← Back **Implementing a simple REST API [001]**
Graded Assignment • 10h

Due Jul 17, 2:00 PM WAT

Your grade: 77.50%

Your latest: **77.50%** • Your highest: **77.50%** • We keep your latest score. Review your overall course grades here.

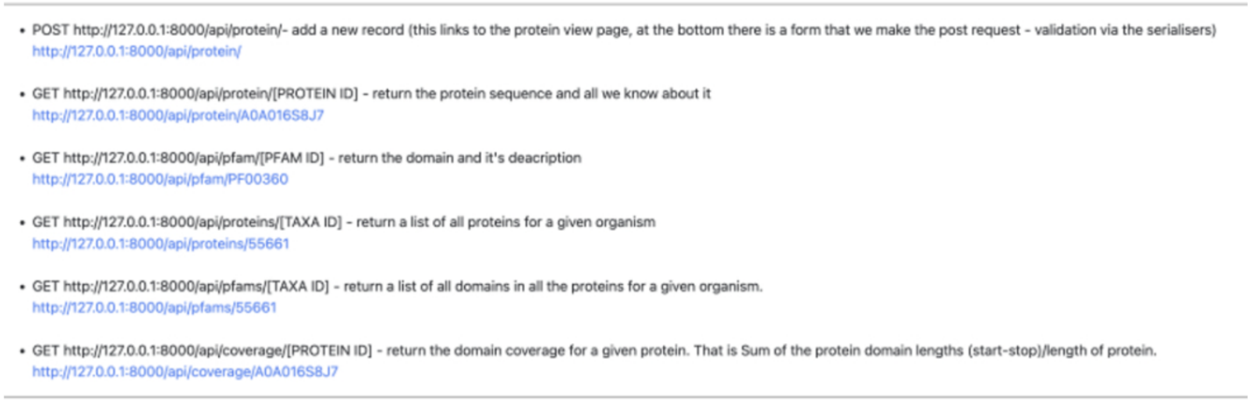
Next item →

Instructions

1.

- A zip file of the Django application that implements the REST API as described in the accompanying REST specification
- The instance of the Django application in the zip file should have all the data loaded
- The zip file should contain a python script or method for loading the provided CSV data in to the database
- A video of your web application showing the main functionalities (This should be included in the zip)
  Your video should show: 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. This should not be longer than 5 minutes. We recommend that you capture the video in mp4 format using software such as OBS
  - This is how we grade the functionality that you have implemented against the rubric so be sure to show all of the functionality within the time limit
  - We will not watch the video beyond the time limit
  - Make sure to use hyperlinks for your endpoints in your homepage. This will make debugging and testing easier and faster, see example below:

41 / 55 points



- 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

awd.zip

Grading Rubric

Application loads as required: a) requirements.txt has the correct requirements, b) loads with no compatibility issues, c) runserver loads localhost in browser

0 points	None of the above achieved
1 point	One of the above achieved
2 points	Two of the above achieved
[✓ **3 points**]	Three of the above achieved

Application implements REST endpoints required.

Give one point for each endpoint - 6 overall.
Score **6/6**

Application implements REST endpoints as links e.g.

http://127.0.0.1:8000/api/protein/A0A016S8J7
http://127.0.0.1:8000/api/pfam/PF00360

0 points	1 or less REST endpoints links
1 point	Some REST endpoints links

[✓ **2 points**]	All REST endpoints links

Database/Model design is appropriate for the data provided

0 points	Database is poorly laid out with no normalisation
1 point	Database is sensibly designed with minimal evidence of normalisation
[✓ **2 points**]	Database is sensibly designed with some evidence of normalisation
3 points	Database is well laid out with correct use of normalisation and tables

A method/system is provided to load data in bulk

[✓ **0 points**]	Data can not be loaded
1 point	Data loading method is functional
2 points	Data loading method is functional and to a high technical standard

Application makes use of functionality of Django as described in class. If a student chooses not to use Django and uses a different web framework, it should be evident that they are making use of course concepts appropriately

0 points	No use of django or submission is absent
[✓ **1 point**]	Django/concepts used as described in the classes
2 points	Django features or material beyond the course content are used

Unit testing of API endpoints/views

Testing of functions and classes

Testing to cover all edge cases

0 points	No unit tests
[✓ **1 point**]	Unit tests are included but do not fully test the applications boundaries
2 points	Unit test cover most expected boundary conditions for the endpoints
3 points	Extensive unit tests covering many endpoint edge cases are included
4 points	Tests are present across all student written functions within the application (i.e. data loading, serialisers etc...)
5 points	Testing covers edge cases and errors
6 points	Testing is unusually thorough and state of the are (i.e may use packages such as Hypothesis, or the report explains the students structured testing methodology)

Code is clean with good syntax and comments. That is, the logic of the application is easy to follow without having to run the code or insert debug statements

