Welcome to the CoGrammar Django 2

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

- ***Timeframe:*** First 2 Weeks
- ***Guided Learning Hours (GLH):*** Minimum of 15 hours
- ***Task Completion:*** First four tasks

✅ Criterion 2: Mid-Course Progress

- ***Guided Learning Hours (GLH): 60***
- ***Task Completion:*** 13 tasks

CoGrammar

Skills Bootcamp Progression Overview

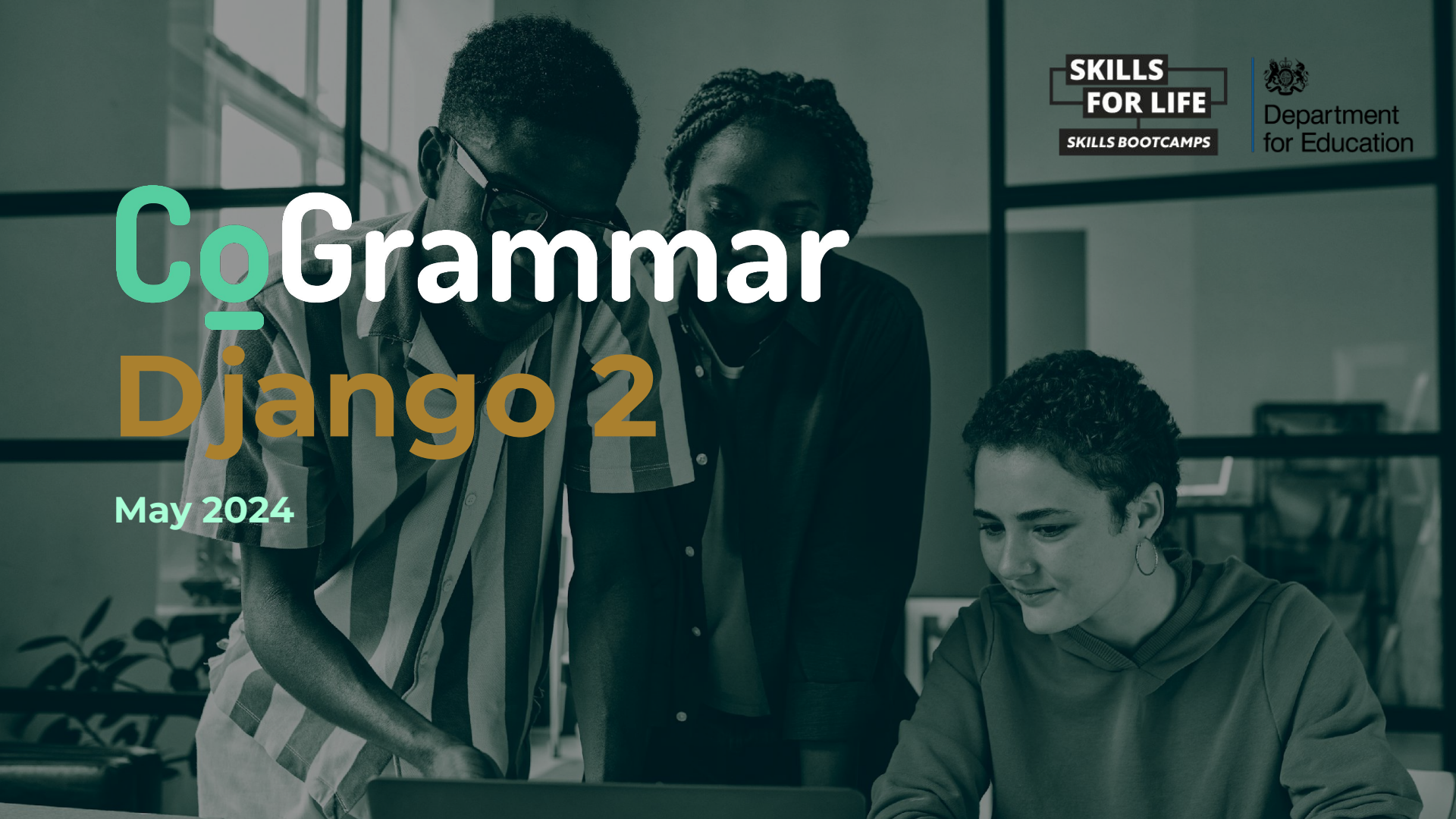
✅ Criterion 3: Course Progress

- ***Completion:*** All mandatory tasks, including Build Your Brand and resubmissions by study period end
- ***Interview Invitation:*** Within 4 weeks post-course
- ***Guided Learning Hours:*** Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

