

Project 3  
Flask-Based Student Registration Web Application  
deployed using Jenkins

Batch

MCA 7 Feb 2025 &  
DevOps 2 Jun 2025

Name

Suraj Molke

Name of the Mentor

Ravindra Bagle Sir  
Swati Zampal Ma'am



Fortune Cloud Technologies

2nd Floor, Shirodkar House, Opposite To Amit Cafe, Congress House Road, Near Pune Municipal Corporation, Shivajinagar, Pune - 411005.

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

## Objective:

Develop and deploy a Flask-based web application that allows users to register students by submitting their details via a web form. The application should store the submitted data in a backend and provide the ability to retrieve or display this information when needed.

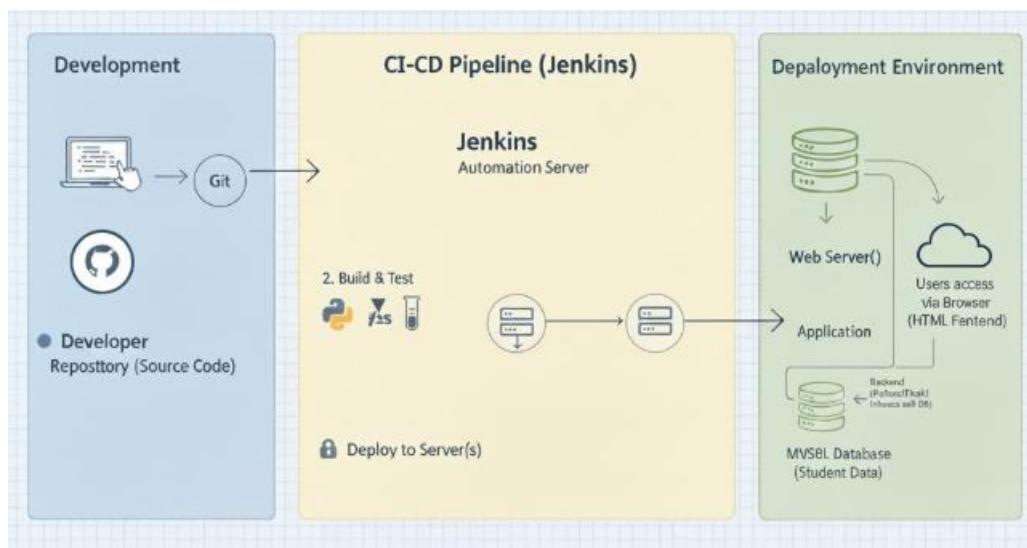
## Project Overview:

This project involves designing a simple web application that mimics a student registration system. The goal is to gain hands-on experience in web development, form handling, backend logic, and data storage using Python Flask and standard frontend technologies.

## Technology Stack:

- ✓ Frontend: HTML, CSS
- ✓ Backend: Python (Flask)
- ✓ Database: MySQL
- ✓ Version Control: Git & GitHub
- ✓ Deployment (Optional): EC2

## Architectural diagram



## **Step .1: Install Python and Dependencies**

1. Install Python 3: Ensure that Python 3 is installed on your system.

- Flask==2.3.2
- mysql-connector-python

```
Flask==2.3.2
mysql-connector-python
```

## **Step 2: Set up MySQL Database**

1. Install and start MySQL Server.
2. Create a database named studentdb.
3. Create students table with fields:
  - id (primary key, auto-increment)
  - name, email, phone, course, address, contact

