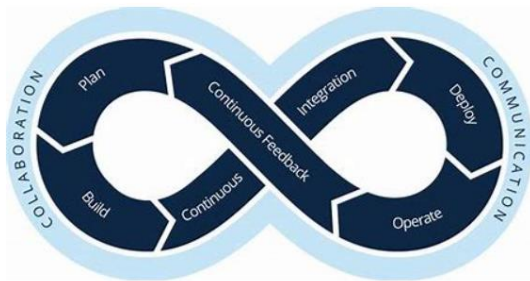
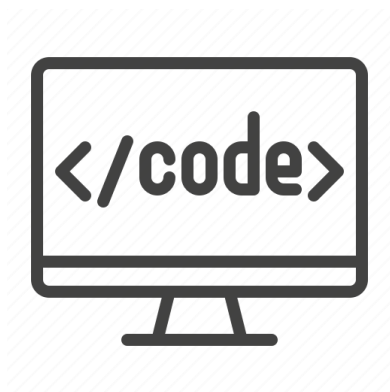


Activity 2.2

Coding Fundamentals: Apply and Prove

(Online and Workplace)



Level 4 DevOps Engineer Apprenticeships

Training Provider: QA Ltd



Company Name: S2R Analytics Ltd

Apprentice Name: Purvi Thakkar

Index

Cover Page	1
Index	2
Python project using Flask and SQL	3-4
<i>Test Driven Development (TDD)</i> _____	3
<i>Steps involved in developing the application</i> _____	3
<i>@app.route explanation</i> _____	4
Agile methodology	4-19
<i>Delivery team</i> _____	5
<i>Project (web page)</i> _____	6
<i>Progress and Iteration</i> _____	7
<i>Continuous learning & Continuous deployment</i> _____	13
<i>Project Artifacts</i> _____	14
<i>User Story</i> _____	14
<i>Product Backlog</i> _____	14
<i>Sprint Board</i> _____	15
<i>Project Outcome</i> _____	16
Tools used	19-20
<i>Front-End</i> _____	19
<i>Back-End</i> _____	19
<i>Database</i> _____	19
<i>Testing</i> _____	19
<i>Visual Collaboration Tool</i> _____	20
<i>Workplace Communication Tool</i> _____	20
<i>Code Hosting Platform</i> _____	20

Python project using Flask and SQL

This Python Flask application provides a **web-based interface** for querying a **SQL database that contains job logs**. The application has several pages, including a home page, an about page, and a contact page. In addition, there are several pages for retrieving job logs, including a page for retrieving the top 20 job logs, a page for retrieving the top 50 job logs, and a page for retrieving the top 100 job logs. There is also a page for retrieving job logs based on a specific job ID, a page for retrieving job logs from the last 24 hours, and a page for retrieving job logs from the last week.

The project will be built using Flask, a **Python web framework**, and **SQL**, a relational database management system. The website's front-end will be created using **HTML**, **CSS**, with the help of **Bootstrap** for responsive design. The website is optimized for desktop and mobile devices using Bootstrap.

The application uses the **Flask web framework to handle HTTP requests** and render HTML templates. The **SQL queries are executed using a cursor object** that is imported from a separate module named "variable". The application uses the **GET and POST methods** to retrieve job logs based on job ID. The application also includes error handling to ensure that the user inputs valid job IDs.

The project has been tested using **pytest**, and **unittest** with test cases for the website's functionality. The project will be deployed to a cloud hosting platform Azure.

Overall, this project will provide a comprehensive introduction to building web applications using Flask and SQL. It will also provide experience with **database design**, **front-end development**, and deployment to a **cloud hosting platform**. This application provides a user-friendly interface for querying job logs from a SQL database, making it easier for S2R employees to retrieve and analyse the data they need.

Test Driven Development (TDD)

Testing is an essential aspect of any project, regardless of size or complexity. It is the process of evaluating a system or application to ensure that it meets its intended requirements and works as expected. Testing helps developers to:

- **Identify defects** or bugs in the system. (Functional or non-functional)
- Improve the **code's quality** by ensuring it meets the customer's requirements as intended.
- Finding defects early in the development cycle can **save significant costs**.
- Ensure the software is **reliable and performs** as expected under different conditions.
- Build **confidence** among stakeholders, users, and project team members.

Steps involved in developing the application:

1. Set up a virtual environment and **install Flask** and other required dependencies.
2. Create a new Flask application and **define the routes** for the web interface.
3. Configure the application to **connect to the SQL database**.
4. Define the database schema and create the required tables.

5. Create a **template for the web interface** using Bootstrap.
 6. Define the view **functions for the different routes** and link them to the corresponding templates.
 7. Implement the **search functionality** for the logs using SQL queries.
 8. **Test the application** thoroughly to ensure it is working as expected.
-

@app.route explanation:

The Flask application is built using Python 3 and the Flask web framework. It consists of several routes that handle HTTP requests and render HTML templates. The routes are defined using the **@app.route()** decorator, and each route corresponds to a specific page on the web interface.

The main route of the application is the home page, which is defined using the **@app.route("/")** decorator. This page displays a simple welcome message and a navigation menu that links to other pages on the web interface.

The other pages on the web interface include the "about" page, the "contact us" page, and several pages for retrieving job logs. Each of these pages is defined using the **@app.route()** decorator and a corresponding HTML template. The HTML templates are stored in the templates folder of the application, and they use Jinja2 templating syntax to render dynamic content.

The pages for retrieving job logs use SQL queries to retrieve data from a SQL database. The database connection and cursor objects are defined in a separate module named "variable", which is imported into the Flask application using the `from variable import cursor` statement. The SQL queries are executed using the **cursor.execute()** method, which takes the SQL query string as an argument. The results of the SQL query are then fetched using the **cursor.fetchall()** method and passed to the corresponding HTML template using the **render_template()** function.

The application also includes error handling to ensure that the user inputs valid job IDs. The job ID is retrieved from the user input using the **request.form.get()** method, and it is validated using Python's built-in **isdigit()** method. If the job ID is invalid, an error message is displayed on the web interface.

Overall, this Flask application provides a simple and intuitive interface for retrieving job logs from a SQL database, making it easier for users to analyze and understand their data.

Agile methodology

Agile methodology is an iterative and collaborative approach to software development that emphasizes flexibility, adaptability, and customer collaboration. I have chosen Scrum as my preferred methodology for this project:

Scrum: Scrum is a popular agile framework for managing and completing complex projects. It involves a cross-functional team working together in short, time-boxed iterations called sprints; in my project, it is a **one-week** sprint. I **focused on delivering a shippable working code** at the end of each sprint.

Scrum emphasizes transparency, inspection, and adaptation through daily stand-up meetings, sprint planning, sprint review, and sprint retrospective meetings.

Delivery team:

- Jonny Gray – Manager: Jonny gave me the project specification and the desired outcome at the end of the project.
- Purvi Thakkar – Project Owner / Developer / Tester
- Simone Roma – Simone was my touch point when I had a few questions regarding the database (on which I was working)
- Rory Patten – Rory was my touch point when I had a few python questions.

Email exchange 1: between my manager and myself.

Hi Jonny,

Can you please give me a python project/exercise on which I can work?

This project will become a part of my main portfolio and is also one of the activities from QA. QA has given the below **Activity outline**.

In this activity, you will:

- Create a Python application that you will subsequently run through a basic CI pipeline.
- Document the project context as well as the design of your application and pipeline.
- Push your project to an appropriate repository.

We recommend that you collaborate with your employer so that you can identify appropriate examples to use. We understand that some data may be sensitive, you are welcome to anonymise this.

