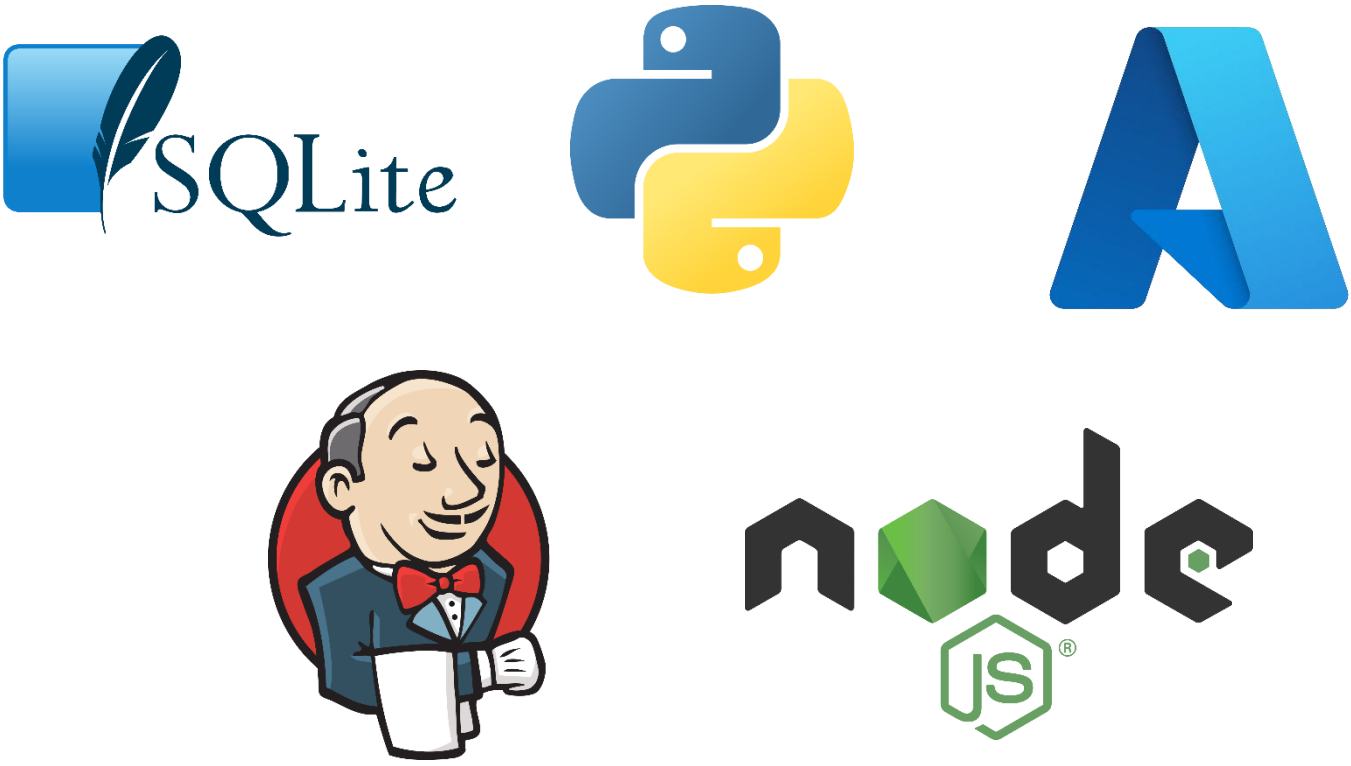
🚀 Capstone Project: DevOps-Driven Flask Application with SQLite and Azure Deployment



Project Title

"Automating Deployment of a Flask REST API with SQLite using Jenkins and Azure"

📌 Project Overview

This project focuses on developing and deploying a **Flask-based REST API** with a **SQLite database**, while integrating **DevOps automation** using **Jenkins** and **deploying the application on Azure**. The goal is to showcase expertise in **Python fundamentals, API development, CI/CD pipelines, and cloud deployment**.

📌 Objectives

1. **Develop a Flask API**
    - Design a RESTful API that allows users to perform CRUD operations.
    - Implement authentication and error handling mechanisms.
2. **Integrate an SQLite Database**
    - Create and manage a database using SQLite.
    - Store and retrieve structured data efficiently.
3. **Automate DevOps Pipeline Using Jenkins**
    - Set up continuous integration and deployment (CI/CD) pipelines.
    - Automate building, testing, and deploying the Flask application.
4. **Deploy the Application on Azure**
    - Host the application on **Azure App Service**.
    - Ensure high availability and scalability.

📌 Scope of the Project

- **Backend Development**: Creating a Flask-based API with well-defined endpoints.
- **Database Management**: Using SQLite for data persistence and efficient queries.
- **DevOps Automation**: Implementing Jenkins pipelines for automated testing and deployment.
- **Cloud Deployment**: Hosting the application on **Azure** for scalability.

📌 Technology Stack

Category	Technology Used
Programming Language	Python 3.x
Web Framework	Flask
Database	SQLite
Version Control	Git & GitHub
CI/CD Tool	Jenkins
Containerization	Docker
Cloud Platform	Microsoft Azure
Infrastructure as Code (IaC)	Azure CLI

📌 Functional Requirements

1. User Management API

- **POST /users** → Add a new user.
- **GET /users** → Retrieve a list of all users.
- **GET /users/{id}** → Retrieve details of a specific user.
- **PUT /users/{id}** → Update user details.
- **DELETE /users/{id}** → Delete a user from the database.

2. Database Management

- Store user data persistently using **SQLite**.
- Ensure database integrity and constraints.

3. CI/CD Pipeline Integration

- **Continuous Integration**:
    - Automate build and test execution via **Jenkins pipelines**.
    - Run unit tests before deployment.
- **Continuous Deployment**:
    - Automate deployment to **Azure App Service**.
    - Ensure rollbacks in case of failures.

4. Deployment on Azure

- Deploy the Flask API as a **Docker container**.
- Use **Azure App Service** for hosting.
- Monitor application performance and logs on Azure.

📌 Non-Functional Requirements

- **Scalability**: Deploy on Azure to handle increasing requests.
- **Security**: Implement authentication and API key-based access.
- **Reliability**: Ensure high availability with automated deployment.
- **Maintainability**: Keep modular code and use logging for debugging.

📌 Project Architecture

The project follows a **Microservices-Based Architecture**, where the Flask API is decoupled from the database and deployed independently using DevOps pipelines.

```
Client (Frontend or API Consumer)
        ↓
Flask REST API (Business Logic)
        ↓
SQLite Database (Data Storage)
        ↓
Jenkins Pipeline (CI/CD Automation)
        ↓
Azure App Service (Cloud Hosting)
```

📌 Flow of Operations

1. A user sends an HTTP request to the Flask API.
2. The API processes the request and interacts with the SQLite database.
3. Data is retrieved, updated, or deleted based on the request.
4. Jenkins automates the testing and deployment process.

5. The application is deployed on **Azure App Service** using a containerized approach.

📌 Expected Outcomes

- A **fully functional REST API** developed using **Flask**.
- Automated **CI/CD pipeline** using **Jenkins**.
- Deployment on **Azure**, making the application **scalable and accessible**.
- A **comprehensive project demonstrating DevOps best practices**.

Conclusion

This capstone project provides hands-on experience in **Python API development, database management, CI/CD automation, and cloud deployment**. It prepares students for **real-world DevOps and cloud-based software development** scenarios.