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    course VARCHAR(100),
    address VARCHAR(200),
    contact VARCHAR(100)
);
```

## **Step 3: Install Jenkins**

1. Install Jenkins server.
2. Install Git Plugin via Manage Jenkins → Manage Plugins.
  - Manage Jenkins → Manage Plugins → Available → Git Plugin
  - Git
  - Pipeline
  - SSH Agent
  - Ansible (optional)
  - Python plugin

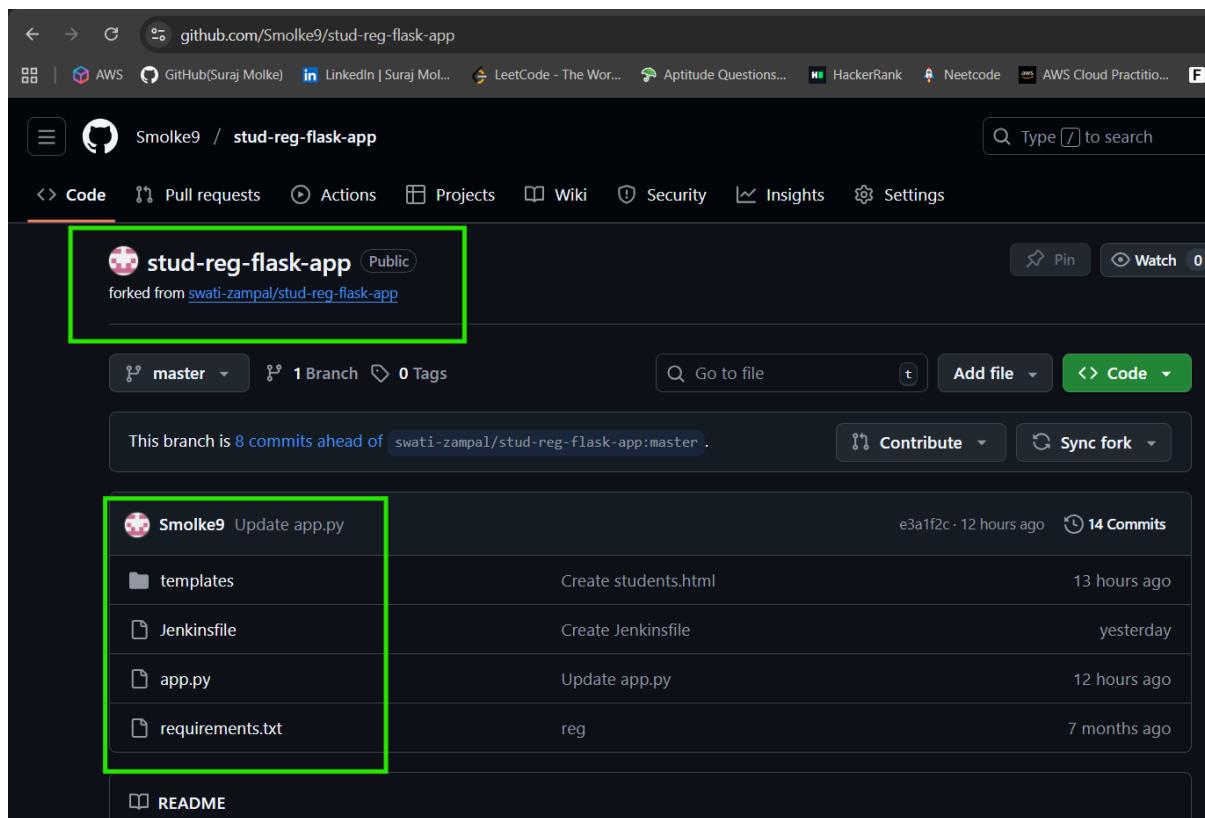
## **Step 4: Install socat**

Install socat on Jenkins agent using:

```
sudo apt-get install socat
```

## Step 5: Application File Structure From Forked Via <https://github.com/swati-zampal/stud-reg-flask-app.git> To My Repo <https://github.com/Smolke9/stud-reg-flask-app.git>

- app.py – Main Flask application logic
- requirements.txt – Python dependencies
- templates/
  - register.html
  - students.html
- Jenkinsfile – CI/CD pipeline stages



## Step 6: Application Code Development

- Uses mysql.connector for database connectivity.
- '/' route displays registration form.
- '/students' displays stored student data.
- Runs on host=0.0.0.0 port=5000.

## Step 7: Templates

- register.html – HTML form for user input.
- students.html

## Step 8: Jenkins Pipeline Stages

### 8.1 Clone GitHub Repo – pulls latest code.

The screenshot shows the Jenkins 'Configure' screen for a job named 'flask1'. The 'General' tab is selected. Under 'GitHub project', the 'Project url' field contains 'https://github.com/Smolke9/stud-reg-flask-app.git'. A green box highlights this section. Below it, under 'Advanced', there are several optional checkboxes: 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterised', 'Throttle Concurrent Builds', and 'Throttle builds'. At the bottom are 'Save' and 'Apply' buttons.

### 8.2 Create Virtual Environment – isolates dependencies.

The screenshot shows the Jenkins 'Configure' screen for the 'flask1' job, with the 'Pipeline' tab selected. The 'Definition' section is set to 'Pipeline script'. The script content is as follows:

