# PROJECT TITLE: Flask-Based Student Registration Web Application deployed using Jenkins

**Github repo link: https://github.com/AtharvaKishorMatale/stud-reg-flask-app/tree/master**

## Objective:

The main objective of this project was to develop a functional Flask-based student registration web application and to automate its deployment process using a robust Jenkins Continuous Integration/Continuous Deployment (CI/CD) pipeline. This project demonstrates the full lifecycle of a web application, from development and version control to automated build, testing, and deployment.

### 1. Technology Stack:

This project uses a combination of technologies to build and deploy the web application:

- **Frontend**: HTML for structuring the web pages.
- **Backend**: Python (Flask) for handling server-side logic, routing, and database interactions.
- **Database**: MySQL for persistent storage of student data.
- **Version Control**: Git and GitHub for managing the source code.
- **CI/CD**: Jenkins for automating the entire deployment workflow.
- **Port Forwarding**: socat for exposing the application running inside the Jenkins agent (e.g., WSL2) to the host machine (e.g., Windows).

### 2. Initial Setup:

Before deploying the application, the necessary tools and environment must be configured.

## *Step 2.1: Install Python and Dependencies:*

1. **Install Python 3**: Ensure that Python 3 is installed on your system. This is the core language for the Flask application.
2. **Create requirements.txt**: Create a file named requirements.txt to list the project's dependencies: Flask==2.3.2 and mysql-connector-python.

### *Step 2.2: Set up MySQL Database*

1. **Install MySQL Server**: Ensure that a MySQL server is installed and running on your system.
2. **Create Database**: Connect to the MySQL server (e.g., using the command line or a GUI tool) and create a database named studentdb.
3. **Create Table**: Within the studentdb database, create a table named students with the following fields: id (primary key, auto-increment), name, email, phone, course, address, and contact. This table structure is essential for the Flask application to store and retrieve student data correctly.

## *Step 2.3: Install Jenkins:*

1. **Set up Jenkins**: Install a Jenkins server on your local machine or a virtual machine. You can download the Jenkins WAR file or use a package manager.
2. **Install Plugins**: From the Jenkins UI, go to **Manage Jenkins** -> **Manage Plugins** and install the **Git Plugin** to enable Jenkins to interact with your GitHub repository.

## *Step 2.4: Install socat:*

Install the socat utility on your Jenkins agent machine. This tool is essential for forwarding network traffic from the Jenkins agent to your host machine. On a Debian-based system, you can use the command: sudo apt-get install socat.

## 3. Application File Structure

The project is organized into a standard Flask application structure:

- **app.py**: This is the main Flask application file.
- **requirements.txt**: This file lists the Python dependencies.
- **templates**: This directory contains the HTML files.
  **register.html**: The HTML for the student registration form.
  **students.html**: The HTML for displaying a list of registered students.
- **Jenkinsfile**: This file contains the Jenkins pipeline code.

## 4. Detailed Steps and Explanations

***Step 4.1: Application Code Development:***

The core of the project is the Flask application, defined in app.py.

- **Database Configuration**: The application connects to a **MySQL database** using mysql.connector with predefined credentials.
- **Routing**: The @app.route('/') decorator handles the registration form, while @app.route('/students') retrieves and displays all student data from the database.
- **Application Startup**: The app is configured to run on host='0.0.0.0' and port=5000 to be accessible from other systems on the network.

***Step 4.2: Templates:***

The templates directory contains HTML files processed by Flask's **Jinja2 templating engine**.

- **register.html**: A static HTML form used to collect student information.
- **students.html**: This page uses a **Jinja2 for loop** to dynamically display student data in a table, showing the power of templating.

## 5. Jenkins Pipeline Stages: Step-by-Step Breakdown:

The Jenkinsfile defines the entire CI/CD workflow, broken down into the following stages:

### *5.1: Clone GitHub Repo:*

This stage fetches the latest code from your GitHub repository, ensuring that Jenkins always builds and deploys the most recent version of the application.

### 5.2: Create Virtual Environment:

A dedicated Python virtual environment is created. This isolates the project's dependencies from the system's global Python packages, preventing conflicts.

### *5.3: Install Dependencies:*

The required packages (Flask, mysql-connector-python) are installed from requirements.txt into the isolated virtual environment, ensuring all necessary libraries are available.

### *5.4: Run Flask App:*

The Flask application is started in the background on port **5000**. The pipeline then verifies that the application is running and listening on the correct port, ensuring a successful launch.

