**Problem Set I – Regex:-**

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)"}]}*

A). 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},{"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)"}]}'

regex = r'"id":(\d+)'

numbers = re.findall(regex, text)

print(numbers)

**output:**-

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

**Regular Expression (Regex):-**

The regex pattern used here is '"id":(\d+)'.

**"id":`matches the literal characters"id":`** in the text.

(\d+) is a capturing group that matches one or more digits (\d+), capturing them for extraction.

So, the regex essentially matches "id": followed by one or more digits, capturing only the digits.

**Code Explanation:-**

**import re:-** This imports Python's built-in regular expression module.

**text:-** This variable holds the input text provided in the assignment.

**regex:**- This variable holds the regex pattern explained above.

**re.findall(regex, text):** This function searches the input text (text) for all non-overlapping matches of the regex pattern (regex), returning a list of all matches.

**print(numbers):-** This prints the list of numbers extracted from the text.

So, when you run this code with the provided text, it will extract all the numbers (ids) with the orange color background and return them as a list.

# Problem Set 2 - A functioning web app with API:-

2)A. To create a website based on the requirements you provided, we'll use Django as the web framework. Here's a step-by-step guide to help you get started:

**Setup Django Project:-**

Install Django using pip:

pip install django

Create a new Django project:

Copy code

django-admin startproject myproject

Navigate to the project directory:

bash

cd myproject

Create Django App:

Create a new Django app for your website:

python manage.py startapp myapp

Define Models:

In myapp/models.py, define the necessary models such as User, App, and Task. Here's an example:

python

from django.db import models

class User(models.Model):

name = models.CharField(max\_length=100)

profile = models.CharField(max\_length=100)

points = models.IntegerField(default=0)

tasks\_completed = models.IntegerField(default=0)

class App(models.Model):

name = models.CharField(max\_length=100)

points = models.IntegerField(default=0)

class Task(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

app = models.ForeignKey(App, on\_delete=models.CASCADE)

screenshot = models.ImageField(upload\_to='screenshots/')

**Implement Authentication:-**

Use Django's built-in authentication system or third-party packages like Django Allauth for user authentication and registration.

**Create Views:-**

Define views for admin-facing and user-facing interfaces in myapp/views.py. These views will render HTML templates and handle user interactions.

**Design Templates:-**

Create HTML templates in the myapp/templates directory for the admin and user interfaces. Use Bootstrap for styling and JavaScript for client-side interactions.

**Define URL Patterns:-**

Configure URL patterns in myapp/urls.py to map URLs to views.

**Implement REST API:-**

Use Django Rest Framework to create REST API endpoints for CRUD operations on models.

Define serializers for converting model instances to JSON format.

**Documentation:-**

Document your API endpoints using Swagger or Django Rest Framework's built-in documentation tools.

**Testing and Debugging:-**

Test your website thoroughly to ensure proper functionality and fix any bugs or issues.

**Deployment:-**

Deploy your Django website on a server using platforms like Heroku, AWS, or DigitalOcean.

Set up proper hosting configurations, database connections, and security measures.

**Continuous Maintenance:-**

Regularly maintain and update your website to add new features, fix bugs, and address user feedback.

**Explanation:-**

**Homepage:-**

The homepage serves as the entry point to the website and provides users with an overview of its purpose and functionality.

It typically includes a welcoming message and brief information about the website's features.

Navigation links are provided to access the Admin and User sections of the website.

Login and Signup buttons allow new users to register or existing users to log in.

**Admin Dashboard:-**

This page is accessible only to users with admin privileges and serves as the control panel for managing the website's content.

Admins can add new apps along with their corresponding points using a form.

Existing apps and their points are listed for reference and management.

Admin authentication is implemented to ensure that only authorized users can access the dashboard.

**User Dashboard:-**

The user dashboard is the central hub for registered users, providing them with personalized information and features.

Users can view their profile information, including their name, profile details, points earned, and tasks completed.

A list of apps added by the admin is displayed, along with their respective points, allowing users to see available tasks.

Users can upload screenshots for specific tasks, providing evidence of task completion.