Purvi Thakkar | Jr DevOps Engineer

Email exchange 2: between my manager and myself.

Hi Jonny,

Can we have a 10 min call anytime today? I want to share the progress and get insights if I am on the right path.

Purvi Thakkar | Jr DevOps Engineer

Email exchange 3: between my manager and myself.

Subject: Is your project in GitLab?

If not, can you put it in the Prototype group and send me a link to the repo.

Then I can prepare some feedback, recommendations, and the next set of requirements. I would like to build on this and have the App also update the Schedule.

Many thanks,

Jonny

Jonny Gray | Founder & Solutions Lead



Purvi Thakkar

To ● Jonny Gray

Hi Jonny,

The project link is here: <https://gitlab.com/s2ranalytics/prototyping/monitor-queues.git>

Thank you, and I look forward to the feedback and recommendation.

Purvi Thakkar | Jr DevOps Engineer

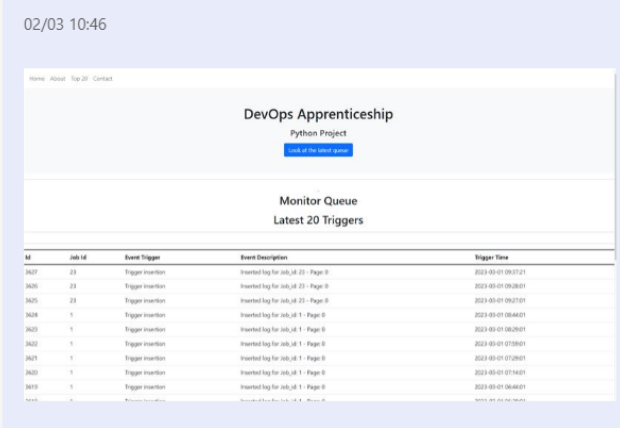
Teams chat: Topic – SQL database name

SR Simone Roma 02/03 10:45
 hi Purvi,
 I'm gonna wipe out the table logs for my tests. But don't worry as I'm gonna duplicate the table with the data and I'll rename it logs2 so you can play with it. As soon as I finish I'll delete logs2 and I'll notify you

is that ok for you?

for the time being you remember to change the table name in your app if you are using it. Otherwise leave it

02/03 10:46



I just managed to crack the code on web app

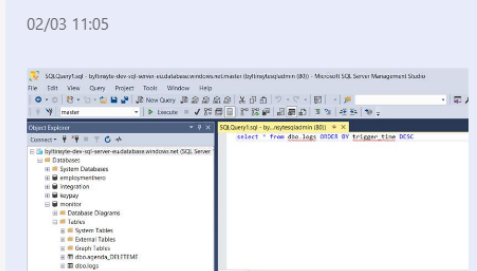
02/03 10:52
 can you please message me once logs2 is created - as i am working on it today.

thank you for the heads-up

SR Simone Roma 02/03 10:53
 ok

done

02/03 11:05



I engaged with my colleagues in **pair programming** as it's an effective way to improve code quality, foster knowledge sharing, speed up problem-solving, reduce errors, and increase accountability. This helped in **strengthening collaboration** and cohesive team culture.

Project (web page):

The objective of this project was to monitor queues from Monitor Database (SQL). My role was to create a flask application that can connect to the SQL database to display the below on a web app:

1. Home page.
2. About page.
3. Top 20 (Top20 displays the **latest 20 queues** from the current time)

4. Top 50 (Top50 displays the **latest 50 queues** from the current time)
5. Top 100 (Top100 displays the **latest 100 queues** from the current time)
6. Job Id (**Job Id takes input** as job_id from the user using the GET method and **displays all the queues** for that job_id using the POST method)
7. Last week (Displays all the queues for the **last week** from the current day/time)
8. Last 24 hours (Displays all queues for the last **24 hours** from the current day/time)
9. Contact page.

Progress and Iteration:

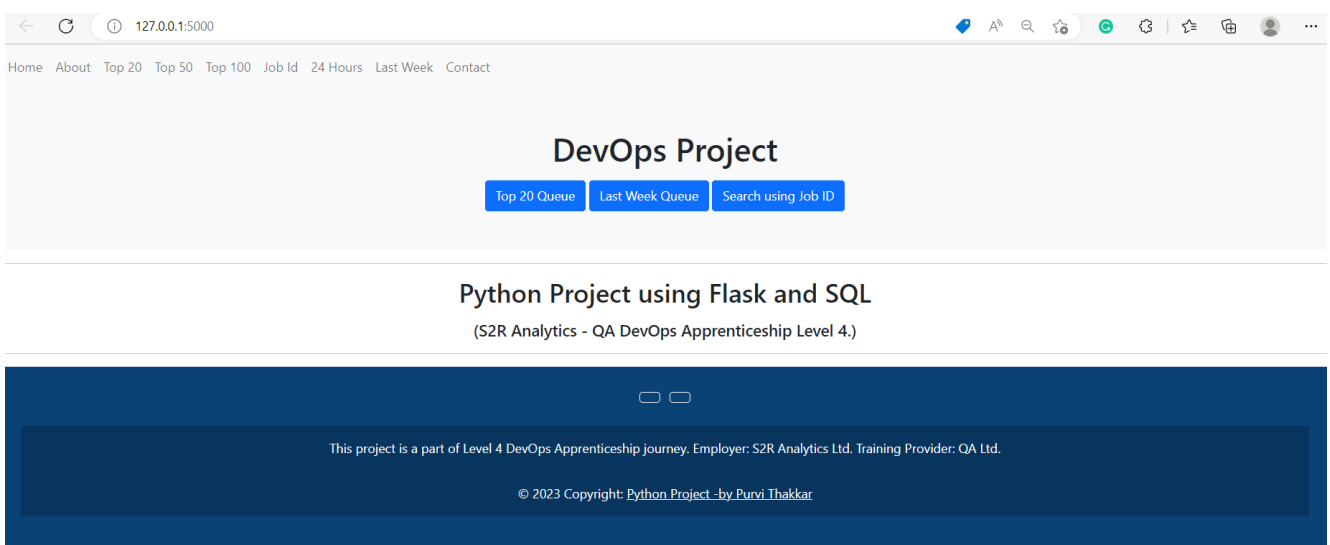
Progress 1: Wrote automated **tests before writing the actual code**, as TDD increases code quality, gives better test coverage, and improved design. By writing tests first, TDD helped in ensuring the code is thoroughly tested and meets the desired requirements. This encouraged me to think more about the design of the code and to create more maintainable and flexible solutions. Below is the failed test:

```
test_app.py
new *
1 def test_home_page():
2     with app.test_client() as client:
3         response = client.get("/")
4         assert response.status_code == 200
5
6 new *
7 @app.route("/")
8 def home():
9     """ Home page function"""
10    return render_template("home.html"), 200
```

