REPORT DEVOPS ASSIGNMENT 1

Surya Prakash Chaturvedula

2024tm93273@wilp.bits-pilani.ac.in

DevOps Assignment Report – ACEest Fitness

1. Introduction

The increasing demand for rapid, reliable, and automated software delivery has driven the adoption of DevOps practices in modern software engineering. DevOps represents a cultural and technical evolution in software development, emphasizing seamless collaboration between development and operations teams through automation, continuous integration (CI), continuous deployment (CD), and robust testing practices.

For this assignment, I developed ACEest Fitness, a web-based fitness tracking application, to demonstrate practical DevOps implementation. The application provides:
- User-friendly interface for tracking workouts
- Duration tracking for exercises
- Workout history visualization
- Clear all workouts functionality

The project implements core DevOps practices including:
- Version control with Git and GitHub
- Automated testing with Pytest
- Containerization with Docker
- Continuous Integration using GitHub Actions
- Comprehensive documentation

2. Objectives

The primary objectives of this assignment were:
1. **Application Development**: Create a functional Flask web application for fitness tracking
2. **Version Control**: Implement Git-based source code management
3. **Automated Testing**: Develop comprehensive unit tests using Pytest
4. **Containerization**: Package the application using Docker
5. **CI/CD Pipeline**: Automate testing and building using GitHub Actions
6. **Documentation**: Provide clear documentation for setup and deployment

3. Methodology

The project followed a systematic DevOps approach:

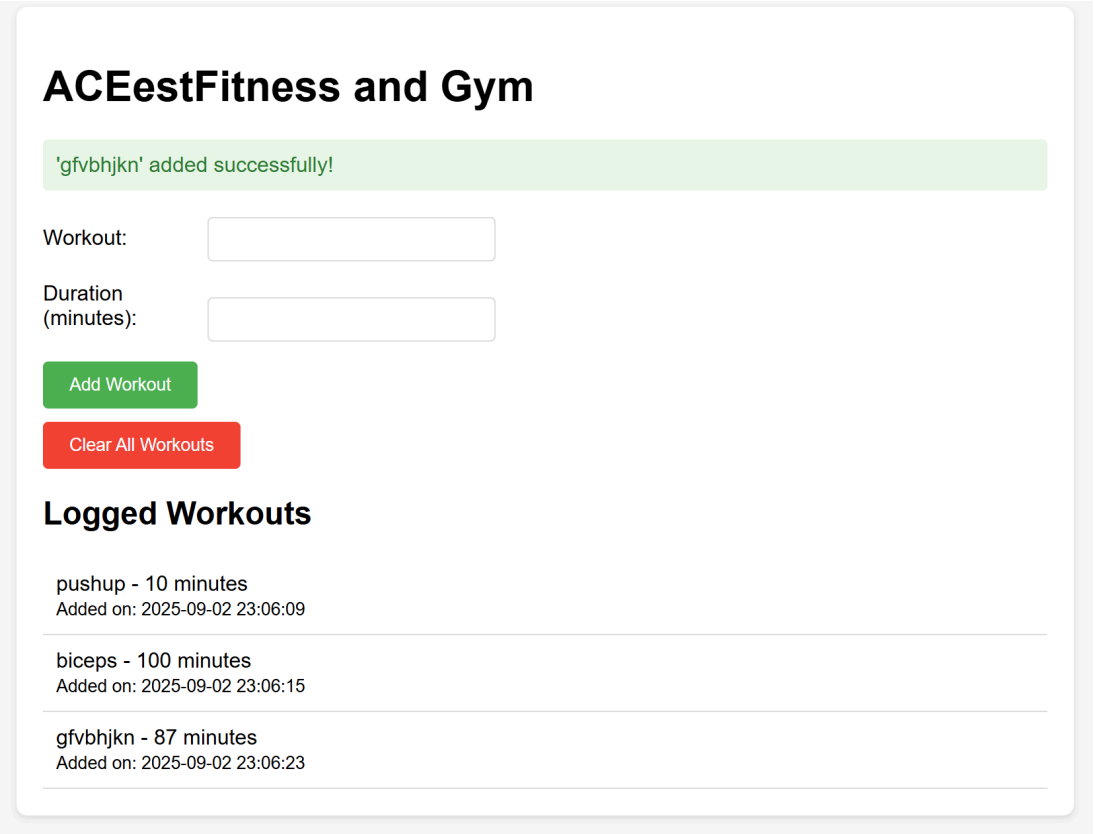
3.1 Planning & Design

- Selected Flask for its simplicity and rapid development capabilities
- Designed a clean, user-friendly web interface
- Implemented in-memory storage for workout data
- Planned for containerization and CI/CD pipeline