- ***Final Job or Apprenticeship Outcome:*** Document within 12 weeks post-graduation
- ***Relevance:*** Progression to employment or related opportunity

CoGrammar



SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Django 2

May 2024

Learning Objectives

- Describe Python-based Django concepts including **models**, **database migration**, **Django admin**, **forms/fields validation**, and data handling.
- **Develop** and **deploy** at least two custom **Django models** with Python-based ORM.
- **Execute** database **migrations** using Django's Python commands.
- Explain what an **API** is and identify the most common **RESTful** API functions such as **GET**, **POST**, **PUT**, and **DELETE**.

CoGrammar

Learning Objectives

- Utilise **Python** to perform **CRUD** operations on database records via the Django admin panel
- Utilise **SQLite** to perform **CRUD** operations on database
- Create Python-based custom **forms** for user input and validation within Django applications.
- Implement Python functions to handle form **submissions** and insert data into corresponding **database tables**.

CoGrammar

Poll

1. What topic do you find difficult to grasp?

   a. Python
   b. HTML/CSS
   c. Databases
   d. Everything

CoGrammar

Poll

2. What is Django primarily used for?

a. Developing mobile applications
b. Data analysis and visualisation
c. Building web applications and backend services
d. Machine learning and AI

CoGrammar

Poll

3. What does Django's ORM (Object-Relational Mapping) allow you to do?

c. Interact with the file system

d. Interact with the database using Python code

e. Create frontend UI components

f. Handle user sessions and cookies

CoGrammar

Introduction

CoGrammar

Intuition

Imagine you're an avid traveller, always seeking new adventures and experiences. Instead of keeping track of your journeys in a physical journal, envision a **Travel Diary web application** where you can **log in**, **document** your trips, **share recommendations**, and **connect** with **fellow** travellers.

Python drives the functionality, **Django** organises the **content**, **HTML/CSS** crafts the interface, and **databases** (MySQL, SQLite ...) **store** your travel tales securely.

This example illustrates how these technologies converge to create an immersive, user-centric platform, highlighting the versatility and interconnectedness of modern web applications in our everyday experiences.

Django: Empowering Python for Web Development

- Django follows the "**batteries-included**" philosophy, offering features like **authentication**, **ORM** (Object-Relational Mapping), **templating**, and more out of the box.
- Built with the **Model-View-Template** (MVT) architectural pattern
- Widely used for **building web applications**, **APIs**, and content management systems (**CMS**)
- Allows developers to **focus on writing application logic** rather than boilerplate code.

CoGrammar

Use Cases for Django

- **Front-End** Development: Building **User Interfaces**
  - E-commerce websites with product **listings**, **shopping carts**, and **checkout functionalities**
  - Personal blogs with **content management**, **commenting systems**, and **user** profiles
  - **Social networking** platforms facilitating communication and interaction **between users**

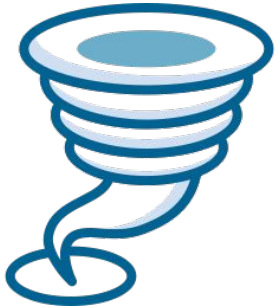
CoGrammar

Use Cases for Django

- **Back-End** Development: **Powering** the Engine
  - **Microservices**: Breaking down complex applications into smaller, **independent services**,
  - **APIs** (Application Programming Interfaces): Enabling communication **between different applications**
  - Django's REST **framework** for building APIs and its seamless **integration** with various **front-end technologies**.

CoGrammar

Django in Comparison with Other Tools



Django Core Concepts

CoGrammar

Django Project

- Describes a Django web application
- The root directory of your application.
- The high-level container for your entire application
- Houses settings, configuration files, and overall application logic
- There is only one project per application
- A project can contain multiple apps.

CoGrammar

Django Project

To start a project:

...\> django-admin startproject project_name

CoGrammar

Django Project

Django project structure

- **project_name**/
  - manage.py
  - **project_name**/
    - __init__.py
    - settings.py
    - urls.py
    - asgi.py
    - wsgi.py