```

agent any

stages {
    stage('Clone GitHub Repo') {
        steps {
            echo "Cloning repository..."
            git branch: 'master', url: 'https://github.com/Smolke9/stud-reg-flask-app.git'
        }
    }

    stage('Create Virtual Environment') {
        steps {
            sh '''
                python3 -m venv venv
            '''
        }
    }
}

```

A green box highlights the entire script area. At the bottom of the script editor, the 'Use Groovy Sandbox' checkbox is checked. Below the editor are 'Save' and 'Apply' buttons.

### 8.3 Install Dependencies – Flask + MySQL connector.

### 8.4 Run Flask App – starts on port 5000.

## 8.5 Expose Port using socat – forwards 5000 → 5050.

The screenshot shows the Jenkins Pipeline interface for a job named 'flask1'. On the left, there's a sidebar with various Jenkins-related links like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, GitHub, Favorite, Open Blue Ocean, Stages, Rename, Pipeline Syntax, and Credentials. The main area is titled 'Stage View' and displays a timeline of six stages: 'Clone GitHub Repo' (978ms), 'Create Virtual Environment' (4s), 'Install Dependencies' (4s), 'Run Flask App' (10s), 'Verify Port 5000' (337ms), and 'Test App Reachability' (368ms). Below the timeline, it says 'Average stage times: (full run time: ~21s)'. A note indicates 'Nov 19 No Changes'. At the bottom, there's a 'Builds' section showing the last four builds, all of which completed successfully. The desktop taskbar at the bottom shows weather (7°C, Sunny), system icons, and the date/time (19-11-2025).

## 8.6 Verify Access – curl health check ensures app is running.

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

The screenshot shows the Jenkins Pipeline console output for build #1. It displays the HTML code of a 'Student Registration Form' page. The output ends with the message '[Pipeline] End of Pipeline' and '[Pipeline] Finished: SUCCESS'. The Jenkins version 'Jenkins 2.536' is visible at the bottom right. The desktop taskbar at the bottom shows weather (7°C, Sunny), system icons, and the date/time (19-11-2025).

**Step 9:** On port 5000 page available

The screenshot shows a web browser window with the following details:

- Address bar: Not secure 13.232.184.169:5000
- Links in the top right: AWS, GitHub(Suraj Molke), LinkedIn | Suraj Mol
- Page title: Register Student
- Form fields:
  - Name: suraj
  - Email: smolke9@gmail.com
  - Phone: 9595333985
  - Course: aws devops
  - Address: ~~pimpri pune~~
  - Contact: 9595333985
- Buttons: Register

**Finally Student data successfully registered in mysql database.**

## Conclusion

- **Separation of Concerns:** The frontend (HTML), backend (Flask), and database (MySQL) are distinct layers, making the application modular and easier to maintain.
- **Version Control Best Practices:** Using Git and GitHub ensures that all code changes are tracked, collaborated on effectively, and easily revertible. This is fundamental for team-based development and stable releases.
- **Automated CI/CD (DevOps):** Jenkins is at the heart of this project, automating the entire workflow from code commit to deployment. This significantly:
  - **Reduces Manual Errors:** Eliminates human mistakes in the build, test, and deployment phases.
  - **Increases Speed:** Speeds up the delivery of new features and bug fixes to users.
  - **Ensures Quality:** Automated tests (unit and integration) run with every change, catching issues early in the development cycle.
  - **Promotes Consistency:** Guarantees that deployments are consistent across environments.
- **Scalability & Maintainability:** A well-structured Flask application interacting with a robust MySQL database provides a solid foundation that can be scaled horizontally if needed and is relatively easy to debug and enhance.

In conclusion, this project provides a comprehensive and automated pipeline for building and deploying a Flask-based web application. It not only delivers a functional student registration system but also establishes a professional and efficient development and deployment process, making it a strong foundation for future enhancements and continuous delivery.