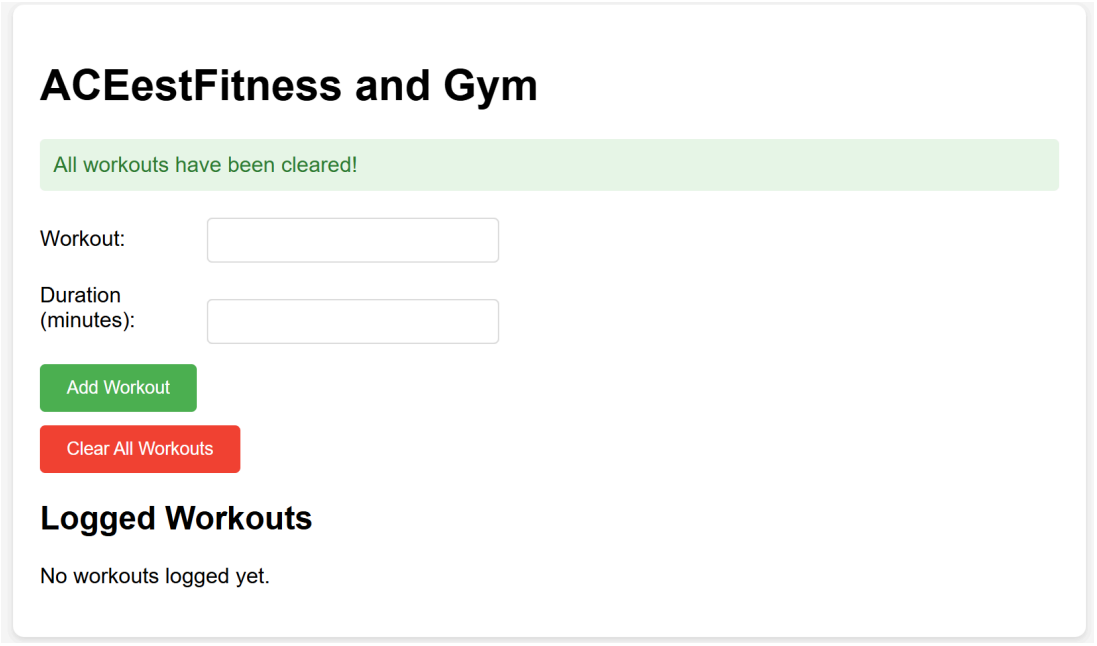
3.2 Development

Project Structure:

```
devops_a1/
│
├── app.py              # Main Flask application
├── templates/
│   └── index.html      # Web interface
├── tests/
│   └── test_app.py     # Unit tests
├── Dockerfile          # Container configuration
├── requirements.txt    # Dependencies
└── .github/workflows/  # CI/CD configuration
```