CoGrammar

Django Project

Run the project

- ➔ ...\> cd project_name
- ➔ ...\> py manage.py runserver

CoGrammar

Django Project

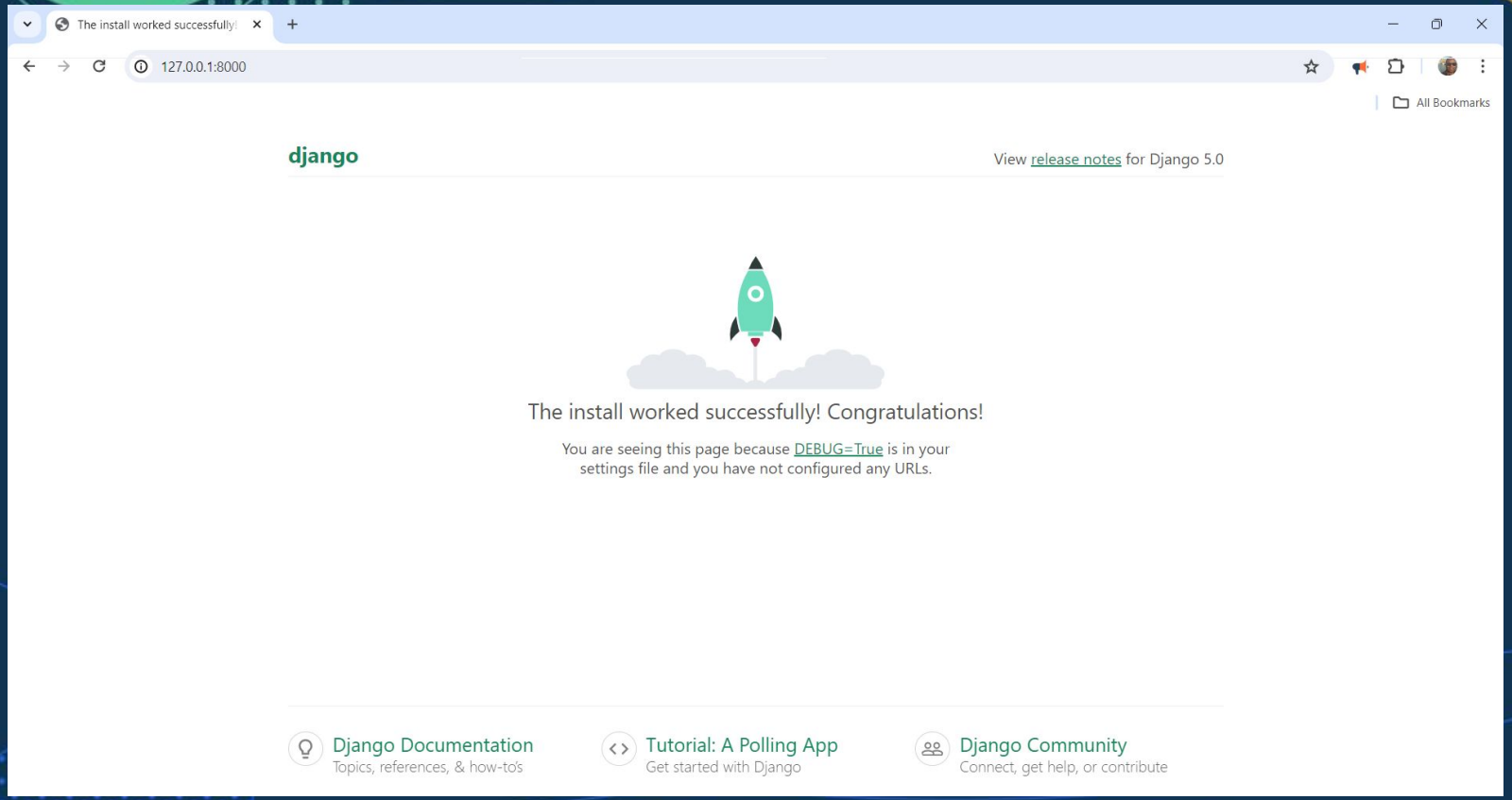
```
(django_venv) PS C:\Users\[redacted]\OneDrive - [redacted]\Documents\django_test> cd .\mysite\
(django_venv) PS C:\Users\[redacted]\OneDrive - [redacted]\Documents\django_test\mysite> py .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes
, sessions.
Run 'python manage.py migrate' to apply them.
May 20, 2024 - 15:04:04
Django version 5.0.6, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

CoGrammar

Django App



CoGrammar

Django App

- Self-contained components within a project
- Contains models, views, templates, etc.
- Encapsulates related functionality
- Can be reused across projects
- Fundamental components in Django projects.
- Represents a distinct functionality or feature area of your application.
- Reusable for common functionalities

CoGrammar

Django App

To start a app (after starting a project):

- ➔ **...\> cd project_name**
- ➔ **...\> py manage.py startapp my_app**

CoGrammar

Django App

Django app structure

- my_app/
  - __init__.py
  - admin.py
  - apps.py
  - migrations/
    - __init__.py
  - models.py
  - tests.py
  - views.py

CoGrammar

Models

- Represent database tables as Python classes
- Define structure of the data, including fields and behaviors
- Use models to interact with the database
- Automatic creation of database tables
- Each model maps to a single database table