**Signup Page:-**

New users can register for an account on this page by providing necessary information such as username, email, and password.

The signup form typically includes validation to ensure that user-provided data meets required criteria.

Upon successful registration, users are redirected to the login page to access their newly created account.

**Login Page:-**

Registered users can log in to their accounts using their username/email and password.

The login form validates user credentials and authenticates users, granting them access to their personalized dashboard upon successful login.

Failed login attempts trigger appropriate error messages to notify users of any issues with their credentials.

**App Detail Page:-**

This page provides detailed information about a specific app listed on the website, including its name, points, and download link.

Users can learn about the app's requirements and instructions for completing associated tasks.

Clear instructions are provided to guide users through the task completion process, ensuring a smooth user experience.

**Profile Page:-**

Users can view and manage their profile information on this page, including their name, profile details, points earned, and tasks completed.

Options to edit profile information, such as updating the name or profile details, may be provided for user convenience.

**Screenshot Upload Page:-**

Users can upload screenshots as evidence of completing specific tasks, such as downloading an app.

The page features a user-friendly form with drag-and-drop functionality, allowing users to easily upload screenshots from their devices.

Clear instructions are provided on how to capture and upload screenshots to ensure accuracy and completeness of task submissions.

**404 Page:-**

This custom error page is displayed when users attempt to access a page that does not exist or cannot be found.

The page features a friendly message informing users that the requested page could not be found.

Navigation links are provided to guide users back to valid pages within the website, helping them continue their browsing experience.

**API Documentation Page:-**

This page serves as a reference for developers or users interested in integrating with the website's REST API.

Detailed documentation is provided for each API endpoint, including descriptions, parameters, and expected responses.

Developers can use this documentation to understand how to interact with the API and incorporate its functionality into their applications.

These pages collectively form the user interface of the website, providing users with a seamless and intuitive experience while interacting with its features and content. Each page is designed to fulfill specific purposes outlined in the requirements and contribute to the overall functionality and usability of the website.

**CODE:-**

[[package]]

name = "asgiref"

version = "3.7.2"

description = "ASGI specs, helper code, and adapters"

category = "main"

optional = false

python-versions = ">=3.7"

[package.dependencies]

typing-extensions = {version = ">=4", markers = "python\_version < \"3.11\""}

[package.extras]

tests = ["mypy (>=0.800)", "pytest", "pytest-asyncio"]

[[package]]

name = "django"

version = "4.2.2"

description = "A high-level Python web framework that encourages rapid development and clean, pragmatic design."

category = "main"

optional = false

python-versions = ">=3.8"

[package.dependencies]

asgiref = ">=3.6.0,<4"

sqlparse = ">=0.3.1"

tzdata = {version = "\*", markers = "sys\_platform == \"win32\""}

[package.extras]

argon2 = ["argon2-cffi (>=19.1.0)"]

bcrypt = ["bcrypt"]

[[package]]

name = "django-environ"

version = "0.10.0"

description = "A package that allows you to utilize 12factor inspired environment variables to configure your Django application."

category = "main"

optional = false

python-versions = ">=3.5,<4"

[package.extras]

develop = ["coverage[toml] (>=5.0a4)", "furo (>=2021.8.17b43,<2021.9.0)", "pytest (>=4.6.11)", "sphinx (>=3.5.0)", "sphinx-notfound-page"]

docs = ["furo (>=2021.8.17b43,<2021.9.0)", "sphinx (>=3.5.0)", "sphinx-notfound-page"]

testing = ["coverage[toml] (>=5.0a4)", "pytest (>=4.6.11)"]

[[package]]

name = "psycopg2-binary"

version = "2.9.6"

description = "psycopg2 - Python-PostgreSQL Database Adapter"

category = "main"

optional = false

python-versions = ">=3.6"

[[package]]

name = "sqlparse"

version = "0.4.4"

description = "A non-validating SQL parser."

category = "main"

optional = false

python-versions = ">=3.5"

[package.extras]

dev = ["build", "flake8"]

doc = ["sphinx"]

test = ["pytest", "pytest-cov"]