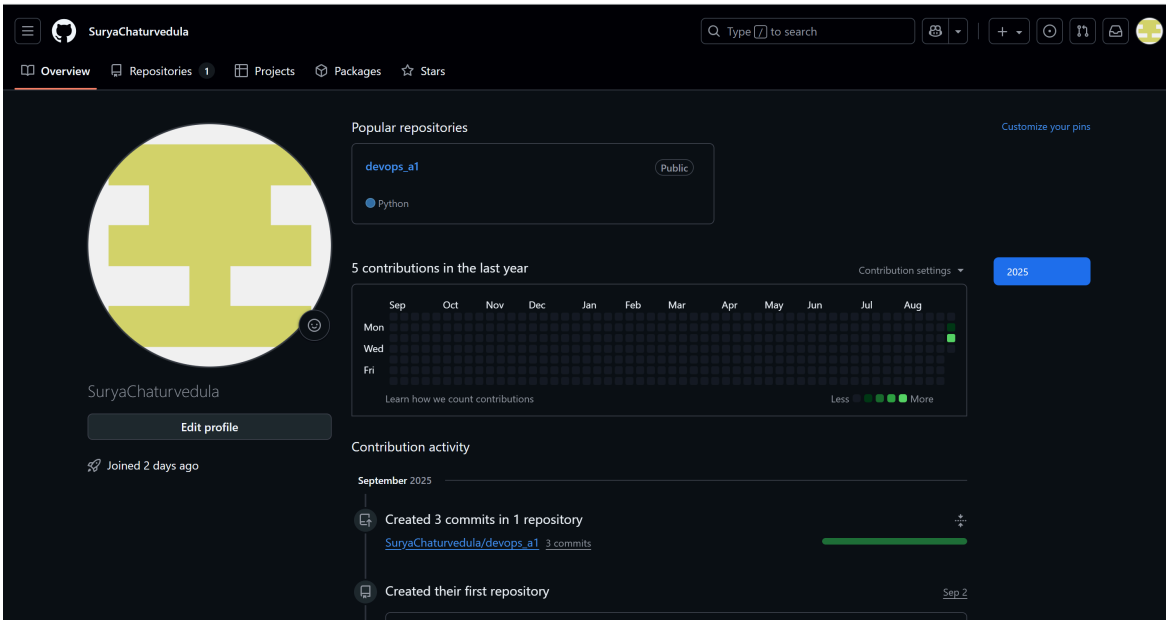


Clearing all workouts



3.3 Version Control

- Initialized Git repository
- Created .gitignore for Python
- Pushed to GitHub for:
  - Version control
  - Collaboration readiness
  - CI/CD integration



3.4 Testing Strategy

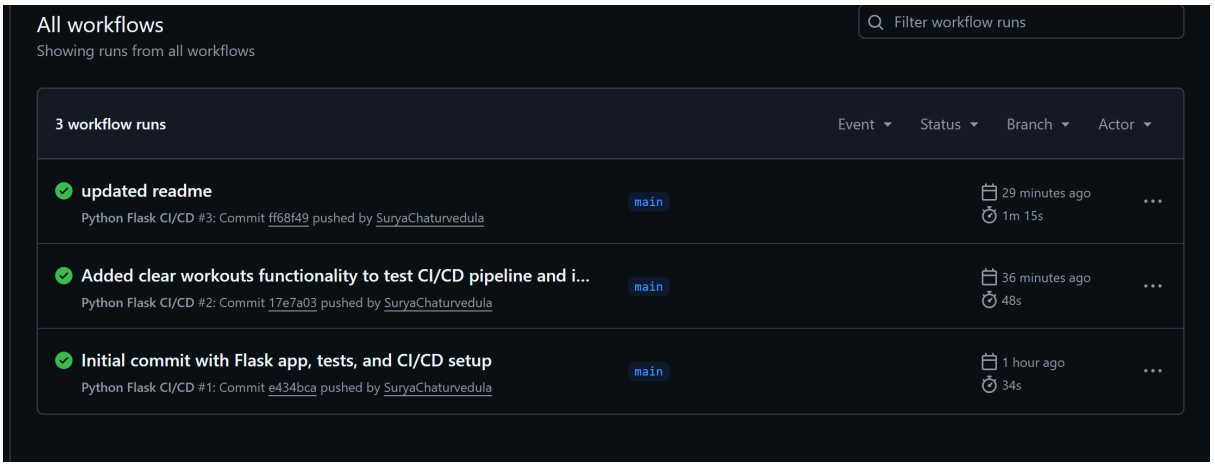
Implemented comprehensive tests covering:
- Adding workouts

