FOSSEE Winter Internship Report

On

Automating Deployments with GitHub Actions CI/CD for Osdag on Cloud

Submitted by

Eeshu

3rd Year B.Tech Student, Department of Computer Science

Ajay Kumar Garg Engineering College

Ghaziabad

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

February 3, 2025

Acknowledgments

I would like to express my heartfelt gratitude to all those who supported me throughout my journey of implementing the CI/CD pipeline for the Osdag project. This endeavor would not have been possible without the guidance, encouragement, and assistance of numerous individuals and organizations.

First and foremost, I would like to extend my deepest thanks to the dedicated members of the Osdag team—Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia. Their continuous support, technical expertise, and insightful feedback were invaluable in ensuring the successful completion of this project.

I am especially grateful to Prof. Siddhartha Ghosh, Principal Investigator (PI) of Osdag and a faculty member in the Department of Civil Engineering at IIT Bombay. His expert guidance, thoughtful suggestions, and unwavering encouragement played a critical role in shaping the project's direction and outcome.

I also owe my sincere thanks to Prof. Kannan M. Moudgalya, Principal Investigator of the FOSSEE project in the Department of Chemical Engineering at IIT Bombay. His support and leadership within the FOSSEE initiative provided the platform and motivation for this work, and I am deeply appreciative of his vision.

My sincere appreciation goes to the FOSSEE Managers, Usha Viswanathan and Vineeta Parmar, along with their team, for their administrative support, coordination, and encouragement throughout the project. Their efforts ensured smooth progress at every stage.

I would like to acknowledge the crucial support provided by the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education, Government of India. Their commitment to advancing educational technology enabled this project to flourish, and I am deeply grateful for their involvement.

I also wish to thank my colleagues who worked alongside me on this project. Their

collaboration, valuable insights, and camaraderie made this experience not only successful but also incredibly enriching.

Finally, I would like to express my sincere gratitude to my college, department, head, and principal for their unwavering support throughout my academic journey. Their encouragement laid the strong foundation that helped me achieve this milestone.

Thank you all for your invaluable contributions and support.

Contents

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [1, 2, 3]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

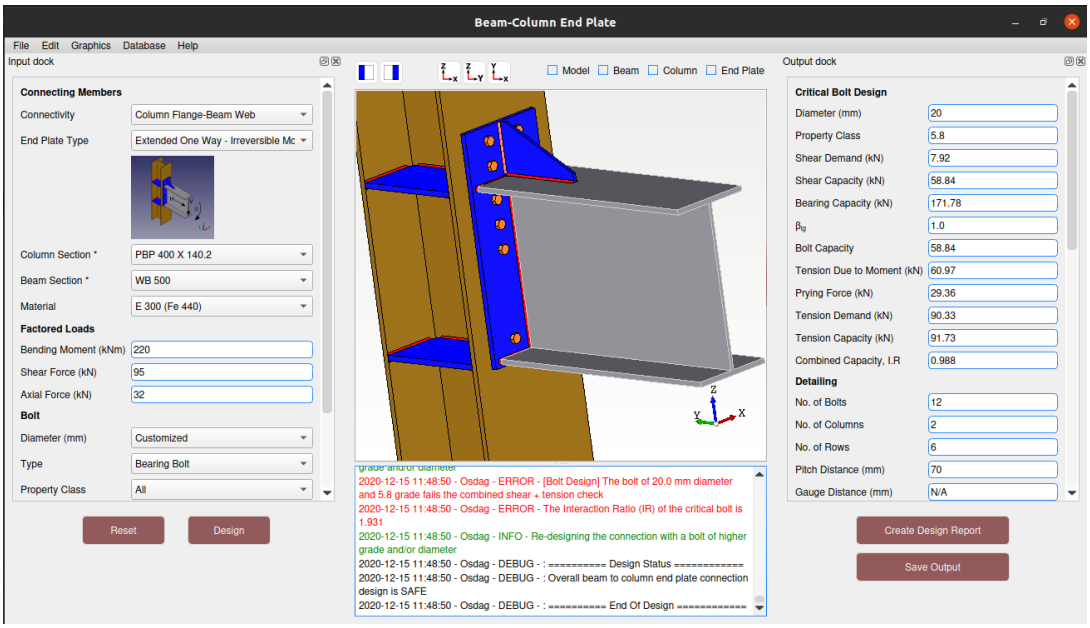


Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

The problem defined during the screening task was to develop a user-friendly interface for the Cleat Angle connection module within the Osdag software. The objective was to enhance the user experience by integrating efficient functionalities, improving the graphical interface, and ensuring seamless data handling for structural connection designs.

2.2 Tasks Done

2.2.1 Development of Cleat Angle Component

The Cleat Angle component was developed using React, incorporating various functionalities to support design input, output visualization, and report generation.

2.2.2 Key Functionalities Implemented

- **User Interface Design:** Implemented a clean and intuitive UI using Ant Design components for dropdown menus, input fields, and modals.
- **3D Model Rendering:** Integrated Three.js through the @react-three/fiber library to render 3D models, enhancing visualization capabilities.
- **Design Preferences Module:** Developed a module to allow customization of design preferences, such as bolt diameter, plate thickness, and property class.