[[package]]

name = "typing-extensions"

version = "4.6.3"

description = "Backported and Experimental Type Hints for Python 3.7+"

category = "main"

optional = false

python-versions = ">=3.7"

[[package]]

name = "tzdata"

version = "2023.3"

description = "Provider of IANA time zone data"

category = "main"

optional = false

python-versions = ">=2"

[metadata]

lock-version = "1.1"

python-versions = "^3.10"

content-hash = "753ecaa732bc7edd6a20ee2313d46366d842c9e5a6e229fe6eb184b68665345c"

[metadata.files]

asgiref = [

{file = "asgiref-3.7.2-py3-none-any.whl", hash = "sha256:89b2ef2247e3b562a16eef663bc0e2e703ec6468e2fa8a5cd61cd449786d4f6e"},

{file = "asgiref-3.7.2.tar.gz", hash = "sha256:9e0ce3aa93a819ba5b45120216b23878cf6e8525eb3848653452b4192b92afed"},

]

django = [

{file = "Django-4.2.2-py3-none-any.whl", hash = "sha256:672b3fa81e1f853bb58be1b51754108ab4ffa12a77c06db86aa8df9ed0c46fe5"},

{file = "Django-4.2.2.tar.gz", hash = "sha256:2a6b6fbff5b59dd07bef10bcb019bee2ea97a30b2a656d51346596724324badf"},

]

django-environ = [

{file = "django-environ-0.10.0.tar.gz", hash = "sha256:b3559a91439c9d774a9e0c1ced872364772c612cdf6dc919506a2b13f7a77225"},

{file = "django\_environ-0.10.0-py2.py3-none-any.whl", hash = "sha256:510f8c9c1d0a38b0815f91504270c29440a0cf44fab07f55942fa8d31bbb9be6"},

]

psycopg2-binary = [

{file = "psycopg2-binary-2.9.6.tar.gz", hash = "sha256:1f64dcfb8f6e0c014c7f55e51c9759f024f70ea572fbdef123f85318c297947c"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-macosx\_10\_9\_x86\_64.whl", hash = "sha256:d26e0342183c762de3276cca7a530d574d4e25121ca7d6e4a98e4f05cb8e4df7"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-macosx\_11\_0\_arm64.whl", hash = "sha256:c48d8f2db17f27d41fb0e2ecd703ea41984ee19362cbce52c097963b3a1b4365"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl", hash = "sha256:ffe9dc0a884a8848075e576c1de0290d85a533a9f6e9c4e564f19adf8f6e54a7"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-manylinux\_2\_17\_i686.manylinux2014\_i686.whl", hash = "sha256:8a76e027f87753f9bd1ab5f7c9cb8c7628d1077ef927f5e2446477153a602f2c"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-manylinux\_2\_17\_ppc64le.manylinux2014\_ppc64le.whl", hash = "sha256:6460c7a99fc939b849431f1e73e013d54aa54293f30f1109019c56a0b2b2ec2f"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl", hash = "sha256:ae102a98c547ee2288637af07393dd33f440c25e5cd79556b04e3fca13325e5f"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-musllinux\_1\_1\_aarch64.whl", hash = "sha256:9972aad21f965599ed0106f65334230ce826e5ae69fda7cbd688d24fa922415e"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-musllinux\_1\_1\_i686.whl", hash = "sha256:7a40c00dbe17c0af5bdd55aafd6ff6679f94a9be9513a4c7e071baf3d7d22a70"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-musllinux\_1\_1\_ppc64le.whl", hash = "sha256:cacbdc5839bdff804dfebc058fe25684cae322987f7a38b0168bc1b2df703fb1"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-musllinux\_1\_1\_x86\_64.whl", hash = "sha256:7f0438fa20fb6c7e202863e0d5ab02c246d35efb1d164e052f2f3bfe2b152bd0"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-win32.whl", hash = "sha256:b6c8288bb8a84b47e07013bb4850f50538aa913d487579e1921724631d02ea1b"},