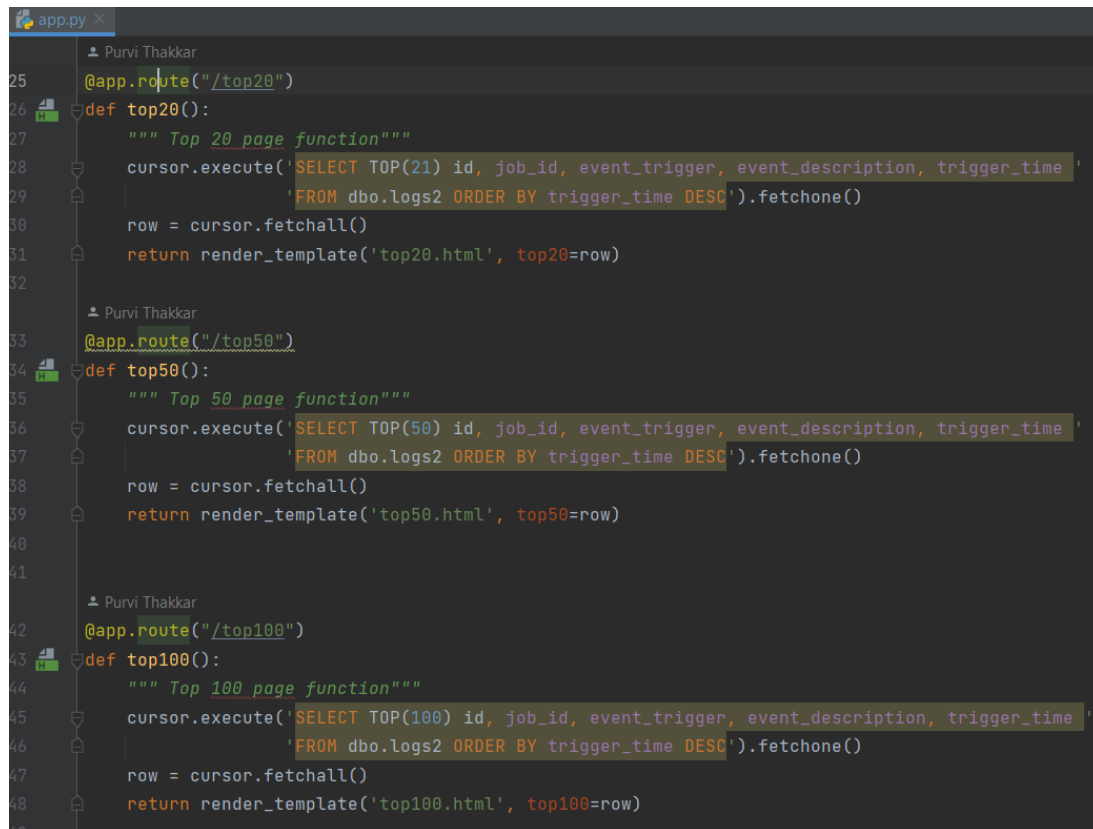
```
test_app.py
new *
11 def test_home_page():
12     with app.test_client() as client:
13         response = client.get("/")
14         assert response.status_code == 200
15         assert b"Welcome to the home page" in response.data
16
```

In this example, I wrote **failing test case** that checks if the home page returns a 200 status code. Then wrote the minimum amount of code necessary to pass this test case, which is to return the home.html template with a 200 status code. Finally, **added another test case to ensure that the template is rendered correctly** and contains the expected text.

Progress 2: Create and run a simple application using flask and display web pages like home, about and contact us.



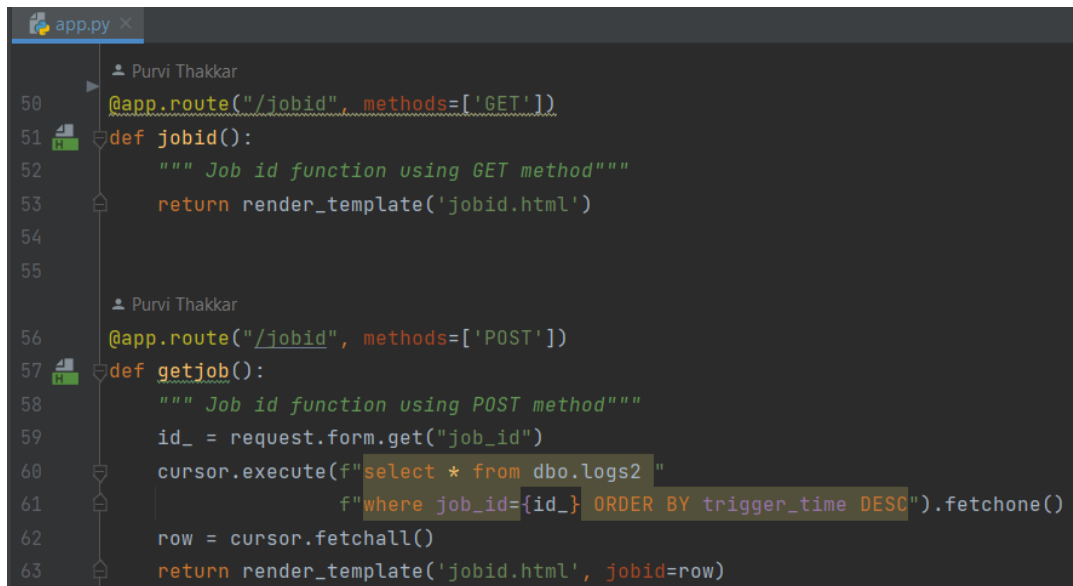
Progress 3: Connect to the database and view the top 20 queues on the Top20 web page.
 Add the Top50 page to show the latest 50 queues from the Monitor database.
 Add the Top100 page to show the latest 100 queues from the Monitor database.



```

25 @app.route("/top20")
26 def top20():
27     """ Top 20 page function """
28     cursor.execute('SELECT TOP(21) id, job_id, event_trigger, event_description, trigger_time '
29                   'FROM dbo.logs2 ORDER BY trigger_time DESC').fetchone()
30     row = cursor.fetchall()
31     return render_template('top20.html', top20=row)
32
33 @app.route("/top50")
34 def top50():
35     """ Top 50 page function """
36     cursor.execute('SELECT TOP(50) id, job_id, event_trigger, event_description, trigger_time '
37                   'FROM dbo.logs2 ORDER BY trigger_time DESC').fetchone()
38     row = cursor.fetchall()
39     return render_template('top50.html', top50=row)
40
41
42 @app.route("/top100")
43 def top100():
44     """ Top 100 page function """
45     cursor.execute('SELECT TOP(100) id, job_id, event_trigger, event_description, trigger_time '
46                   'FROM dbo.logs2 ORDER BY trigger_time DESC').fetchone()
47     row = cursor.fetchall()
48     return render_template('top100.html', top100=row)
  
```

Progress 4: Job Id page added, which uses the GET method to take input from the user and displays the database as per user input using the POST method.



```

50 @app.route("/jobid", methods=['GET'])
51 def jobid():
52     """ Job id function using GET method """
53     return render_template('jobid.html')
54
55
56 @app.route("/jobid", methods=['POST'])
57 def getjob():
58     """ Job id function using POST method """
59     id_ = request.form.get("job_id")
60     cursor.execute(f"select * from dbo.logs2 "
61                   f"where job_id={id_} ORDER BY trigger_time DESC").fetchone()
62     row = cursor.fetchall()
63     return render_template('jobid.html', jobid=row)
  
```

Progress 5: By making **small incremental changes** to the codebase, I could ensure that the software is always up-to-date and optimized for performance, which is essential in today's fast-paced and ever-changing business environment. Below is the screenshot of **teams chat**.
 Topic – add more functionalities on the web app.

02/03 08:49

Sure, in the meantime, I will send you a teams link

02/03 10:48

ID	Job ID	Event Trigger	Event Description	Trigger Time
2427	23	Trigger insertion	Inserted log for job_id: 23 - Page 0	2023-03-01 09:07:07
2426	23	Trigger insertion	Inserted log for job_id: 23 - Page 0	2023-03-01 09:06:57
2425	23	Trigger insertion	Inserted log for job_id: 23 - Page 0	2023-03-01 09:07:03
2424	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 08:44:57
2423	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 08:25:07
2422	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 07:55:07
2421	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 07:25:07
2420	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 07:14:07
2419	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 06:44:07
2418	1	Trigger insertion	Inserted log for job_id: 1 - Page 0	2023-03-01 06:14:07



Jonny Gray 02/03 10:50
Good.

