BSc Computer Science
CM3035 - Advanced Web Development
Midterm Coursework: RESTful Web Service

Introduction

During the course so far, we have developed several database backed web servers using Django. For this assignment you are tasked with developing a RESTful web service using the knowledge you have gained so far. The server should use Django Models to appropriately design and model the input data and provide usable migrations for the underlying database. As per the examples in the lessons you will be working with a small dataset of a domain of your choice: it can be a dataset of medical, biological, financial, demographical etc data. You should justify the use of this dataset in your report. Your dataset can have several .csv files. The only requirement for this dataset is to have a maximum of 10,000 entries overall to load and process in short time. You could search for some datasets using these resources:

https://datasetsearch.research.google.com/
https://github.com/awesomedata/awesome-public-datasets
https://data.nhm.ac.uk/search

To successfully implement the application, you should have a good understanding of the flow of data through a Django application. You should make appropriate use of serializers, forms and validation. You should ensure you make appropriate use of Django Views. The application should be laid out in a manner that is clear and easy to follow.

This assignment is worth 50% of the total mark for this module.

Task

Build a RESTful web application that will store the data in a relational database and perform queries to retrieve, add, delete, and update entries. You should implement 6 interesting RESTful endpoints where you can demonstrate your Django knowledge and the aspects of your chosen dataset e.g. return a list of all road accidents which involve alcohol use during the weekends. You should consider the complexity of your queries to maximise your marks. One of the endpoints should use POST e.g., to save new information. Use serialisation in at least one of your queries and finally implement unit testing for your endpoints. The application should include a 'load and store' python script to load your data from the .csv file or files and store it in a database, perform the necessary processing and return the JSON documents of your REST endpoints.

For this application ensure you use the default SQLite3 database. DO NOT use Postgres.

Deliverables

D1. The Django application that implements the REST API.

D2. The instance of the Django application should have all the data loaded.

D3. A python script or method for loading the .csv data into the database. You must include a pip requirements.txt file to ensure that your application libraries can be loaded.

D4. A short report (2000 words) in .PDF format explaining the code in your application and how it meets the requirements (see below for criteria R1-R4). You may use focused, short code extracts if they make your explanation clearer. Explain the logic of your approach, why is your code arranged as it is. Talk about the dataset and the endpoints and why these are interesting. Include a section on how to unpackage and run your application:

- A list of all packages and the versions used for your implementation
- Your development environment i.e. the operating system and python version
- Instruction for logging into the django-admin site i.e. username and password
- Include how to run the unit tests and the location of the data loading script

D5. A video of your web application showing and verbally explaining deployment and the main functionalities i.e.: unzipping the application, installing and deploying the application using the requirements.txt file, populating the database using a seeding script, running the python tests and finally demonstrating the RESTful endpoints. Also, we would like to hear why you chose this dataset, why your endpoints are interested and finally talk about the database design. This should not be longer than 8 minutes. We recommend that you capture the video in mp4 format using software such as OBS. You can upload the video in mp4 format or use the alternative link – see submission page.

D6. Bonus points will be given to those who deploy their app using AWS, Digital ocean, etc. You should supply details in your report i.e. app address and login details.