{file = "psycopg2\_binary-2.9.6-cp310-cp310-win\_amd64.whl", hash = "sha256:61b047a0537bbc3afae10f134dc6393823882eb263088c271331602b672e52e9"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-macosx\_10\_9\_x86\_64.whl", hash = "sha256:964b4dfb7c1c1965ac4c1978b0f755cc4bd698e8aa2b7667c575fb5f04ebe06b"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-macosx\_11\_0\_arm64.whl", hash = "sha256:afe64e9b8ea66866a771996f6ff14447e8082ea26e675a295ad3bdbffdd72afb"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl", hash = "sha256:15e2ee79e7cf29582ef770de7dab3d286431b01c3bb598f8e05e09601b890081"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-manylinux\_2\_17\_i686.manylinux2014\_i686.whl", hash = "sha256:dfa74c903a3c1f0d9b1c7e7b53ed2d929a4910e272add6700c38f365a6002820"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-manylinux\_2\_17\_ppc64le.manylinux2014\_ppc64le.whl", hash = "sha256:b83456c2d4979e08ff56180a76429263ea254c3f6552cd14ada95cff1dec9bb8"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl", hash = "sha256:0645376d399bfd64da57148694d78e1f431b1e1ee1054872a5713125681cf1be"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-musllinux\_1\_1\_aarch64.whl", hash = "sha256:e99e34c82309dd78959ba3c1590975b5d3c862d6f279f843d47d26ff89d7d7e1"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-musllinux\_1\_1\_i686.whl", hash = "sha256:4ea29fc3ad9d91162c52b578f211ff1c931d8a38e1f58e684c45aa470adf19e2"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-musllinux\_1\_1\_ppc64le.whl", hash = "sha256:4ac30da8b4f57187dbf449294d23b808f8f53cad6b1fc3623fa8a6c11d176dd0"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-musllinux\_1\_1\_x86\_64.whl", hash = "sha256:e78e6e2a00c223e164c417628572a90093c031ed724492c763721c2e0bc2a8df"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-win32.whl", hash = "sha256:1876843d8e31c89c399e31b97d4b9725a3575bb9c2af92038464231ec40f9edb"},

{file = "psycopg2\_binary-2.9.6-cp311-cp311-win\_amd64.whl", hash = "sha256:b4b24f75d16a89cc6b4cdff0eb6a910a966ecd476d1e73f7ce5985ff1328e9a6"},

{file = "psycopg2\_binary-2.9.6-cp36-cp36m-win32.whl", hash = "sha256:498807b927ca2510baea1b05cc91d7da4718a0f53cb766c154c417a39f1820a0"},

{file = "psycopg2\_binary-2.9.6-cp36-cp36m-win\_amd64.whl", hash = "sha256:0d236c2825fa656a2d98bbb0e52370a2e852e5a0ec45fc4f402977313329174d"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-macosx\_10\_9\_x86\_64.whl", hash = "sha256:34b9ccdf210cbbb1303c7c4db2905fa0319391bd5904d32689e6dd5c963d2ea8"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl", hash = "sha256:84d2222e61f313c4848ff05353653bf5f5cf6ce34df540e4274516880d9c3763"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-manylinux\_2\_17\_i686.manylinux2014\_i686.whl", hash = "sha256:30637a20623e2a2eacc420059be11527f4458ef54352d870b8181a4c3020ae6b"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-manylinux\_2\_17\_ppc64le.manylinux2014\_ppc64le.whl", hash = "sha256:8122cfc7cae0da9a3077216528b8bb3629c43b25053284cc868744bfe71eb141"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl", hash = "sha256:38601cbbfe600362c43714482f43b7c110b20cb0f8172422c616b09b85a750c5"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-musllinux\_1\_1\_aarch64.whl", hash = "sha256:c7e62ab8b332147a7593a385d4f368874d5fe4ad4e341770d4983442d89603e3"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-musllinux\_1\_1\_i686.whl", hash = "sha256:2ab652e729ff4ad76d400df2624d223d6e265ef81bb8aa17fbd63607878ecbee"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-musllinux\_1\_1\_ppc64le.whl", hash = "sha256:c83a74b68270028dc8ee74d38ecfaf9c90eed23c8959fca95bd703d25b82c88e"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-musllinux\_1\_1\_x86\_64.whl", hash = "sha256:d4e6036decf4b72d6425d5b29bbd3e8f0ff1059cda7ac7b96d6ac5ed34ffbacd"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-win32.whl", hash = "sha256:a8c28fd40a4226b4a84bdf2d2b5b37d2c7bd49486b5adcc200e8c7ec991dfa7e"},