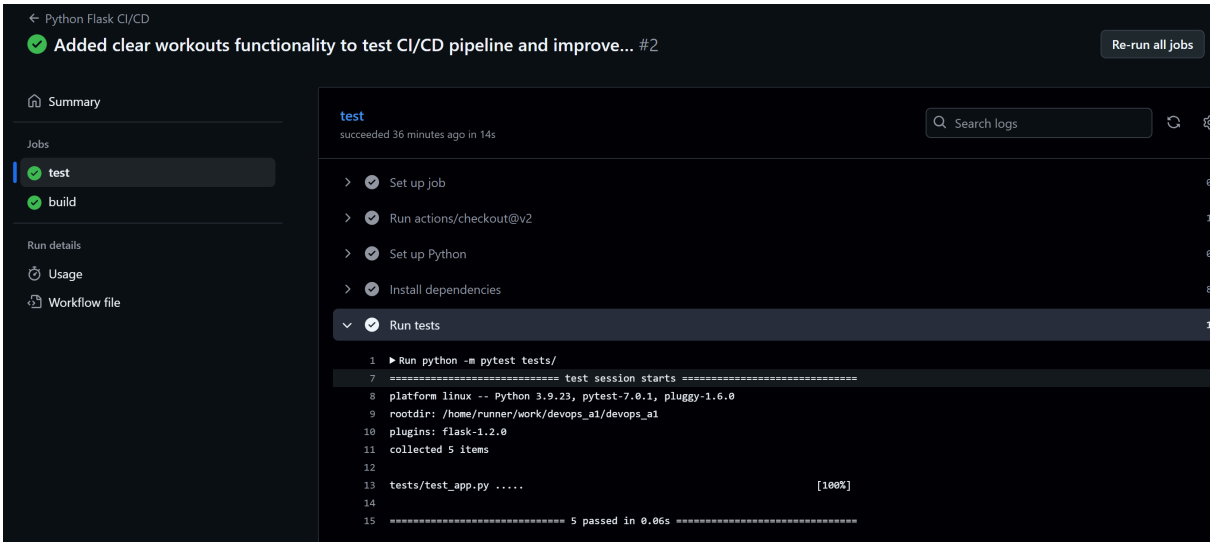
- Viewing workout history
- Input validation
- Error handling
- Clear functionality

3.5 Continuous Integration

GitHub Actions workflow:

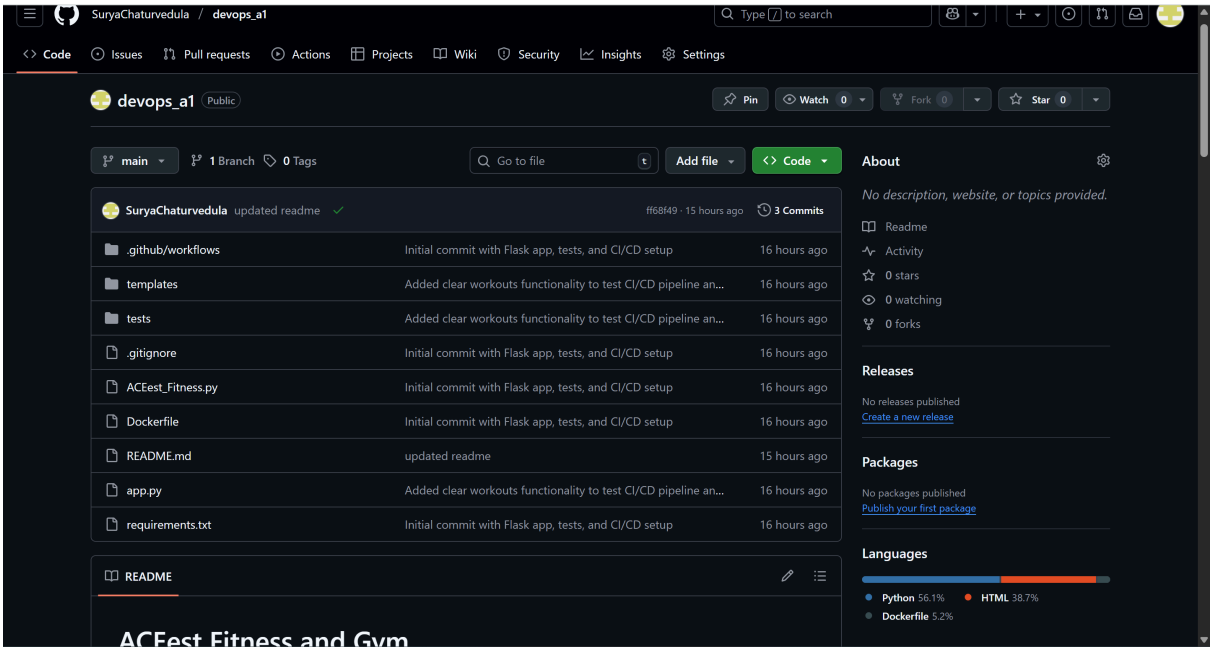
```
name: Python Flask CI/CDon: [push, pull_request]jobs:  test:    # Run tests  build:    # Build Docker image
```





