Django Evaluation - Final

Deadline - Within 7 days of the date when it was assigned
Grace period - 1 day
This evaluation has three problem sets:

1. Problem Set 1 - Working with Regex
2. Problem Set 2 - Functional web app
3. Problem Set 3
4. Important Notes
5. Submission Process

Problem Set I - Regex

Repo Link

1. Write a regex to extract all the numbers with orange color background from the below text in italics (Output should be a list).

{"orders":[{"id":1},{"id":2},{"id":3},{"id":4},{"id":5},{"id":6},{"id":7},{"id":8},{"id":9},{"id":10},{"id":11},{"id":648},{"id":649},{"id":650},{"id":651},{"id":652},{"id":653}],"errors":[{"code":3,"message":"[PHP Warning #2] count(): Parameter must be an array or an object that implements Countable (153)"}]}

Ans : \d+

a regex pattern to extract all the numbers in the provided text.

This pattern matches any sequence of one or more digits. To extract all the numbers in the text,you can use a regex engine that supports lookaheads,which allows you to match only digits that are surrounded by a specific pattern.

(?<="orders":\[)[^\]]*\d+[^\]]*(?=\])

This pattern matches any sequence of digits that are inside the "orders" array, ignoring any other characters that may appear before or after the digits. The (?<="orders":\[) positive lookbehind ensures that the pattern is preceded by the "orders":[substring, and the (?=\]) positive lookahead ensures that the pattern is followed by a closing square bracket. The [^\]]* and ([^\]]*) patterns match any characters that are not a closing square bracket, allowing the pattern to match numbers that are surrounded by other characters, such as commas or whitespace.

Code:

```
import re

text =
'{"orders":[{"id":1},{"id":2},{"id":3},{"id":4},{"id":5},{"id":6},{"id":7},{"id":8},{"id":9},{"id":10},{"id":11},{"i
d":648},{"id":649},{"id":650},{"id":651},{"id":652},{"id":653}],"errors":[{"code":3,"message":"[PHP
Warning #2] count(): Parameter must be an array or an object that implements Countable
(153)"}]}'

pattern = r'\d+'

numbers = re.findall(pattern, text)

print(numbers)
```

Output:

```
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '648', '649', '650', '651', '652', '653', '2', '153']
```

Problem Set 2 - A functioning web app with API

I. Please create a website - which should have two components:

Expose all the endpoints using Rest API, with proper permissions, authentication and documentation. Please refer to the image link - https://i.imgur.com/T0ZCO9A.png

1. **Admin Facing** - Where admin user can add an **android app** as well as the number of points - earned by user for downloading the app. (Please do not use the default Django Admin)
2. **User Facing** - where the user can see the apps added by the admin and the points. The user should be able to see the following fields.

- Signup and Login (Feel free to use any package for the same).
- Their Name and Profile
- Points Earned.
- Tasks completed.
- Option to upload a screenshot (which must include drag and drop) for that particular task. (Like if a user downloads a particular app), he can send a screenshot of the open app to confirm that he has indeed downloaded the app.

Problem Set 3

Please answer the below questions:

A. Write and share a small note about your choice of system to schedule periodic tasks (such as downloading a list of ISINs every 24 hours). Why did you choose it? Is it reliable enough; Or will it scale? If not, what are the problems with it? And, what else would you recommend to fix this problem at scale in production?

B. In what circumstances would you use Flask instead of Django and vice versa?

-> When it comes to scheduling periodic tasks, I would choose Celery. Celery is a distributed task queue that allows you to schedule and run periodic tasks, such as downloading a list of ISINs every 24 hours. It is reliable and scalable, making it a great choice for production environments.

Celery uses a broker to send and receive messages between the task queue and the workers. You can use a variety of brokers with Celery, such as RabbitMQ, Redis, or Amazon SQS. This makes it easy to scale up or down depending on your needs.

One potential problem with Celery is that it can be complex to set up and configure, especially if you are new to distributed task queues. Additionally, Celery relies on external dependencies such as brokers and result backends, which can introduce additional complexity and potential points of failure.

To address these issues, I would recommend using a managed service like AWS Elastic Beanstalk or Google Cloud Tasks, which simplify the process of setting up and managing Celery in production.

B. Flask and Django are both popular web frameworks for Python, but they have different strengths and use cases.

I would use Flask for small to medium-sized projects that require flexibility and customization. Flask is a lightweight and modular framework that allows you to pick and choose the components you need, making it easy to tailor to your specific requirements. Flask also has a lower learning curve than Django, which can make it a better choice for smaller projects where speed of development is important.

On the other hand, I would use Django for larger and more complex projects that require a more batteries-included approach. Django comes with a lot of built-in functionality, such as an ORM, authentication, and admin interface, which can help you get up and running quickly. Django also has a robust ecosystem of plugins and packages, making it easy to extend and customize the framework as needed.

Overall, the choice between Flask and Django depends on the specific requirements of your project. If you need a lightweight and flexible framework, Flask may be the better choice. If you need a more fully-featured framework with built-in functionality, Django may be the way to go.

Bonus points

- For good s/w eng practices: e.g. modularity, well documented (comments and README), Normalized db schema, requirements.txt etc. Blow my mind!

Important Notes

- Feel free to Google or Stackoverflow (or even go as far as read a book) to understand anything, but please do NOT copy/paste any code/snippet.
- Finish your assignment before the assigned deadline. If you need to extend the deadline, please make sure you drops us an email.
- Document how you deployed the project (in READMe)

How to Submit

Fill this submission form https://forms.gle/tTUPT9x9rrqK8S7G9 We only consider the submissions through Google Form wef April 15th, 2021. (Before you fill up the form, please read the below four important pointers)

1. Please create a repository in GitLab(Not Github), and add ***@NLEvaluations*** for evaluation (Only maintainer's access), and deploy it to any server and share the link.
2. Please include a short video/screencast - running the evaluator through both the problem statement and your solution (While we will not assess the longer videos negatively, we'd prefer the videos to be under 5 minutes - and it should be very concise and to the point). You can add the link to the Readme of the repository.
3. Is there any way that you can automate the delivery? Wow us and get bonus points for the task.
4. All the answers to the questions (As asked in Part D) should go in ReadMe.