{file = "psycopg2\_binary-2.9.6-cp37-cp37m-win\_amd64.whl", hash = "sha256:51537e3d299be0db9137b321dfb6a5022caaab275775680e0c3d281feefaca6b"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-macosx\_10\_9\_x86\_64.whl", hash = "sha256:cf4499e0a83b7b7edcb8dabecbd8501d0d3a5ef66457200f77bde3d210d5debb"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-macosx\_11\_0\_arm64.whl", hash = "sha256:7e13a5a2c01151f1208d5207e42f33ba86d561b7a89fca67c700b9486a06d0e2"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl", hash = "sha256:0e0f754d27fddcfd74006455b6e04e6705d6c31a612ec69ddc040a5468e44b4e"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-manylinux\_2\_17\_i686.manylinux2014\_i686.whl", hash = "sha256:d57c3fd55d9058645d26ae37d76e61156a27722097229d32a9e73ed54819982a"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-manylinux\_2\_17\_ppc64le.manylinux2014\_ppc64le.whl", hash = "sha256:71f14375d6f73b62800530b581aed3ada394039877818b2d5f7fc77e3bb6894d"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl", hash = "sha256:441cc2f8869a4f0f4bb408475e5ae0ee1f3b55b33f350406150277f7f35384fc"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-musllinux\_1\_1\_aarch64.whl", hash = "sha256:65bee1e49fa6f9cf327ce0e01c4c10f39165ee76d35c846ade7cb0ec6683e303"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-musllinux\_1\_1\_i686.whl", hash = "sha256:af335bac6b666cc6aea16f11d486c3b794029d9df029967f9938a4bed59b6a19"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-musllinux\_1\_1\_ppc64le.whl", hash = "sha256:cfec476887aa231b8548ece2e06d28edc87c1397ebd83922299af2e051cf2827"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-musllinux\_1\_1\_x86\_64.whl", hash = "sha256:65c07febd1936d63bfde78948b76cd4c2a411572a44ac50719ead41947d0f26b"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-win32.whl", hash = "sha256:4dfb4be774c4436a4526d0c554af0cc2e02082c38303852a36f6456ece7b3503"},

{file = "psycopg2\_binary-2.9.6-cp38-cp38-win\_amd64.whl", hash = "sha256:02c6e3cf3439e213e4ee930308dc122d6fb4d4bea9aef4a12535fbd605d1a2fe"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-macosx\_10\_9\_x86\_64.whl", hash = "sha256:e9182eb20f41417ea1dd8e8f7888c4d7c6e805f8a7c98c1081778a3da2bee3e4"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-macosx\_11\_0\_arm64.whl", hash = "sha256:8a6979cf527e2603d349a91060f428bcb135aea2be3201dff794813256c274f1"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl", hash = "sha256:8338a271cb71d8da40b023a35d9c1e919eba6cbd8fa20a54b748a332c355d896"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-manylinux\_2\_17\_i686.manylinux2014\_i686.whl", hash = "sha256:e3ed340d2b858d6e6fb5083f87c09996506af483227735de6964a6100b4e6a54"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-manylinux\_2\_17\_ppc64le.manylinux2014\_ppc64le.whl", hash = "sha256:f81e65376e52f03422e1fb475c9514185669943798ed019ac50410fb4c4df232"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl", hash = "sha256:bfb13af3c5dd3a9588000910178de17010ebcccd37b4f9794b00595e3a8ddad3"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-musllinux\_1\_1\_aarch64.whl", hash = "sha256:4c727b597c6444a16e9119386b59388f8a424223302d0c06c676ec8b4bc1f963"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-musllinux\_1\_1\_i686.whl", hash = "sha256:4d67fbdaf177da06374473ef6f7ed8cc0a9dc640b01abfe9e8a2ccb1b1402c1f"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-musllinux\_1\_1\_ppc64le.whl", hash = "sha256:0892ef645c2fabb0c75ec32d79f4252542d0caec1d5d949630e7d242ca4681a3"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-musllinux\_1\_1\_x86\_64.whl", hash = "sha256:02c0f3757a4300cf379eb49f543fb7ac527fb00144d39246ee40e1df684ab514"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-win32.whl", hash = "sha256:c3dba7dab16709a33a847e5cd756767271697041fbe3fe97c215b1fc1f5c9848"},