Can you remove teh apprenticeship reference etc. as we would like to be able to use this

02/03 10:50
Sure!



Jonny Gray 02/03 10:51

Then you can think of some other options, so we have a button for top 20 (call it last 20), you can add an option for last 100, you can search for a specific job id or a date range.

Tuesday



Jonny Gray Tuesday 11:41

[How To Use Web Forms in a Flask Application | DigitalOcean](#)



How To Use Web Forms in a Flask Application | ...

Flask is a lightweight Python web framework that provides useful tools and features for creating web applications in...

www.digitalocean.com

Progress/ Iteration 6: I applied a **systematic approach to solving problems** that helped me to achieve reliable and more efficient results. My **unit test kept failing**, and then I tried pytest, which failed a few times.

```

✖ Tests failed: 1 of 1 test - 0 ms

test_app.py::test_home FAILED [100%]
test_app.py:0 (test_home)
pytestconfig = <_pytest.config.Config object at 0x000001C75B86FF10>

    def test_home(pytestconfig):
>     response = pytestconfig.get("/")
E     AttributeError: 'Config' object has no attribute 'get'

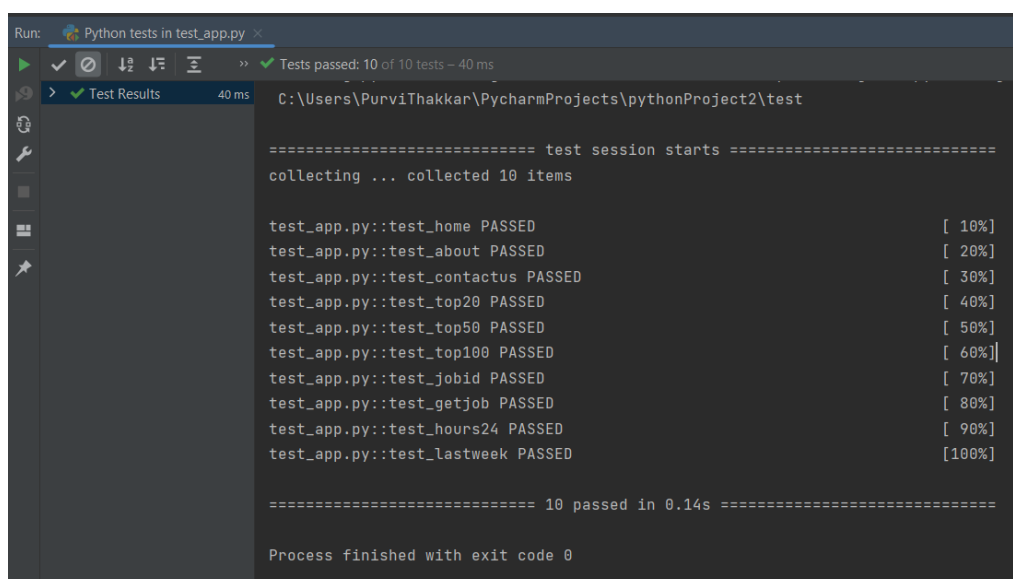
test_app.py:2: AttributeError

===== 1 failed in 0.26s =====

Process finished with exit code 1

```

Later, I googled the error and tried a couple of suggestions, I found on stackoverflow. I applied those suggestions to the code and then finally, the tests were passed.



The screenshot shows the PyCharm interface with the 'Test Results' window open. The window displays the following output:

```

Run: Python tests in test_app.py
>> Tests passed: 10 of 10 tests – 40 ms

> Test Results 40 ms
C:\Users\PurviThakkar\PycharmProjects\pythonProject2\test

===== test session starts =====
collecting ... collected 10 items

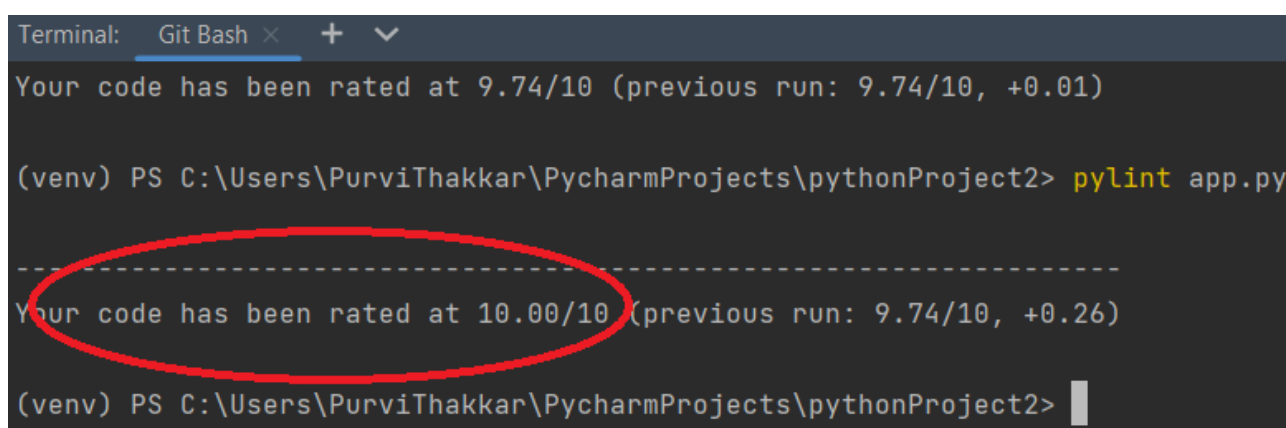
test_app.py::test_home PASSED [ 10%]
test_app.py::test_about PASSED [ 20%]
test_app.py::test_contactus PASSED [ 30%]
test_app.py::test_top20 PASSED [ 40%]
test_app.py::test_top50 PASSED [ 50%]
test_app.py::test_top100 PASSED [ 60%]
test_app.py::test_jobid PASSED [ 70%]
test_app.py::test_getjob PASSED [ 80%]
test_app.py::test_hours24 PASSED [ 90%]
test_app.py::test_lastweek PASSED [100%]

===== 10 passed in 0.14s =====

Process finished with exit code 0

```

Progress 7: Install Pylint and run app.py – fix the issues. Below is the **linting score** rated at **10.00/10**.



The screenshot shows a terminal window with the following output:

```

Terminal: Git Bash x + v
Your code has been rated at 9.74/10 (previous run: 9.74/10, +0.01)

(venv) PS C:\Users\PurviThakkar\PycharmProjects\pythonProject2> pylint app.py

-----
Your code has been rated at 10.00/10 (previous run: 9.74/10, +0.26)
(venv) PS C:\Users\PurviThakkar\PycharmProjects\pythonProject2>

```

The line "Your code has been rated at 10.00/10" is circled in red.

Progress 8: Capture screenshots of all the pages – shared in the project outcome on page number 14.

Progress 9: Continuously push the code to **GitHub** (version control system) this allowed me to manage changes to the code over time. It helped to track modifications to my codebase and easily roll back to a previous version when needed. Each version information gets stored for e.g, what, when, and who made changes to the code. This allowed to see **a history of changes made to the code** and to track the evolution of the project over time.