### *5.5: Expose Port to Windows (via socat):*

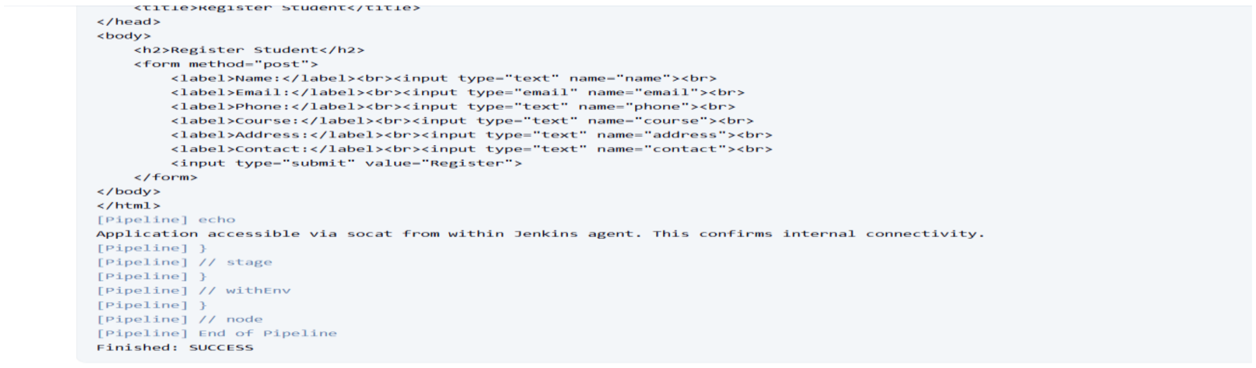
socat is used to forward the application's port from **5000** on the Jenkins agent to **5050** on the host machine. This makes the application accessible via a browser on the host.

### *5.6: Verify Application Access (from within Jenkins Agent):*

A curl command is executed to perform a final health check. A successful response confirms that the entire deployment chain, from the Flask app to the socat port forwarding, is working correctly.

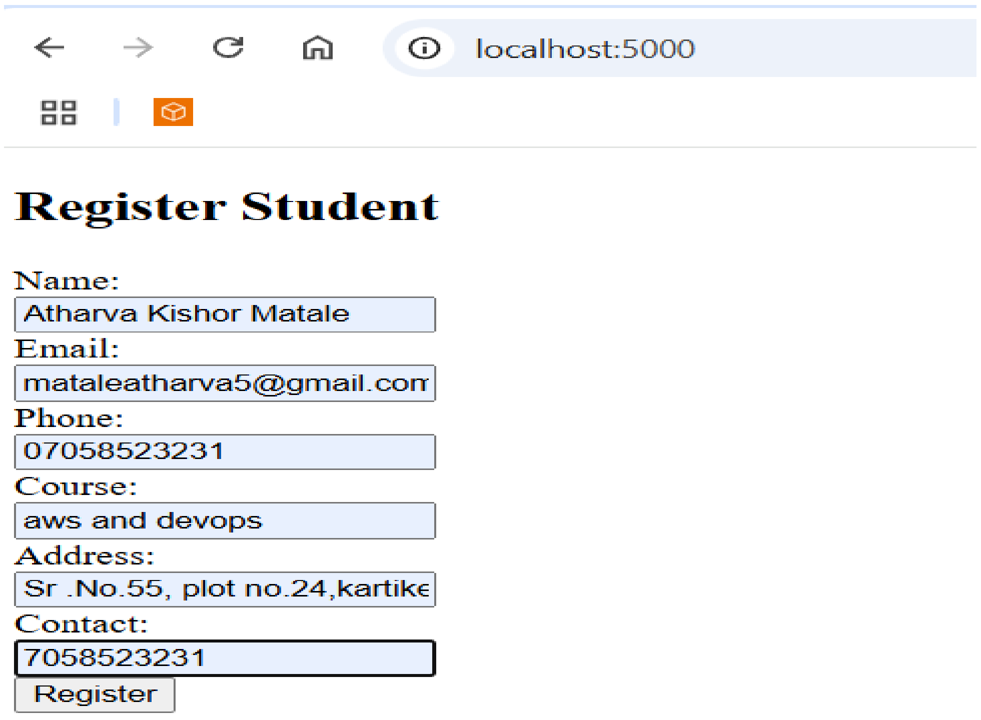
## 6)Final testing:

### 1)pipeline successfully executed:



```
    <title>Register Student</title>
</head>
<body>
    <h2>Register Student</h2>
    <form method="post">
        <label>Name:</label><br><input type="text" name="name"><br>
        <label>Email:</label><br><input type="email" name="email"><br>
        <label>Phone:</label><br><input type="text" name="phone"><br>
        <label>Course:</label><br><input type="text" name="course"><br>
        <label>Address:</label><br><input type="text" name="address"><br>
        <label>Contact:</label><br><input type="text" name="contact"><br>
        <input type="submit" value="Register">
    </form>
</body>
</html>
[Pipeline] echo
Application accessible via socat from within Jenkins agent. This confirms internal connectivity.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

**2)on port 5000 the page is visible:**

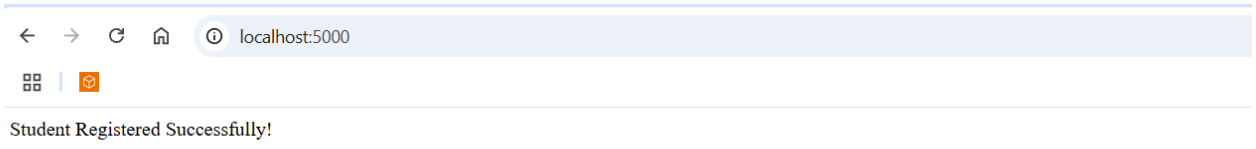


localhost:5000

# Register Student

Name:
Atharva Kishor Matale
Email:
mataleatharva5@gmail.com
Phone:
07058523231
Course:
aws and devops
Address:
Sr .No.55, plot no.24,kartike
Contact:
7058523231

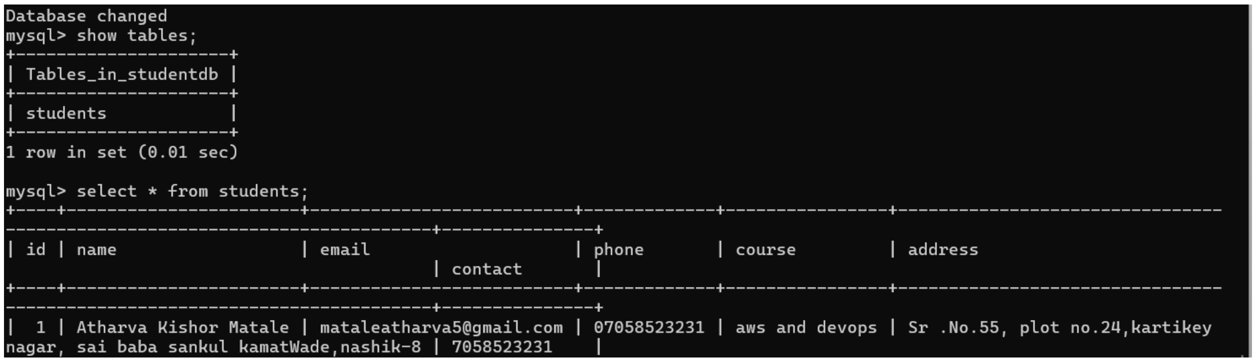
**2)student data successfullly registered in the mysql database:**



localhost:5000

Student Registered Successfully!

**3)we can see the data in mysql:**



```
Database changed
mysql> show tables;
+---------------------+
| Tables_in_studentdb |
+---------------------+
| students            |
+---------------------+
1 row in set (0.01 sec)

mysql> select * from students;
+----+-----------------------+--------------------------+-------------+----------------+--------------------------------
------------------------------------------+----------------+
| id | name                  | email                    | phone       | course         | address
                                          | contact        |
+----+-----------------------+--------------------------+-------------+----------------+--------------------------------
------------------------------------------+----------------+
|  1 | Atharva Kishor Matale | mataleatharva5@gmail.com | 07058523231 | aws and devops | Sr .No.55, plot no.24,kartikey
nagar, sai baba sankul kamatWade,nashik-8 | 7058523231     |
```

## 6. Summary:

This project successfully developed a Flask-based student registration application with a MySQL database backend. The entire process, from code check-out to live deployment, was automated using a Jenkins pipeline. By leveraging a virtual environment, the pipeline ensures a clean and reproducible build. The use of socat successfully addresses the challenge of exposing the application from a virtualized Jenkins environment to the host machine, resulting in a fully functional and verifiable deployment. The pipeline's final verification stage provides confidence that the application is not only running but is also accessible and ready for use.