```python
from django.db import models

class Blog(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    published_date = models.DateTimeField(auto_now_add=True)
```

CoGrammar

Forms and Data Handling

- Forms handle user input in Django applications.
- They define the fields users can interact with and validate the submitted data.
- Forms make collecting and processing user input secure and efficient.

```
from django import forms
from .models import Blog

class BlogForm(forms.ModelForm):
    class Meta:
        model = Blog
        fields = ['title', 'content']
```

CoGrammar

Data Migration

- Django migrations track changes to your models.
- When you modify a model, migrations create instructions to update the database schema.
- Track changes and roll back if needed
- Automatically generate SQL for database changes
- This ensures your database reflects your latest data structure.
- Version control for your database
- Help maintain data integrity during schema changes

CoGrammar

Data Migration: Workflow

To start a app (after starting a project):

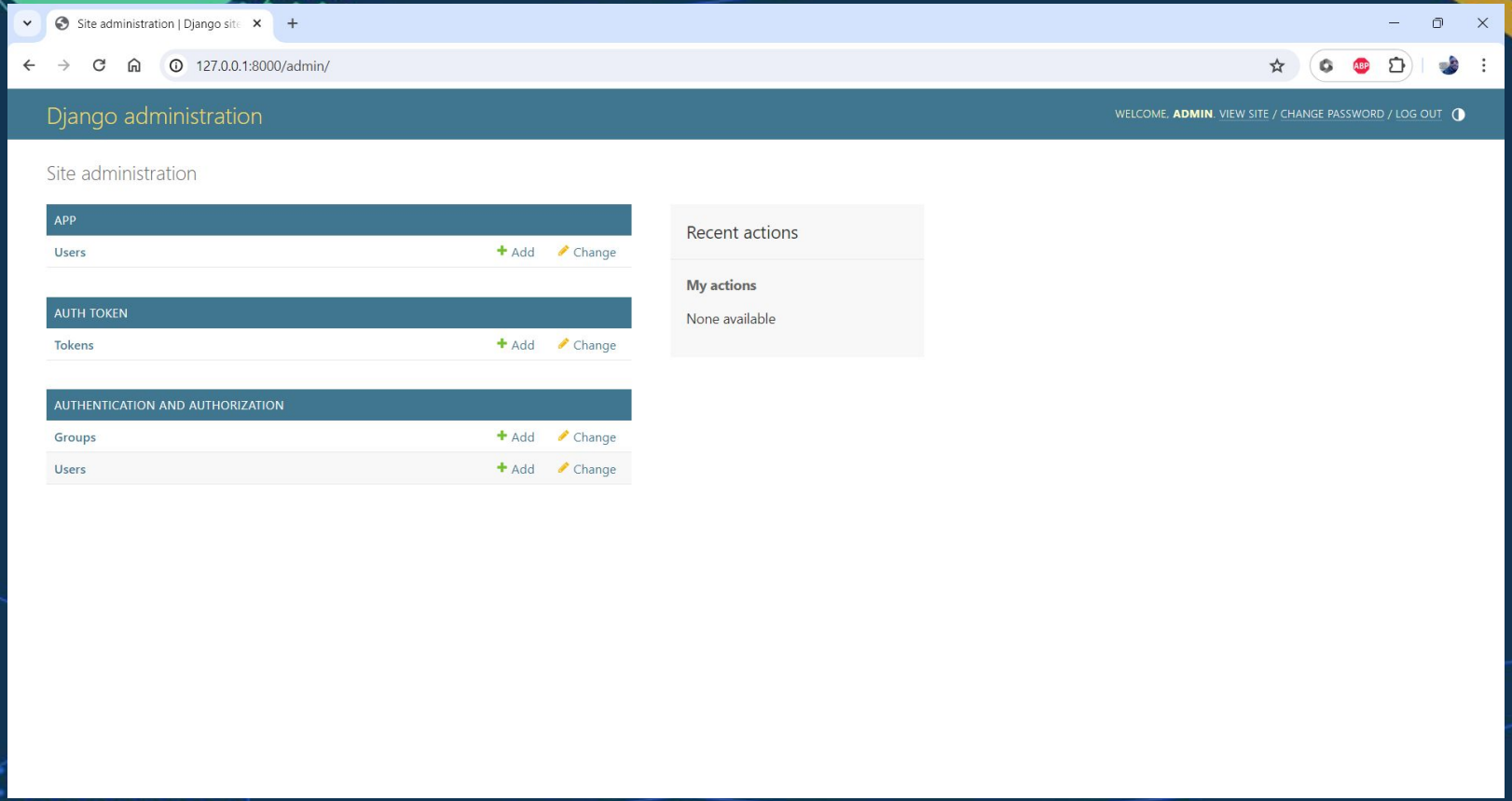
- ➔ **...\> python manage.py makemigrations**
- ➔ **...\> python manage.py migrate**

Django Admin

- Provides a web-based interface to your application's data.
- Built-in admin interface
- Automatically generates admin interface for registered models
- Allows CRUD (Create, Read, Update, Delete) operations on data
- Provides user authentication and authorization out of the box

CoGrammar

Django Admin



CoGrammar

Let’s get coding!

CoGrammar

Agenda

1. Create a Django Project
2. Create a Django app
3. Implement Model and Logic
4. Implement forms, data handling
5. Retrieve data from both Template view

CoGrammar

Final Assessment

CoGrammar

Poll

1. Which Django command is used to start a new Django project?

   a. django-admin startapp
   b. django-admin startproject
   c. python manage.py startproject

CoGrammar

Poll

2. What is the correct syntax to create a Django model named Book with fields title?
   Book is defined as such "**class Book(models.Model):**"

   a. title = models.CharField(max_length=100)
   b. title = models.TextField()
   c. title = models.CharField()

CoGrammar

Poll

3. In Django, how do you retrieve all objects from a model named Book?

   a. Book.objects.fetch_all()
   b. Book.objects.all()
   c. Book.objects.get_all()

CoGrammar

Lesson Conclusion and Recap

CoGrammar

Lesson Conclusion and Recap

- **Django Core Concepts:** Fundamentals of Django, including models, views, templates, and forms, which form the backbone of web application and backend development.
- **Database Management:** Django's migration system facilitates database management by synchronising changes in models with the database schema.