0 points	Not attempted/messy code and hard to follow
1 point	Code is acceptably organised with some comments
[✓ **2 points**]	Code is well organised and clear with consistent comments
3 points	Code is exceptionally well organised and adheres to pep8 standard

Code is functional (ie free of errors, reproducibility of results is reasonable)

0 points	Not attempted/non-functional
1 point	Code is functional with many bugs or obvious errors

1 point Code is functional with many bugs or obvious errors

2 points Code is functional, few apparent bugs or errors

[✓ **3 points**] Code is functional with minimal or no bugs or errors

Code is modular and well organised (i.e. naming conventions, correct use of functions and classes, no huge code blocks)

0 points Not attempted

1 point Code makes minimal or no use of sensible organisation strategies (i.e. no use of appropriate functions, naming schemes are inconsisten)

2 points Code makes some use of sensible organisation strategies

[✓ **3 points**] Good use of code organisation (such as functions and good naming strategies)

4 points Code is excellently organised. Functions, classes, closures all used appropriately.

Student evidences advanced web programming and python techniques, beyond the current course content, are used (e.g OpenAPI, Hypothesis, lambdas)

[✓ **0 points**] Not attempted

1 point Rare use of advanced coding methods

2 points Some use of more advanced web programming ideas beyond the content in topics 1-5

3 points Extensive use of more advanced web programming ideas beyond the content in topics 1-5

Bonus points if you deploy your own app using AWS, Digital ocean, etc.

0 points No

[✓ **8 points**] Yes

Code labels - Did you clearly label all sections of your code?

[0]No code or code provided with no labels

[2]Incorrect labels e.g. student didn't use start/end labeling for the code

[4]Some labels exist but leave some grey areas

[7]Most of labels in place

[10]All labels in place
Score **10/10**

2. A brief report (approx 2000 words) explaining the code of the application and how it meets the requirements (criteria R1-R4 and C1-C6). You should clearly explain the design of your application and the decisions you have made. This report is assessed alongside the application zip file. **16 / 20 points**

The report should explain the code in your application and how it meets the requirements. Explain the logic of your approach - why is your code arranged as it is?

Also, include instructions on how to install and run your app:

- A list of all packages and the versions they used for your implementation
- A list of your development environment i.e. the operating system you used and the version of python
- 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

Only a .pdf format is accepted.

AWD%20MIDTERM.pdf

Grading Rubric

Report is clearly written

0 points No report

1 point	Report is present but with poor writing and organisation
[✓ 2 points]	Report is good enough and to the expected standard
3 points	Report is written to an excellent standard and very well organised

Report explains how the application requirements have been met

0 points	Report does not address requirements
1 point	Report explains how some requirements have been met
2 points	Report explains how most of the requirements have been met
[✓ 3 points]	Report explains how all requirements have been met

Good clear Application of the techniques taught (i.e. best practice use of django and django-rest-framework evidenced)

0 points	Not attempted
1 point	little evidence of best practice or the material taught
2 points	some of the material from topics 1 to 5 has been used
3 points	most of the material and techniques from topics 1 to 5 evidenced
[✓ 4 points]	Clear evidence of all the material and ideas taught in topics 1 to 5

The report explains, and code evidences, how the student was systematic in their design of the application and this design process was relevant to the nature of the task (e.g. there is evidence of how they broke down the database design, how they tackled implementing the application bases on their database design)

0 points	Approach does not attempt to fulfil brief
1 point	Approach attempts to fulfil brief but is incorrect
[✓ 2 points]	Report explains how the application was designed
3 points	Report explains clearly all steps and the structure of the approach to designing the application

Report show evidence of critical evaluation of work/approach in relation to state of the art in web development

0 points	Report does not not attempt to evaluate their work
1 point	Report shows minimal evidence of evaluation of the project
[✓ 2 points]	Report has evidence of some evaluation of the application and highlights some strengths or some weaknesses
3 points	Student has evaluated the strengths and weaknesses of their application and shows some insights in to possible improvements
4 points	Report shows excellent evidence of critical evaluation of their work, what could be improved and why. The student can relate that to ideas and themes in contemporary web development

Report includes necessary run info such as OS, Python version and login credentials

0 points	None provided
1 point	One of the above
2 points	Two of the above
[✓ 3 points]	All of the above

Grader, please briefly comment on any/some of the following bullet points.

Grader, please briefly comment on any/some of the following bullet points:

Positives

- Has the learner exceeded what has been asked of them?
- Has an aspect of the learners submission surprised you, the grader, in any way? Is the learner's approach unique?
- Does the learner's submission demonstrate a deep understanding of the concepts covered in class?

Negatives

- Is there a specific aspect of the topics covered that you feel the learner should revisit?
- Any recurring mistakes that the learner made?
- Has the student forgotten to include necessary files?