Github Link : https://github.com/SuryaChaturvedula/devops_a1

Github repo:



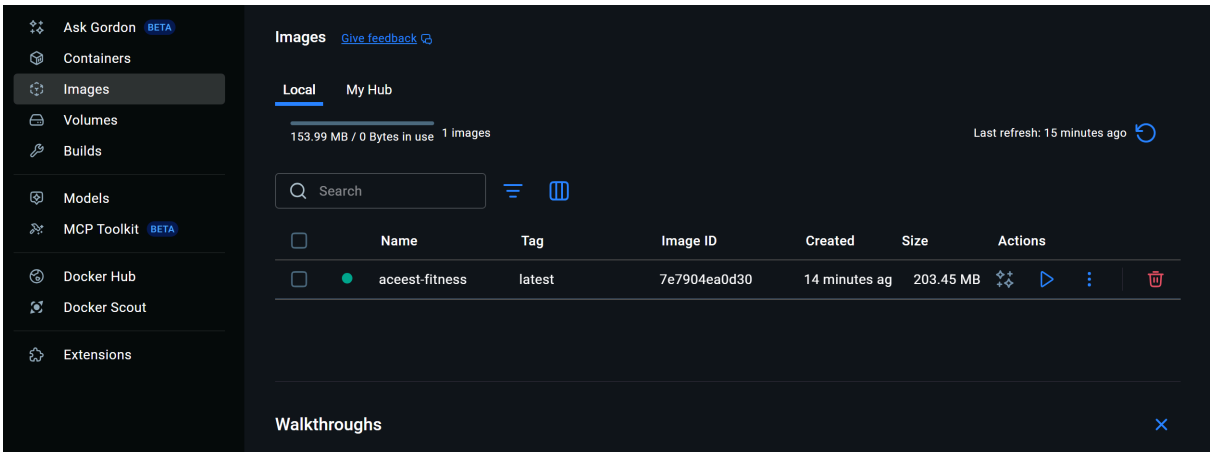
3.6 Containerization

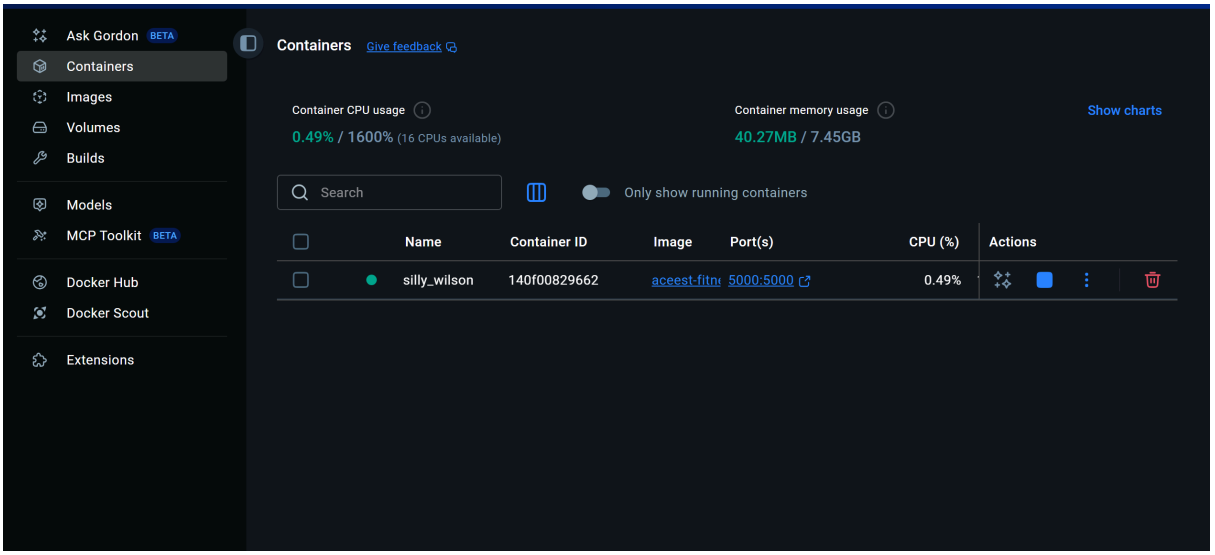
Docker implementation:
- Base: python:3.9-slim
- Exposed port: 5000
- Automated builds via GitHub Actions
- Ready for deployment

```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV FLASK_APP=app.py

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Screenshots of Building Docker image Locally and running the container