Requirements

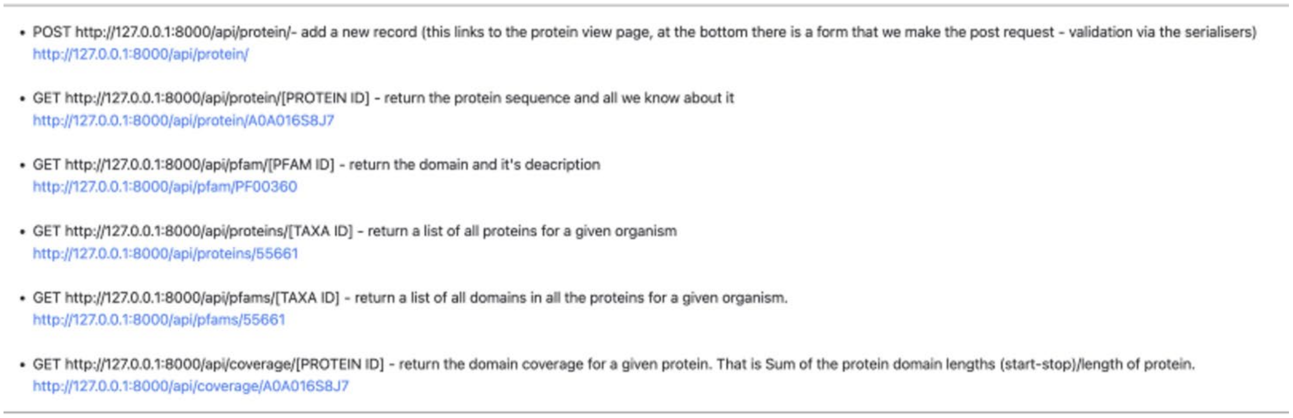
We will assess your work based on the following requirements and criteria:

R1: The application contains the basic functionality described in class
 a) correct use of models and migrations
 b) correct use of form, validators and serialisation
 c) correct use of Django-rest-framework
 d) correct use of URL routing
 e) appropriate use of unit testing
R2: Implements an appropriate data model for the data
R3: Implementation of appropriate code for the required REST endpoints
R4: Implementation of appropriate method of bulk loading data

Code style and technique

Your code should be written according to the following style and technique guidelines:

C1: Code is clearly organised into appropriate files (i.e. view code is placed in an appropriate view.py or api.py file, models are placed in an appropriate models.py file)
C2: Appropriate comments are included to ensure the code is clear and readable
C3: Code is laid out clearly with consistent indenting, ideally following python pep8 standard
C4: Code is organised into appropriate functions with clear, limited purpose
C5: Functions, classes and variables have meaningful names, with a consistent naming style
C6: Appropriate Unit tests to cover the REST API functionality are provided
C7: Code implements endpoints as hyperlinks. This should be found at the main page – see below



- POST http://127.0.0.1:8000/api/protein/- add a new record (this links to the protein view page, at the bottom there is a form that we make the post request - validation via the serialisers)
  http://127.0.0.1:8000/api/protein/
- GET http://127.0.0.1:8000/api/protein/[PROTEIN ID] - return the protein sequence and all we know about it
  http://127.0.0.1:8000/api/protein/A0A016S8J7
- GET http://127.0.0.1:8000/api/pfam/[PFAM ID] - return the domain and it's deacription
  http://127.0.0.1:8000/api/pfam/PF00360
- GET http://127.0.0.1:8000/api/proteins/[TAXA ID] - return a list of all proteins for a given organism
  http://127.0.0.1:8000/api/proteins/55661
- GET http://127.0.0.1:8000/api/pfams/[TAXA ID] - return a list of all domains in all the proteins for a given organism.
  http://127.0.0.1:8000/api/pfams/55661
- GET http://127.0.0.1:8000/api/coverage/[PROTEIN ID] - return the domain coverage for a given protein. That is Sum of the protein domain lengths (start-stop)/length of protein.
  http://127.0.0.1:8000/api/coverage/A0A016S8J7

Submission

You should write a brief report, record a video demo and submit your source code. The submission should contain the following items and information:

S1: Deliverables D1, D2 and D3 compressed in standard .ZIP format.

S2: Deliverables D4 and D6 in .PDF format.

S3: Deliverable D5 in .mp4 format.

S4: Deliverable D5 - alternative link – use of YouTube or similar and submit the link.

Marking Criteria

The application will be graded on whether it is technically correct and implements the API as requested. Code should be clear and easy to follow. The application should be well organised - for instance - it should make correct use of models, API, view and serializer files. A good application will include a suite of tests that ensure that application correctly implements the API that is described.

See separate rubric file for details.