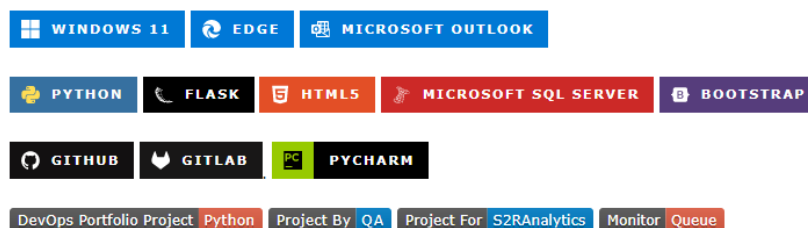
Code repository link: (GitHub)

https://github.com/ThakkarPurvi/QA_Activity_2.2_Python_Project_Monitor_Queue.git

Progress 10: Create a Readme file to display project information.

Python Project using Flask and SQL

(S2R Analytics - QA DevOps Apprenticeship Level 4.)



Aim and Objective of the Project:

- Develop an application using the Flask framework
- Provide a web interface to the processing logs generated by the BytInsyde platform
- Project will use Bootstrap for formatting
- Project will access an SQL database
- Show a list of logs with the ability to search for different criteria
- Live view where it shows incomplete jobs

Assessment criteria with evidence

Program

Files created to run the project successfully.

variable.py

- This file connects the application to the database stored in Microsoft SQL server which has been requested by the user.

```
Database name: Monitor
Table name: dbo.logs 2
```

app.py

- It file will run to display the web application using Flask.

- Home page using function - def home():
- renders home page (designed using bootstrap and HTML)

- About page using function - def about():
- renders about page (designed using bootstrap and HTML)

- Top 20 page using function - def top20():
- renders Top 20 queues (designed using bootstrap and HTML intergates to databases to fetch live data)

- Top 50 page using function - def top50():
- renders Top 50 queues (designed using bootstrap and HTML intergates to databases to fetch live data)

- Top 100 page using function - `def top100():`
- renders Top 100 queues (designed using bootstrap and HTML intergates to databases to fetch live data)
- Job Id page using function - `def jobid():` Using GET method
- User is prompted to share his preference for job id (designed using bootstrap and HTML intergates to databases to fetch live data)
- Job Id page using function - `def getjob():` Using Post method
- displays queues as per user choice of job id (designed using bootstrap and HTML intergates to databases to fetch live data)
- Hours 24 page using function - `def hours24():`
- renders latest 24 hours queues (designed using bootstrap and HTML intergates to databases to fetch live data)
- Last week page using function - `def lastweek():`
- renders all last week's queues (designed using bootstrap and HTML intergates to databases to fetch live data)

Tech Stack

Front-end :

- [HTML](#) — to style the web page
- [CSS](#) — to style the web page
- [Bootstrap](#) — for frontend design, responsive and mobile-first website.

Back-end :

- Python - version 3.10
 - [PyCharm](#) — to run the application
- [Flask](#) — base for everything.

DataBase :

- [MicrosoftSQLServer](#) — Workbench to design, model, generate and manage database.

Testing :

- [PyTest](#) — pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

Workplace Communication Tool:

- [Microsoft Teams](#) — to communicate with your team

Code Hosting Platform:

- [GitHub](#) — for project submission.
- [GitLab](#) — collaboration with team members and version control.

This Python Project is designed as a part of the DevOps Apprenticeship Level 4 in Feb 2023

Profile

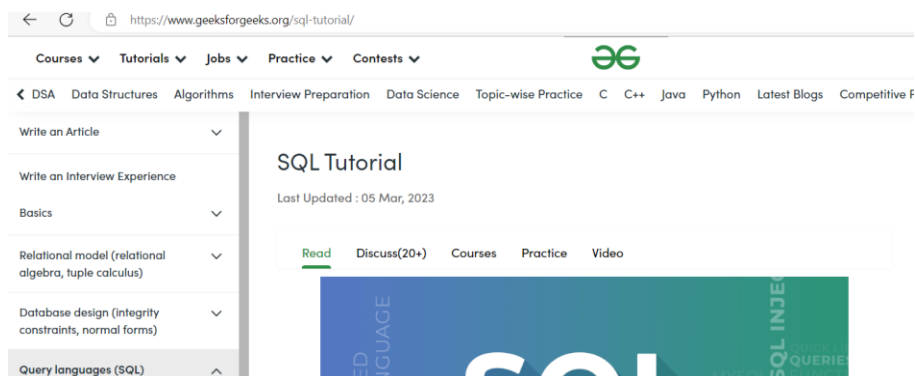
- Employer Name: [S2R Analytics Ltd.](#)
- Training Provider: [QA Ltd.](#)
- Authored by : [Purvi Thakkar](#)



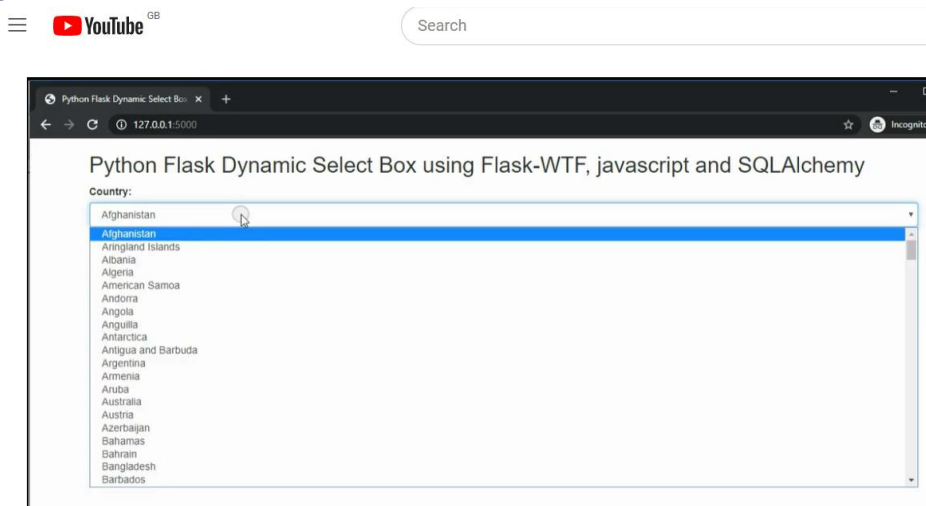
Continuous learning & Continuous deployment

Continuous learning is an essential component of Agile as it enables me to adapt to latest changes in technology, improve skills and knowledge, and **continuously improve** the processes and products.

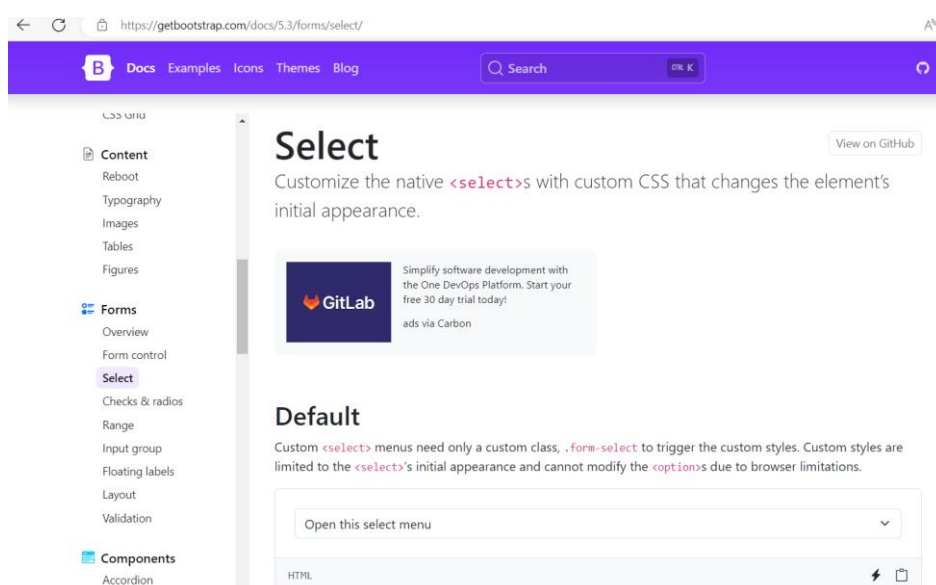
So, my goal was to **learn new tools** as a part of this project and **apply the learnings** towards the success of this project. **Tutorials from geekforgeeks**



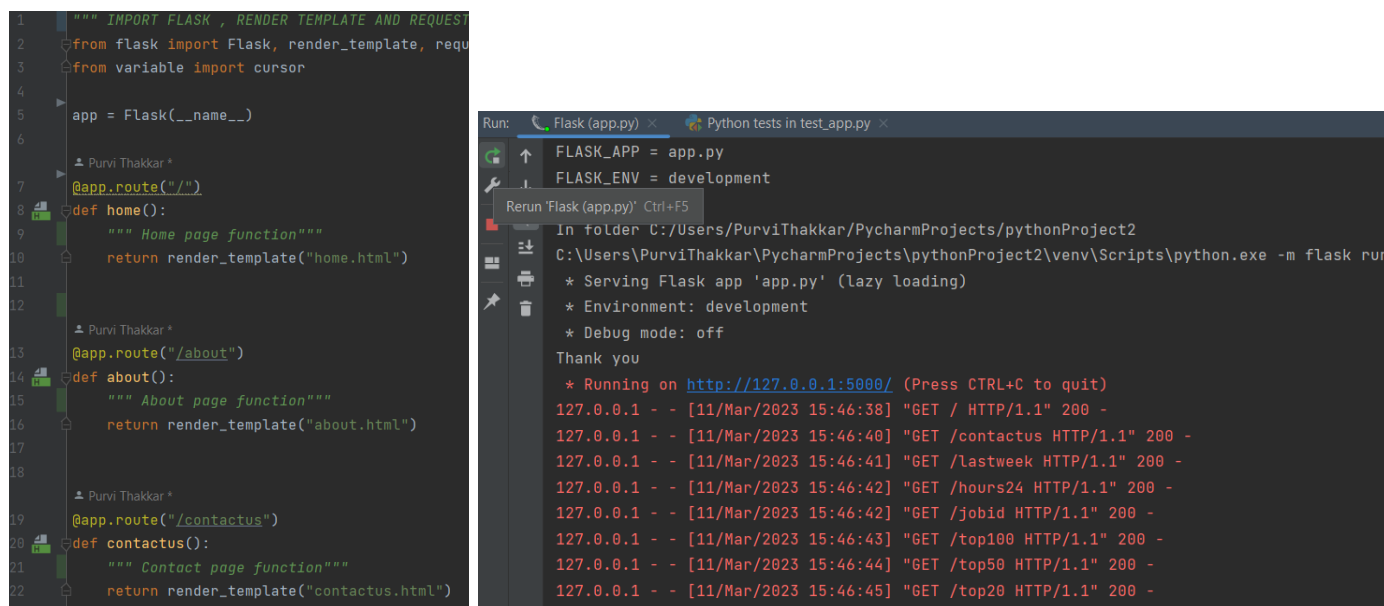
YouTube Videos:



Bootstrap:



In Agile, building and running are done **iteratively and continuously** to adapt to changing requirements and deliver high-quality software products. **Continuous deployment** is important to deliver software products faster, with higher quality, and in a more collaborative and iterative manner. Below is my application and its execution. (app.py)



The image shows a code editor on the left and a terminal window on the right. The code editor displays a Flask application (app.py) with the following code:

```

1  """ IMPORT FLASK , RENDER TEMPLATE AND REQUEST
2  from flask import Flask, render_template, request
3  from variable import cursor
4
5  app = Flask(__name__)
6
7  @app.route("/")
8  def home():
9      """ Home page function"""
10     return render_template("home.html")
11
12
13  @app.route("/about")
14  def about():
15      """ About page function"""
16      return render_template("about.html")
17
18
19  @app.route("/contactus")
20  def contactus():
21      """ Contact page function"""
22      return render_template("contactus.html")

```

The terminal window shows the command to run the application and the output:

```

Run: Flask (app.py) x Python tests in test_app.py x
FLASK_APP = app.py
FLASK_ENV = development
Rerun 'Flask (app.py)' Ctrl+F5
In folder C:/Users/PurviThakkar/PycharmProjects/pythonProject2
C:\Users\PurviThakkar\PycharmProjects\pythonProject2\venv\Scripts\python.exe -m flask run
* Serving Flask app 'app.py' (lazy loading)
* Environment: development
* Debug mode: off
Thank you
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Mar/2023 15:46:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:40] "GET /contactus HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:41] "GET /lastweek HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:42] "GET /hours24 HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:42] "GET /jobid HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:43] "GET /top100 HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:44] "GET /top50 HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2023 15:46:45] "GET /top20 HTTP/1.1" 200 -

```

Project artifacts:

The items on the Product Backlog are often referred to as "User Stories," which are short, simple **descriptions of a feature** from a user's perspective.

- User Story
- Project Backlog
- Sprint Board

User story:

As a user, I want to view the latest triggers online quickly and easily,

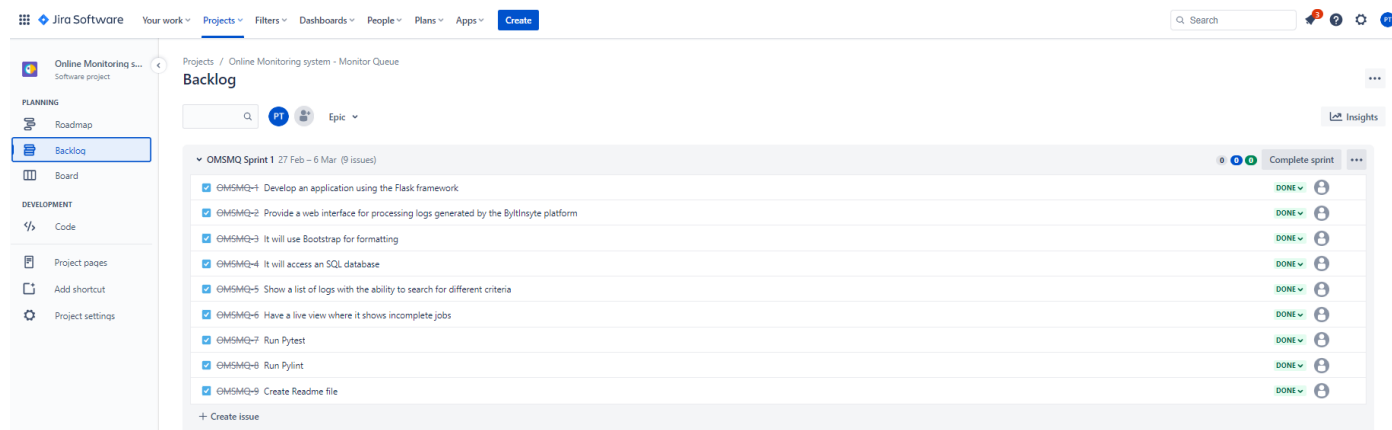
So that I can save time and avoid the hassle of going to the portal and tracking the latest queues.

