**Project Documentation**

|  |
| --- |
| **Name: Rajat Soni** |
| **Collage Name: Poornima University, Jaipur (Rajasthan), 302001** |
| **Branch & Year: B.Tech [Computer Science & Engineering] / 4th Year** |
| **Project Guide: Mr. Bipin Bihari** |
| **Department: IT&C (MES)** |
| **Project Date & Duration: 1 May – 31 July** |
| **Contact No.:6350089790** |
| **Email Id: girrajs967@gmail.com** |

1. **Auto Mail Trigger System**

**Project Description:**

The **Auto Mail Trigger System** is a Python-based script designed to automate the process of notifying concerned authorities via email when a quality check record is due in the PostgreSQL database.

**Key Features:**

* **Database Integration:** Connects to a PostgreSQL database and fetches data from a quality\_check table.
* **Due Check Logic:** Checks whether any records in the table are due based on the current date and predefined conditions.
* **Email Notification:** Automatically sends an email to the concerned authority when a due record is found.
* **Deployment & Scheduling:**
  + The script is deployed on a dedicated server.
  + A **cron job** is configured to execute the script at scheduled intervals (e.g., daily at 9:00 AM).

**Technology Stack:**

* **Language:** Python
* **Database:** PostgreSQL
* **Email:** SMTP (e.g., Gmail, Outlook)
* **Scheduler:** Cron Job (Linux)

1. **Python Backend Service with CI/CD Pipeline**

**Project Description:**

This project is a Python backend service that provides JSON-formatted data from a database through an API endpoint. It is containerized using Docker and deployed using a Jenkins-based CI/CD pipeline.

**Key Features:**

* **Backend Service:**
  + Developed in Python (e.g., Flask or FastAPI).
  + Connects to a database and fetches data.
  + Serves the data via an API endpoint in JSON format.
* **Containerization:**
  + The entire application is containerized using **Docker** for consistency across environments.
* **CI/CD Pipeline:**
  + **Source Control:** Code is maintained in Git and integrated via Git SCM in Jenkins.
  + **Pipeline Steps:**
    1. Pull the latest code from the Git repository.
    2. Build and test the Docker container.
    3. Deploy to the **Quality Server**.
    4. If the deployment is successful, proceed to deploy on the **Production Server**.
    5. Send email notifications to the concerned person for each stage of the deployment (e.g., build success/failure, deployment status).
* **Automation:** Fully automated Jenkins pipeline ensures minimal manual intervention.

**Technology Stack:**

* **Language:** Python
* **Framework:** Flask or FastAPI
* **Containerization:** Docker
* **CI/CD Tool:** Jenkins
* **Source Control:** Git
* **Notification:** Email Alerts
* **Servers:** Quality Server, Production Server

1. **Spring Boot Backend Service with CI/CD Pipeline**

**Project Description:**

This project is a **Java-based backend service** developed using **Spring Boot** and **Maven**. It connects to a database, fetches data, and serves the data via REST endpoints in **JSON format**. The application is containerized with **Docker** and deployed via a **Jenkins CI/CD pipeline**.

**Key Features:**

**🔗 Backend Service:**

* Developed using **Spring Boot** with RESTful architecture.
* Connects to a **relational database** (e.g., PostgreSQL or MySQL).
* Fetches data using **Spring Data JPA** or **JDBC Template**.
* Serves data in **JSON format** through REST API endpoints.

**🐳 Containerization:**

* The complete application is **containerized using Docker**.
* Dockerfile is used to build lightweight, portable containers for development, testing, and production environments.

**🔁 CI/CD Pipeline:**

* **Source Control**: Code is maintained in a **Git** repository.
* **Jenkins Pipeline**:
  1. Pulls the latest code from Git (using Git SCM).
  2. Builds the project using **Maven** (mvn clean install).
  3. Builds the **Docker image**.
  4. Deploys the application to the **Quality Server**.
  5. On success, promotes and deploys the build to the **Production Server**.
  6. Sends **email notifications** to relevant stakeholders after each stage (e.g., build success/failure, deployment completion).

**🔔 Automation:**

* The entire process, from code fetch to production deployment, is automated using **Jenkins Pipelines**.
* Notifications keep the team updated with minimal manual intervention.