{file = "psycopg2\_binary-2.9.6-cp39-cp39-win\_amd64.whl", hash = "sha256:f6a88f384335bb27812293fdb11ac6aee2ca3f51d3c7820fe03de0a304ab6249"},

]

sqlparse = [

{file = "sqlparse-0.4.4-py3-none-any.whl", hash = "sha256:5430a4fe2ac7d0f93e66f1efc6e1338a41884b7ddf2a350cedd20ccc4d9d28f3"},

{file = "sqlparse-0.4.4.tar.gz", hash = "sha256:d446183e84b8349fa3061f0fe7f06ca94ba65b426946ffebe6e3e8295332420c"},

]

typing-extensions = [

{file = "typing\_extensions-4.6.3-py3-none-any.whl", hash = "sha256:88a4153d8505aabbb4e13aacb7c486c2b4a33ca3b3f807914a9b4c844c471c26"},

{file = "typing\_extensions-4.6.3.tar.gz", hash = "sha256:d91d5919357fe7f681a9f2b5b4cb2a5f1ef0a1e9f59c4d8ff0d3491e05c0ffd5"},

]

tzdata = [

{file = "tzdata-2023.3-py2.py3-none-any.whl", hash = "sha256:7e65763eef3120314099b6939b5546db7adce1e7d6f2e179e3df563c70511eda"},

{file = "tzdata-2023.3.tar.gz", hash = "sha256:11ef1e08e54acb0d4f95bdb1be05da659673de4acbd21bf9c69e94cc5e907a3a"},

]

**pyproject.toml:-**

[tool.poetry]

name = "nextlab-django-assignment-evaluation"

version = "0.1.0"

description = ""

authors = ["5P4RK3R <mahendransparker@gmail.com>"]

readme = "README.md"

packages = [{include = "nextlab\_django\_assignment\_evaluation"}]

[tool.poetry.dependencies]

python = "^3.10"

django = "^4.2.2"

psycopg2-binary = "^2.9.6"

django-environ = "^0.10.0"

[build-system]

requires = ["poetry-core"]

build-backend = "poetry.core.masonry.api"

**regex.py:-**

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

import re

inp = {"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)"}]}

op = re.findall(r': \d+',str(inp))

op = map(lambda x:x.replace(": ",""),op)

op = list(op)

print(op)

#Expected o/p: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '648', '649', '650', '651', '652', '653', '3']

3)A. **Problem Set 3:-**

A. **Choice of System for Scheduling Periodic Tasks:-**

For scheduling periodic tasks, such as downloading a list of ISINs every 24 hours, I would choose Celery with Redis or RabbitMQ as the message broker. Celery is a distributed task queue that supports scheduling periodic tasks and executing them asynchronously. Here's why I chose it:

**Reliability:-** Celery is highly reliable and widely used in production environments. It provides fault tolerance, retry mechanisms, and monitoring capabilities, ensuring that tasks are executed reliably.

**Scalability:-** Celery is designed to scale horizontally, allowing tasks to be distributed across multiple worker nodes. It can handle large workloads and scale seamlessly as the application grows.

**Flexibility:-** Celery integrates seamlessly with Python frameworks like Django and Flask, making it easy to incorporate into existing applications.

**Periodic Task Support:-** Celery provides built-in support for scheduling periodic tasks using cron-like syntax, making it ideal for tasks that need to be executed at regular intervals.

However, for large-scale production environments with complex task dependencies and scheduling requirements, a more advanced task scheduling system such as Apache Airflow or Kubernetes CronJobs may be recommended. These systems offer more sophisticated scheduling features, workflow orchestration, and scalability options tailored for enterprise-level deployments.

**B. Choice Between Flask and Django:-**

**Use Flask When**:-

**Microservices Architecture:-** Flask is lightweight and minimalist, making it well-suited for building microservices or small-scale applications where flexibility and simplicity are prioritized.

**Customization:-** Flask offers more freedom and flexibility in terms of project structure and component selection. It allows developers to cherry-pick libraries and extensions based on project requirements.

**RESTful APIs:-** Flask is commonly used for building RESTful APIs due to its simplicity and minimal overhead. It's a popular choice for developing lightweight API services.

**Use Django When:-**

**Full-Stack Web Development:-** Django is a full-featured web framework that comes with built-in components for ORM, authentication, admin interface, and more. It's ideal for rapid development of complex, database-driven web applications.

**Batteries-Included Approach:-** Django follows a batteries-included philosophy, providing a comprehensive set of tools and features out-of-the-box. It reduces the need for external dependencies and simplifies development.

**Convention Over Configuration:-** Django enforces a "batteries-included" approach and follows the "convention over configuration" principle, which promotes consistency and reduces boilerplate code. This makes it suitable for large-scale projects with standardized development practices.

In summary, Flask is preferable for small, lightweight projects or microservices, while Django is more suitable for large-scale web applications requiring a full-stack framework with built-in features and conventions. The choice between Flask and Django depends on the specific requirements, complexity, and scalability considerations of the project.

Certainly! Let's provide some examples and sample syntax for both the choice of system for scheduling periodic tasks and the use of Flask and Django:-

A. Choice of System for Scheduling Periodic Tasks:

Celery with Redis as the Message Broker:

Sample Syntax:

python

Copy code

# tasks.py

from celery import Celery

app = Celery('tasks', broker='redis://localhost:6379/0')

@app.task

def download\_isins():

# Task logic to download ISINs

pass

Explanation:

This example defines a Celery task download\_isins() that can be scheduled to run periodically.

Celery is configured to use Redis as the message broker for task queuing and distribution.

Apache Airflow:

Sample Syntax (DAG Definition):

python

Copy code

from airflow import DAG

from airflow.operators.python\_operator import PythonOperator

from datetime import datetime, timedelta

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': datetime(2024, 1, 1),

'email\_on\_failure': False,

'email\_on\_retry': False,

'retries': 1,

'retry\_delay': timedelta(minutes=5),

}

dag = DAG('download\_isins', default\_args=default\_args, schedule\_interval=timedelta(days=1))

def download\_isins():

# Task logic to download ISINs

pass

download\_task = PythonOperator(

task\_id='download\_task',

python\_callable=download\_isins,

dag=dag,

)

Explanation:

This example defines an Apache Airflow Directed Acyclic Graph (DAG) named download\_isins.

The DAG includes a PythonOperator download\_task that executes the download\_isins() function as a task scheduled to run every day.

B. Choice Between Flask and Django:

Flask Example:

Sample Syntax (Minimal Flask App):

python

Copy code

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

Explanation:

This example defines a minimal Flask application with a single route / that returns the message "Hello, World!".

Flask's @app.route decorator is used to specify the URL endpoint and associated view function.

Django Example:

Sample Syntax (Django View):

python

Copy code

from django.http import HttpResponse

from django.shortcuts import render

def hello\_world(request):

return HttpResponse("Hello, World!")

Explanation:

This example defines a Django view function hello\_world() that returns an HTTP response with the message "Hello, World!".

Views in Django are Python functions or methods that take a web request and return a web response.

These examples provide a glimpse of how each system and framework can be used in practice. Depending on your specific requirements and project context, you can customize and expand upon these examples to build robust and scalable solutions.