In this user story, the users are the **employees of S2R Analytics** who need to track the latest triggers (queues). The user's **goal is to save time and avoid the inconvenience** of going to the database every time just to check the latest queue. This user story would be used to guide the development of a web app that meets the needs of busy employees.

Product Backlog:

The Product Backlog is essentially a dynamic, ordered list of items or features that need to be developed for a project. I have created the below Product Backlog; I am responsible **for prioritizing the items based on business value** and market needs.

The Product Backlog items are constantly evolving throughout the project and serves as a living document that is regularly reviewed and updated. The purpose of the Product Backlog is to **provide a clear roadmap** of what needs to be accomplished to deliver a successful project. By prioritizing items based on value and need, I can focus on delivering the most important features first, while ensuring that the product remains aligned with the overall business objectives. Below is the screenshot of my **project Product Backlog**.



Sprint Board:

The Sprint Board provides a way for the development team to track the progress of the current Sprint, which is a **time-boxed iteration** that typically lasts 1-4 weeks. For my project **1 week sprint is designed**.

The Sprint Board typically consists of a physical or virtual board that is divided into columns, with each column representing a specific stage in the development process. The columns in my sprint board include **To Do**, **In Progress**, and **Done**. It can be customized to reflect the specific needs of the team.

The Sprint Board is used **to track the progress my project's user stories** and tasks throughout the Sprint. The user stories are represented by sticky notes or cards, with each card representing a single user story or feature. The cards are moved through the columns as they progress through the development process, with the goal of moving them all the way to the Done column by the end of the Sprint.

The Sprint Board is a useful tool for the development team because it provides **a clear, visual representation of the work that needs to be done**, and the progress that has been made. It also helps **to identify any bottlenecks** or issues in the development process, allowing the team to address them quickly and effectively. Below is the screenshot of my project Sprint Board.

The screenshot displays the Jira Software interface for a project named 'Online Monitoring system - Monitor Queue'. The main view is 'OMSMQ Sprint 1', which is a Kanban board. The board is divided into three columns: 'TO DO', 'IN PROGRESS', and 'DONE 9 ISSUES'. The 'DONE' column contains 9 tasks, all of which are marked as completed with a green checkmark. The tasks are as follows:

- Provide a web interface for processing logs generated by the Bytlnsyte platform (OMSMQ-2)
- It will use Bootstrap for formatting (OMSMQ-3)
- It will access an SQL database (OMSMQ-4)
- Show a list of logs with the ability to search for different criteria (OMSMQ-5)
- Have a live view where it shows incomplete jobs (OMSMQ-6)
- Run Pytest (OMSMQ-7)
- Run Pylint (OMSMQ-8)
- Create Readme file (OMSMQ-9)

The left sidebar shows the project navigation menu, including 'Online Monitoring s...', 'PLANNING' (Roadmap, Backlog, Board), 'DEVELOPMENT' (Code), 'Project pages', 'Add shortcut', and 'Project settings'. At the bottom of the sidebar, it states 'You're in a team-managed project' with a 'Learn more' link.

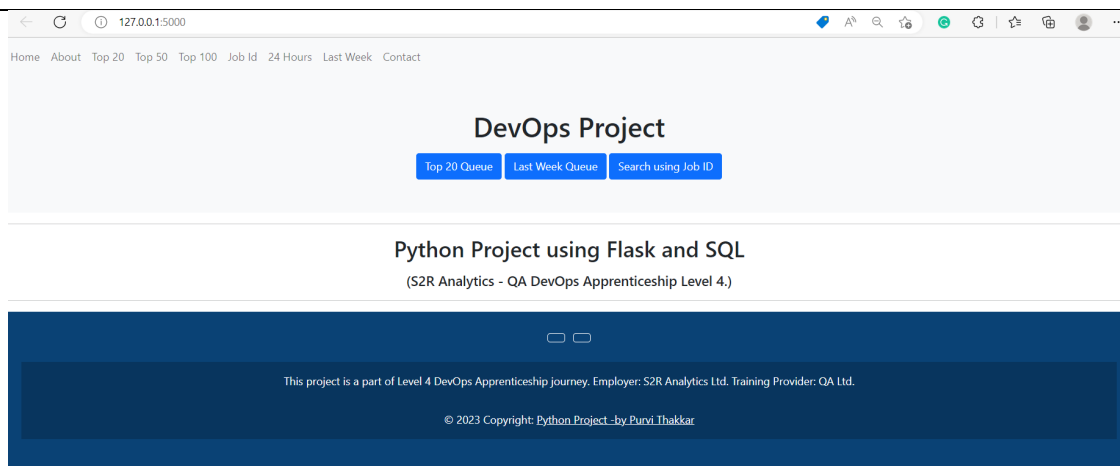
Project Outcome:

This project **saved time and effort to track the latest triggers** and search using the job id as per the **user's choice**. Developers don't have to always go into the database to view the latest triggers, they can simply view from the website.

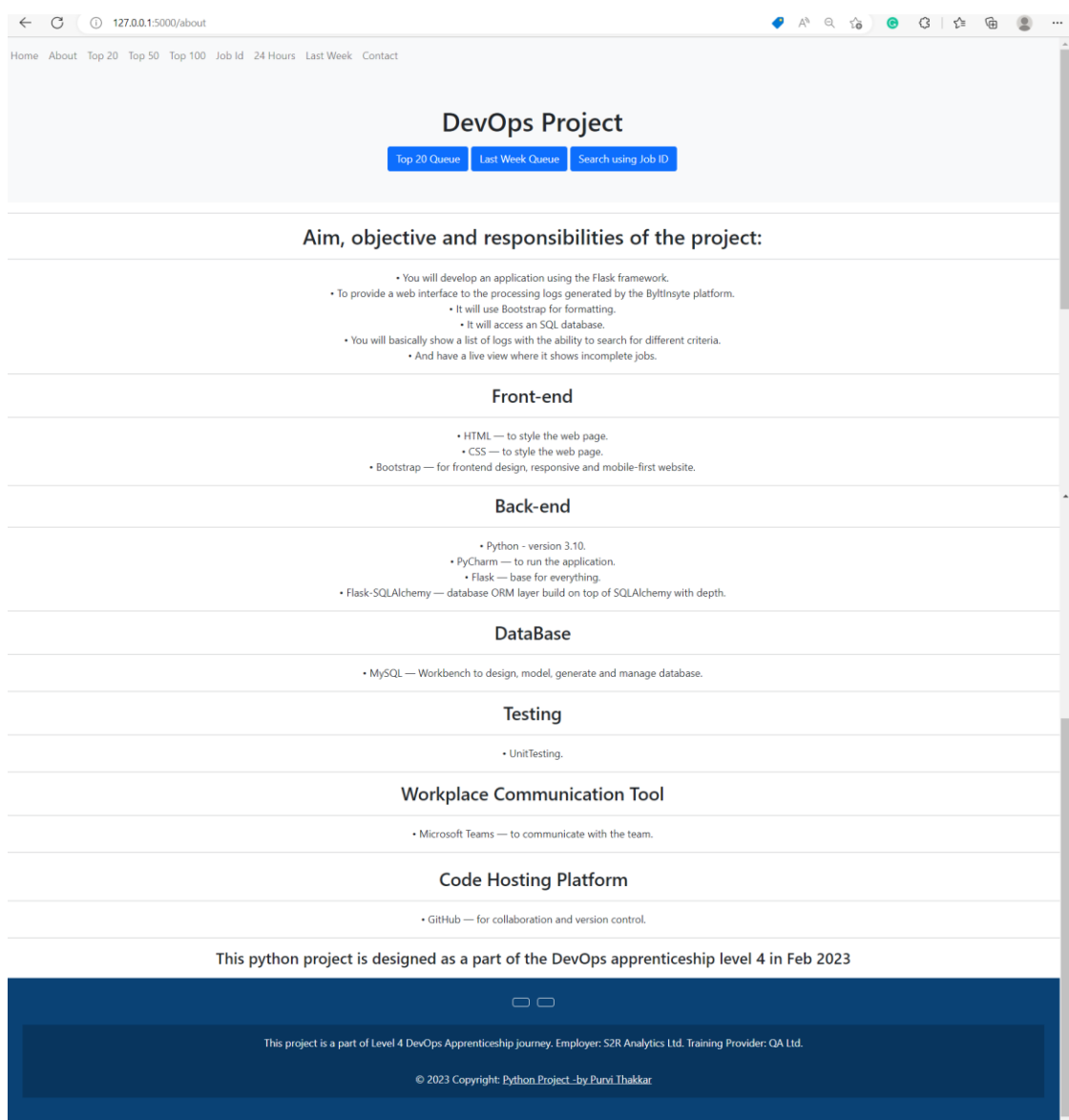
Code repository link: (GitHub)