- **Admin Interface:** Django's admin interface, a powerful tool for managing application data through a user-friendly interface, without writing custom views or forms.
- **Forms and Validation:** The importance of custom forms in Django for handling user input, validation, and data submission, ensuring data accuracy and security.
- **API Development:** The significance of RESTful APIs in Django, aided by Django REST Framework, for enabling communication between different parts of a web application or external systems, enhancing interoperability and scalability.

Homework or Follow-up Activities

CoGrammar

Homework or Follow-up Activities

Objective: Practise writing and deploying apps with Django

1. **Easy**
    a. Remove the photo item. It is not useful. Just keep the following:
        i. Name, email, password, mobile_number, date_of_birth
    b. Make sure that the phone number is actually a number. That field is not validated. It can take any character. Hint: Check the password field.
    c. Add a column that would be associated to each user, to provide their age given their date of birth.

CoGrammar

Homework or Follow-up Activities

2. **Mid-Hard**
   a. Let's assume that you are the CEO of a startup. Your main idea is to collect data from estate agent companies around London, say company A, B and C for a start.
   b. Your solution will happen in 2 folds:
      i. You provide a webpage where customers can log on and be recommended the best property to buy for their budget. On the front end, the web page.
      ii. You will need to make money.
         1. That would happen through premium services, like showing more than one house if the fee is paid. This is still the frontend,
         2. Now, banks need your solution. Create an API endpoint for the same service, billing them per request.

Homework or Follow-up Activities

Instructions:

- Make use of Django's tutorial to get started!
- The assumption here for exercise 2:
  - You will create your population in terms of the people who will register through the Django form. Get all the required and validated data, name, age, home address ...
  - The houses will also be fictitious. Create enough variation in prices and location such that it is random enough
  - You will use the Django REST framework for the rest endpoints. Please use the provided code, used during the practical.

CoGrammar

References

- https://www.django-rest-framework.org/
- https://docs.djangoproject.com/en/5.0/intro/tutorial01/
- https://medium.com/dsc-umit/mvc-vs-mvt-architectural-pattern-d306a56dce55
- https://www.linkedin.com/pulse/decoding-design-patterns-comparative-analysis-mvc-mvt-ahmed-el-banna-pzefe/
- https://code.visualstudio.com/docs/python/tutorial-django

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar