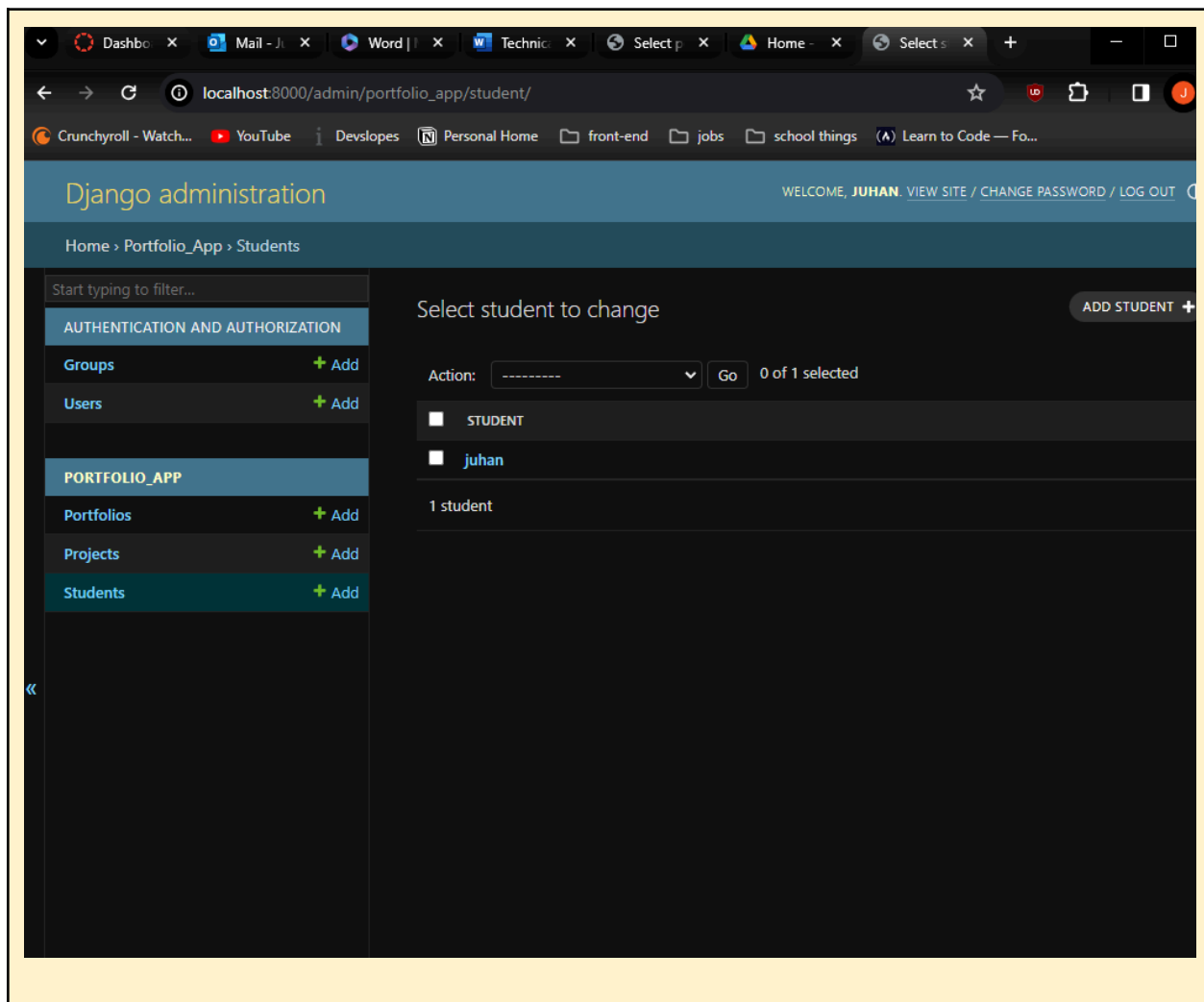
```
juhan@DESKTOP-53RQ2HD MINGW64 ~/OneDrive/Desktop/school/CS3300 (main)
$ git branch
* main
  sprint01
  sprint02
juhan@DESKTOP-538Q2HD MINGW64 ~/OneDrive/Desktop/school/CS3300 (main)
```

3. Put a screenshot showing you are logged into the admin tool that shows you can access the new models for Student, Portfolio and Project









4. Go to the database in your IDE and paste a screenshot showing information in your three tables associated with the Student, Portfolio and Project models.

```
class Portfolio(models.Model):
    title = models.CharField(max_length=200)
    contact_email = models.CharField(max_length=200)
    is_active = models.BooleanField(default=False)
    about = models.TextField(blank=True)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('portfolio-detail', args=[str(self.id)])
```

```

class Project(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True)
    portfolio = models.ManyToOneRel('portfolio', on_delete=models.CASCADE,
to='portfolio_app.portfolio', field_name='portfolio', related_name='portfolio',
related_query_name='portfolio')

    #Define default String to return the name for representing the Model
object."
    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('project-detail', args=[str(self.id)])

class Student(models.Model):
    #List of choices for major value in database, human readable name
    MAJOR = [
        ('CSCI-BS', 'BS in Computer Science'),
        ('CPEN-BS', 'BS in Computer Engineering'),
        ('BIGD-BI', 'BI in Game Design and Development'),
        ('BICS-BI', 'BI in Computer Science'),
        ('BISC-BI', 'BI in Computer Security'),
        ('CSCI-BA', 'BA in Computer Science'),
        ('DASE-BS', 'BS in Data Analytics and Systems Engineering')
    ]
    name = models.CharField(max_length=200)
    email = models.CharField("UCCS Email", max_length=200)
    major = models.CharField(max_length=200, choices=MAJOR)
    # create one-to-one relationship with Portfolio
    portfolio = models.OneToOneField('Portfolio', on_delete=models.CASCADE,
null=True, blank=True)

```

```
def __str__(self):
    return self.name

def get_absolute_url(self):
    return reverse('student-detail', args=[str(self.id)])
```

5. Explain following (Resource, URI, HTTPS methods and CRUD)

Resource: entity or object that can be identified by a URI

URI: String of characters that identifies a particular resources. URI includes URLs which specify where the resources are located.

HTTPS: defines the actions that can be performed on a resource.

CRUD: create, read, update, and delete.

- Create: add new resource
- Read: read info or resource
- Update: modify existing resource
- Delete: remove resources

http://127.0.0.1:8000/admin/portfolio_app/student/add/

http://127.0.0.1:8000/admin/portfolio_app/portfolio/1/change/

http://127.0.0.1:8000/admin/portfolio_app/project/1/delete/

The three addresses above represent a URL. The base URL being <https://127.0.0.1:8000>.

The remaining portion of the address such as [admin/portfolio_app/student/add](#) or [/admin/portfolio_app/portfolio/1/change/](#) are the path to the specific resources.

When looking at the terminal while navigating through the portfolio app, we get the following:

```
[29/Feb/2024 01:27:57] "GET /admin/portfolio_app/student/add/ HTTP/1.1" 200 12307
[29/Feb/2024 01:27:57] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[29/Feb/2024 01:29:55] "GET /admin/portfolio_app/project/ HTTP/1.1" 200 9405
[29/Feb/2024 01:29:55] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[29/Feb/2024 01:29:55] "GET /admin/portfolio_app/student/ HTTP/1.1" 200 9395
[29/Feb/2024 01:29:55] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[29/Feb/2024 01:29:56] "GET /admin/portfolio_app/student/1/change/ HTTP/1.1" 200 12621
[29/Feb/2024 01:29:56] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[29/Feb/2024 01:29:58] "GET /admin/auth/user/ HTTP/1.1" 200 11934
[29/Feb/2024 01:29:58] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
```

Showing the GET which is one of the HTTPS methods that is being used.

6. Run queries in the python shell command line using the model manager. Copy and paste the commands and results.

Query to get all the project objects and print out the results.

python manage.py shell

Python 3.11.8 (tags/v3.11.8:db85d51, Feb 6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

```
>>> import django
```

```
>>> from portfolio_app.models import *
```

```
>>> all_entries = Project.objects.all()
```

```
>>> all_entries
```

```
<QuerySet [<Project: project 1>, <Project:some cool project>, <Project: paper>]>
```

```
>>> projects = Project.objects.all()
```

```
>>> for project in projects:
```

```
...     print(project.title, project.description)
```

```
...
```

```
project 1 very cool project that will get me hired
```


```
some cool project very cool thing happening here
```

```
paper some very informative paper
```

From the query set of objects above get the first project and print the portfolio

As you can see, i think i got the relationship down to work. meaning projects going into the portfolio. But I was not able to get some answers in time to get query set for this question. When i run the command: all_entires[0].portfolio, it keeps saying 'Project' object has no attribute 'portfolio'

Django administration

WELCOME, **JUHAN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#) 

Home > Portfolio_App > Portfolios > project1

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add
Users + Add

PORTFOLIO_APP

Portfolios + Add
Projects + Add
Students + Add

Change portfolio

HISTORY

VIEW ON SITE >

project1

Title:

project1

Contact email:

me@uc

☒ Is active

About:

Projects:

project 1

some cool project

paper

+

Hold down "Control", or "Command" on a Mac, to select more than one.

7. Model Relationships

Paste the code of the field you added to the Student model to allow one to one portfolio.

```
portfolio = models.OneToOneField('Portfolio', on_delete=models.CASCADE,
unique=True, null=True, blank=True)
```

Paste the code of the field you added to Project model to allow multiple projects in a portfolio

```
# in portfolio
portfolio = models.ManyToManyField('portfolio', on_delete=models.CASCADE,
to='portfolio_app.portfolio', field_name='portfolio', related_name='portfolio',
related_query_name='portfolio')
```

8. List any issues encountered by you or someone else and how you approached resolving them.

Only issue i had was with the query. I could have set up the relationships with the models wrong but I think it is working.....?

4 Professional Communication of Technical Content (Individual)

These are individual responses and not team responses.

This is just a start of your understanding of different architectures, design patterns and concepts used when learning a framework to build SaaS apps. The goal is to see how you grow over the semester in your learnings.

Create a separate professional document (word or pdf) where main ideas are clearly presented with supporting evidence.

- Present a summary of the following concepts in connection to Django models and database for Student, Portfolio, and Project that includes
 - Active Record Pattern and Object Relational Mapping where Django automatically gives you a database-abstraction API to get or change data without writing database queries.
 - Routes in admin app for creating, updating and deleting information in the database
- Include resources used such as lectures, books, and your own research. You do not need to include citations in the writing but list the resources at the end.
- This should contain technical vocabulary, **your own** words, code snippets, diagrams (digitally or drawn on paper). Do not use an image that someone else created.
- About 1 ½ to 2 pages. Should not be more than two pages.