https://github.com/ThakkarPurvi/QA_Activity_2.2_Python_Project_Monitor_Queue.git

Home Page:



About Page:



Top 20
page:

← 127.0.0.1:5000/top20

Home About Top 20 Top 50 Top 100 Job Id 24 Hours Last Week Contact

DevOps Project

Top 20 Queue Last Week Queue Search using Job ID

Monitor Queue

Latest 20 Triggers

Id	Job Id	Event Trigger	Event Description	Trigger Time
3634	5	trigger	inserting	2023-03-07 14:57:01.163000
3633	4	trigger	inserting	2023-03-07 14:56:54.990000
3632	3	trigger	inserting	2023-03-07 14:56:47.190000
3631	2	trigger	inserting	2023-03-06 15:04:14.013000
3630	2	trigger	inserting	2023-03-06 15:03:51.277000
3628	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 10:07:14
3627	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:37:21
3626	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:28:01
3625	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:27:01
3624	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 08:44:01
3623	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 08:29:01
3622	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 07:59:01
3621	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 07:29:01
3620	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 07:14:01
3619	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 06:44:01
3618	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 06:29:01
3617	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 06:14:01
3616	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 05:44:01
3615	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 05:29:01
3614	1	Trigger insertion	Inserted log for Job_id: 1 - Page: 0	2023-03-01 05:14:01

End of 20 trigger queue

This project is a part of Level 4 DevOps Apprenticeship journey. Employer: S2R Analytics Ltd. Training Provider: QA Ltd.

© 2023 Copyright: Python Project - by Purvi Thakkar

Job id
page:

← 127.0.0.1:5000/jobid

Home About Top 20 Top 50 Top 100 Job Id 24 Hours Last Week Contact

DevOps Project

Top 20 Queue Last Week Queue Search using Job ID

Monitor Queue

Enter Job ID: Load

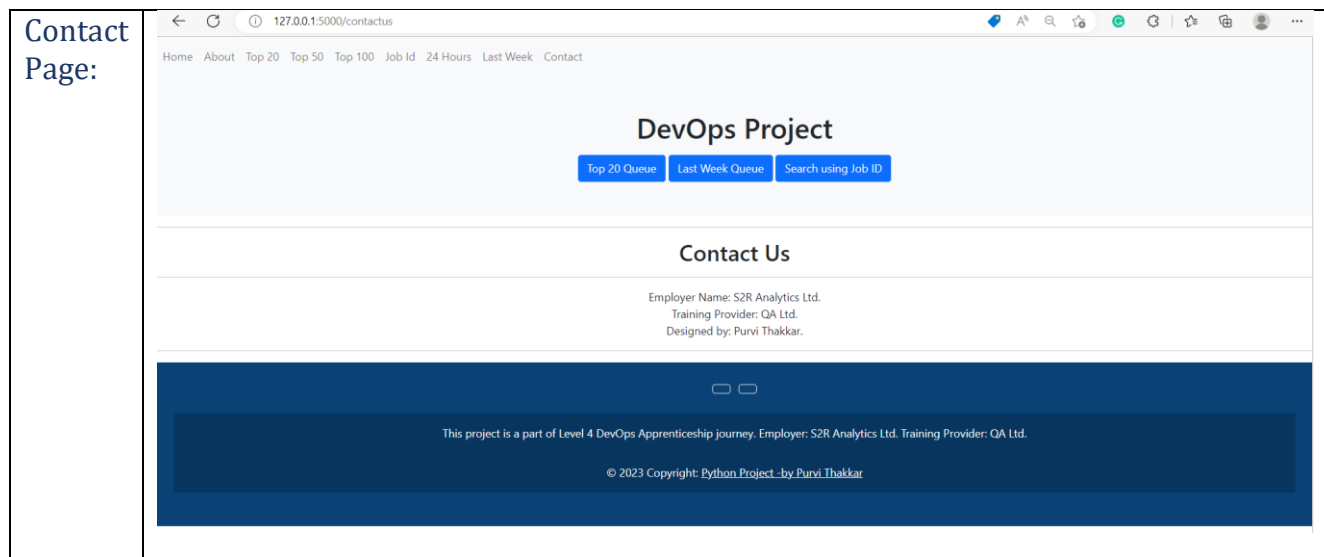
Select your search criteria: Using Job ID

Id	Job Id	Event Trigger	Event Description	Trigger Time
3627	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:37:21
3626	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:28:01
3625	23	Trigger insertion	Inserted log for Job_id: 23 - Page: 0	2023-03-01 09:27:01

End of job id trigger

This project is a part of Level 4 DevOps Apprenticeship journey. Employer: S2R Analytics Ltd. Training Provider: QA Ltd.

© 2023 Copyright: Python Project - by Purvi Thakkar



Tools used:

Front-end:

- [HTML](<https://html.com/html5/>) — to style the web page
- [CSS](<https://www.w3schools.com/Css/>) — to style the web page
- [Bootstrap](<https://getbootstrap.com/>) — for frontend design, responsive and mobile-first website.

Back-end:

- Python version 3.10
[PyCharm](<https://www.jetbrains.com/pycharm/>) — to run the application.
- [Flask](<https://flask.palletsprojects.com/en/2.1.x/>) — base for everything.
- [Flask-SQLAlchemy](<https://flask-sqlalchemy.palletsprojects.com/en/2.x/>) — database

Database:

- [MicrosoftSQLServer](<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>) — Workbench to design, model, generate and manage database.

Testing:

- UnitTesting
- Linting — Linting is the process of using static code analysis tool identify stylistic errors in your code.
- [PyTest] (<https://docs.pytest.org/en/7.2.x/>) — pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

Visual Collaboration Tool:

- [Jira Board](<https://jira.atlassian.com>) — to monitor and track progress throughout the project's lifecycle.

Workplace Communication Tool:

- [Teams](<https://www.microsoft.com/microsoft-teams>) — to communicate with your team

Code Hosting Platform:

- [GitHub](<https://github.com>) — for project submission.
 - [GitLab](<https://gitlab.com/>) — collaboration with team members and version control.
-

Thank you

Employer Name: S2R Analytics Ltd.

Training Provider: QA Ltd.

Code and Design by: Purvi Thakkar

Date: Feb 2023