**Technology Stack:**

| **Component** | **Technology** |
| --- | --- |
| Language | Java |
| Framework | Spring Boot |
| Build Tool | Maven |
| Database | PostgreSQL / MySQL |
| ORM | Spring Data JPA |
| REST API | Spring MVC |
| Containerization | Docker |
| CI/CD | Jenkins |
| Source Control | Git |
| Notification | Email (SMTP or Jenkins Email Plugin) |
| Servers | Quality Server, Production Server |

**High-Level Architecture:**

+------------------------+

| Git Source Repository |

+-----------+------------+

|

Jenkins CI/CD Pipeline

|

+---------+----------+

| Build with Maven |

+---------------------+

|

+---------v----------+

| Docker Build/Image |

+---------+----------+

|

+---------v---------------------+

| Deploy to Quality Environment |

+------------------------------+

|

(If successful)

|

+---------v-----------------------+

| Deploy to Production Environment |

+--------------------------------+

|

+---------v--------+

| Send Notifications |

+-------------------+

## ****Conclusion:****

This Spring Boot project replicates the data-fetching and JSON-serving logic of the original Python service, but with an enterprise-ready Java tech stack and robust CI/CD practices. It ensures high reliability, easy deployment, and maintainable backend architecture.

**Attendance Sheet**

|  |  |  |
| --- | --- | --- |
| **Date** | **IN Time** | **Out Time** |
| **08-05-2025** | **08:50** | **06:05** |
| **09-05-2025** | **08:34** | **06:20** |
| **12-05-2025** | **08:45** | **06:34** |
| **13-05-2025** | **08:26** | **06:30** |
| **14-05-2025** | **08:23** | **06:35** |
| **15-05-2025** | **09:00** | **06:10** |
| **16-05-2025** | **08:22** | **06:11** |
| **19-05-2025** | **08:41** | **06:09** |
| **20-05-2025** | **08:41** | **06:12** |
| **21-05-2025** | **08:52** | **06:05** |
| **22-05-2025** | **08:21** | **06:03** |
| **23-05-2025** | **08:29** | **06:00** |
| **24-05-2025** | **08:58** | **06:18** |
| **26-05-2025** | **08:30** | **06:15** |
| **27-05-2025** | **08:34** | **06:05** |
| **28-05-2025** | **08:28** | **06:02** |
| **29-05-2025** | **08:32** | **06:10** |
| **30-05-2025** | **08:46** | **06:20** |
| **31-05-2025** | **08:58** | **06:18** |
| **02-06-2025** | **08:34** | **06:03** |
| **03-06-2025** | **08:45** | **06:11** |
| **04-06-2025** | **08:49** | **06:30** |
| **05-06-2025** | **08:51** | **06:20** |
| **06-06-2025** | **08:40** | **06:05** |
| **07-06-2025** | **08:58** | **06:20** |
| **09-06-2025** | **08:51** | **06:01** |
| **10-06-2025** | **08:55** | **06:09** |
| **18-06-2025** | **08:55** | **06:11** |
| **19-06-2025** | **08:43** | **06:05** |
| **20-06-2025** | **08:58** | **06:17** |
| **21-06-2025** | **09:30** | **01:00** |
| **23-06-2025** | **08:51** | **06:19** |
| **24-06-2025** | **08:59** | **06:30** |
| **25-06-2025** | **08:50** | **06:20** |
| **26-06-2025** | **09:00** | **06:15** |
| **28-06-2025** | **09:02** | **06:30** |
| **30-06-2025** | **08:53** | **06:29** |
| **01-07-2025** | **08:50** | **06:05** |
| **02-07-2025** | **08:59** | **06:40** |
| **03-07-2025** | **09:00** | **06:43** |
| **04-07-2025** | **09:02** | **06:31** |
| **07-07-2025** | **09:00** | **06:30** |
| **08-07-2025** | **09:20** | **06:21** |
| **09-07-2025** | **09:10** | **04:00** |
| **10-07-2025** | **08:57** | **06:00** |
| **11-07-2025** | **08:52** | **06:01** |
| **12-07-2025** | **09:03** | **01:00** |
| **14-07-2025** | **09:10** | **06:12** |
| **15-07-2025** | **08:58** | **06:03** |
| **16-07-2025** | **08:56** | **06:31** |
| **17-07-2025** | **09:10** | **06:30** |
| **18-07-2025** | **08:58** | **06:21** |
| **21-07-2025** | **09:06** | **06:11** |
| **22-07-2025** | **09:30** | **06:20** |
| **23-07-2025** | **09:01** | **06:38** |
| **24-07-2025** | **09:10** | **06:30** |
| **25-07-2025** | **09:00** | **06:15** |
| **26-07-2025** | **08:52** | **06:18** |
| **28-07-2025** | **09:01** | **06:31** |
| **29-07-2025** | **08:58** | **06:13** |
| **30-07-2025** | **08:59** | **06:21** |
| **31-07-2025** | **08:55** | **06:05** |

THANK YOU