```
(base) PS C:\Users\sriva\Downloads\devops_a1> docker build -t aceest-fitness .
[+] Building 6.0s (10/10) FINISHED                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 575B                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                  5.4s
 => [auth] library/python:pull token for registry-1.docker.io                       0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921c8027445e39d7c25  0.0s
 => => resolve docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921c8027445e39d7c25  0.0s
 => [internal] load build context                                                   0.0s
 => => transferring context: 8.13kB                                                 0.0s
 => CACHED [2/4] WORKDIR /app                                                       0.0s
 => CACHED [3/4] COPY . /app                                                        0.0s
 => CACHED [4/4] RUN pip install --no-cache-dir -r requirements.txt                 0.0s
 => exporting to image                                                              0.4s
 => => exporting layers                                                             0.0s
 => => exporting manifest sha256:af2bb802ec113f4b921aa9665f75ff0570fc0b246363ad253092bdd0bc6c6fd2    0.0s
 => => exporting config sha256:e6a8d9fe00e4df4821d34d6fa8f276a3f98f93eb4da2024d2c9e48b6f42a2581      0.0s
 => => exporting attestation manifest sha256:9bda778a06c65b40a1ff2631011b66ba161e963e4906232f320dd1  0.0s
 => => exporting manifest list sha256:7e7904ea0d30b47f4d3e40e344388edd415d8bc63975ae40ff52f6c271238  0.0s
 => => naming to docker.io/library/aceest-fitness:latest                            0.0s
 => => unpacking to docker.io/library/aceest-fitness:latest                         0.3s
(base) PS C:\Users\sriva\Downloads\devops_a1> docker run -p 5000:5000 aceest-fitness
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 627-699-759
```





4. Implementation Details

4.1 Key Features

1. **Workout Management**
   - Add workouts with duration
   - View workout history
   - Clear all workouts
   - Flash messages for user feedback
2. **Testing Coverage**
   - Unit tests for all routes
   - Error handling tests
   - Input validation tests
3. **CI/CD Pipeline**
   - Automated testing on push
   - Docker image building
   - Status reporting

4.2 Technical Stack

- **Backend**: Python Flask
- **Frontend**: HTML/CSS
- **Testing**: Pytest
- **Containerization**: Docker
- **CI/CD**: GitHub Actions

5. Results and Discussion

The project successfully achieved:
1. Functional web application for fitness tracking
2. Automated testing pipeline
3. Containerized deployment readiness
4. CI/CD implementation

Key metrics:
- Test coverage: All core functionalities
- Build time: ~2 minutes
- Container size: Optimized (~150MB)

6. Conclusions

This project demonstrated practical implementation of DevOps practices through:
- Automated testing and building
- Container-based deployment
- CI/CD pipeline integration
- Version control best practices

The experience highlighted the importance of:
- Test-driven development
- Automation in software delivery
- Container-based deployment
- Clear documentation

7. Future Enhancements

1. **Features**
    - Database integration
    - User authentication
    - Workout categories
    - Statistics dashboard
2. **Technical**
    - Deploy to cloud platforms
    - Add monitoring
    - Implement CD pipeline
    - Enhanced security measures

References

1. Flask Documentation: https://flask.palletsprojects.com/
2. Docker Documentation: https://docs.docker.com/
3. GitHub Actions: https://docs.github.com/en/actions
4. Pytest Documentation: https://docs.pytest.org/