The script didn't work. You need to look at how you defined the path in the file.

You have written defensively with try except blocks.

The application is well programmed.

You have correctly use views, urls, routing and templates.

The code is clean, well formatted and uses appropriate naming strategies.

You have written two tests which isn't enough.

You have commented the code in detail.

I liked the ERD diagram in your report.

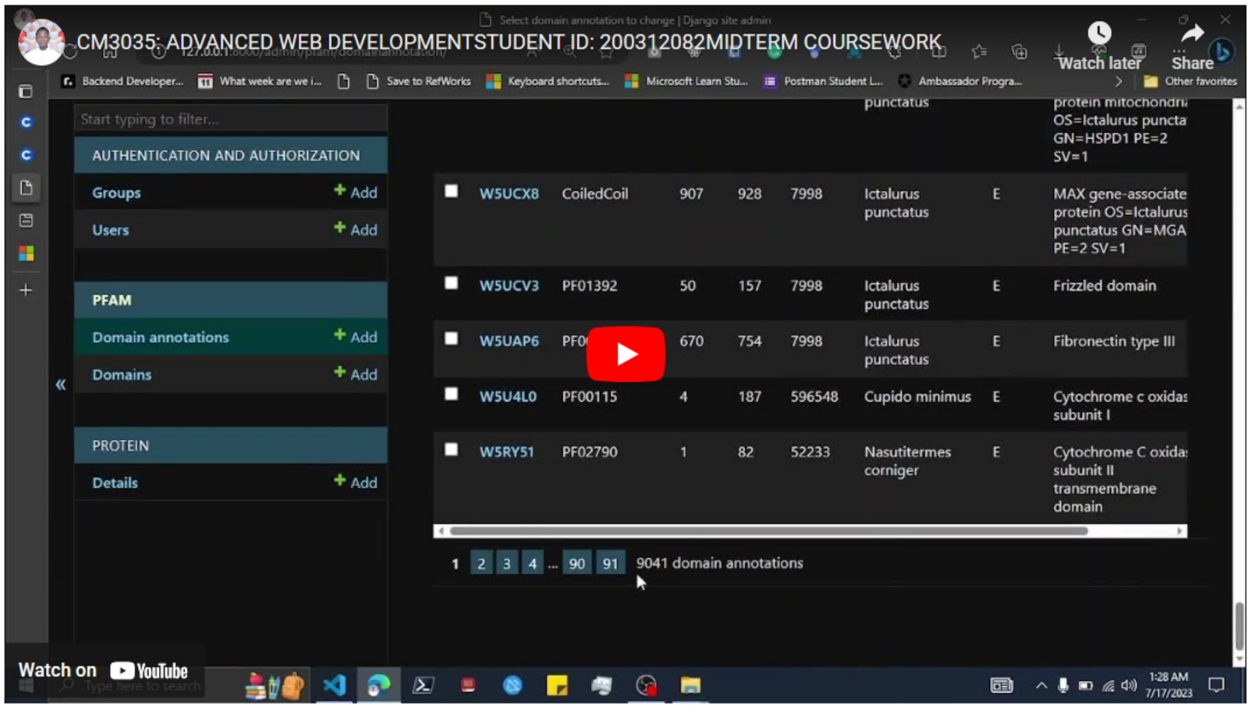
You didn't evaluate your work at the end with what went well, what didn't and what you would do differently if you had more time.

Well done.

3. **Alternative video submission link.**

5 / 5 points

Use this prompt to submit the video demonstrating your application. Remember to keep this under 5 minutes. You could upload the video to youtube or any other channel and submit the address below. Alternatively, you can include the video in the .zip file of your app.



CM3035: ADVANCED WEB DEVELOPMENTSTUDENT ID: 200312082MIDTERM COURSEWORK

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

PFAM
- Domain annotations + Add
- Domains + Add

PROTEIN
- Details + Add

						punctatus		protein mitochondri OS=Ictalurus puncta GN=HSPD1 PE=2 SV=1
☐	W5UCX8	CoiledCoil	907	928	7998	Ictalurus punctatus	E	MAX gene-associate protein OS=Ictalurus punctatus GN=MGA PE=2 SV=1
☐	W5UCV3	PF01392	50	157	7998	Ictalurus punctatus	E	Frizzled domain
☐	W5UAP6	PF0	670	754	7998	Ictalurus punctatus	E	Fibronectin type III
☐	W5U4L0	PF00115	4	187	596548	Cupido minimus	E	Cytochrome c oxidas subunit I
☐	W5RY51	PF02790	1	82	52233	Nasutitermes corniger	E	Cytochrome C oxida subunit II transmembrane domain

1 2 3 4 ... 90 91 9041 domain annotations

Grading Rubric

Submission contains a video

0 points No

[✓ 5 points] Yes