- **Session Management:** Utilized React context (ModuleContext and UserContext) for managing design data, user sessions, and application state.
- **Error Handling:** Implemented robust error handling mechanisms to ensure smooth operation and user feedback in case of invalid inputs or system errors.
- **Data Handling and API Integration:** Managed input data through controlled components, integrated with backend APIs for design calculations, and processed the output for display.
- **Design Report Generation:** Enabled users to create comprehensive design reports with customizable fields such as company name, designer details, and project information.
- **File Handling:** Implemented functionalities for saving, loading, and downloading design inputs and outputs, including CAD images and 3D model snapshots.
- **Responsive Design:** Ensured the application is responsive and performs optimally across different devices and screen sizes.

2.2.3 Technologies Used

- React.js
- Ant Design
- @react-three/fiber for 3D rendering
- Context API for state management
- Base-64 encoding for secure data handling

2.2.4 Challenges and Solutions

- **State Management Complexity:** Managed complex state transitions using React's context API and hooks.
- **Performance Optimization:** Optimized rendering performance for 3D models by conditionally rendering components and minimizing re-renders.

- **Error Handling:** Addressed common errors like missing data inputs, ensuring user-friendly error messages and guidance.

Chapter 3

Implementing CI/CD Pipeline for Osdag Project

3.1 Problem Statement

The primary objective of this task is to design and implement a Continuous Integration and Continuous Deployment (CI/CD) pipeline for the Osdag project using GitHub Actions and Docker. The CI/CD pipeline aims to automate the critical stages of software development, including building, testing, and deploying the application. This automation ensures that the codebase remains in a deployable state at all times, reducing manual intervention, minimizing errors, and accelerating the development lifecycle.

The implementation of this pipeline enhances code quality through continuous integration, where each code commit triggers automated tests, and continuous deployment, which ensures seamless and consistent deployment to the production environment. The use of Docker standardizes the development and deployment environments, reducing the chances of environment-specific issues.

3.2 Tasks Done

The CI/CD pipeline implementation involved the following key tasks:

- **Set up GitHub Actions Workflow:** Configured GitHub Actions to automate the build, test, and deployment processes. Created workflow YAML files defining jobs and steps for CI/CD stages.

- **Created Dockerfiles for Backend, Frontend, and PostgreSQL:** Developed Dockerfiles for the backend and frontend applications, along with a Dockerfile for PostgreSQL to ensure consistent environment setup across development and production.
- **Built Docker Images and Pushed Them to Docker Hub:** Built Docker images from the Dockerfiles and pushed them to Docker Hub for easy accessibility and deployment in different environments.
- **Updated `docker-compose.yml` to Use Docker Images:** Modified the `docker-compose.yml` file to integrate the newly created Docker images, facilitating seamless multi-container deployment.
- **Implemented CI/CD Pipeline Steps in GitHub Actions:** Defined and implemented CI/CD pipeline stages in GitHub Actions, including code checkout, environment setup, building Docker images, running automated tests, and deploying to production.

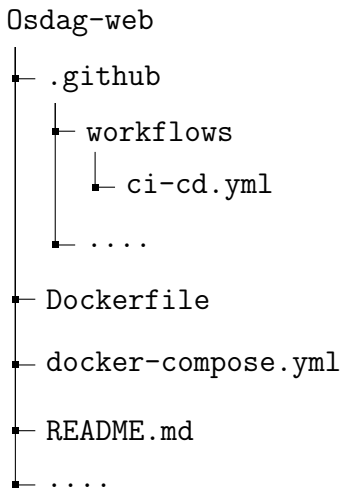
3.3 Full Code

The full codebase for the Osdag project, including backend application logic, models, views, Docker configurations, and CI/CD workflows, is available in the project repository. The repository contains well-structured modules, making it easier to understand the integration of CI/CD components.

3.4 Documentation

3.4.1 Directory Structure

The directory structure of the Osdag project reflects the organization of CI/CD-related files, Docker configurations, and application code:



This structure highlights:

- **.github/workflows/ci-cd.yml:** Defines the CI/CD workflows using GitHub Actions.
- **Osdag-web:** Contains the backend and frontend application code along with their respective Dockerfiles.
- **docker-compose.yml:** Manages multi-container Docker applications, integrating backend, frontend, and database services.
- **README.md:** Provides documentation and instructions for setting up and running the project.

Further details, including configuration files and scripts, are documented within the repository to assist developers in understanding and maintaining the CI/CD pipeline.

Chapter 4

Building and Pushing Docker Images

4.1 Problem Statement

The goal of this task is to build Docker images for the backend and frontend services of the Osdag project and push these images to Docker Hub. Docker images encapsulate the backend and frontend services into containers, which can then be deployed consistently across different environments. By pushing the images to Docker Hub, we ensure that anyone can pull and run the application without worrying about system dependencies or configuration inconsistencies.

4.2 Tasks Done

- Built Docker images using the Dockerfiles located in the respective backend and frontend directories.
- Pushed the created Docker images to Docker Hub to facilitate easy access and deployment.

4.3 Documentation

The following sections detail the commands used to build and push Docker images for both the backend and frontend services. Docker provides a powerful way to package applications into containers, making them portable and easy to deploy. By using Docker

commands, we are able to automate the process of building the containerized applications and storing them in a central repository for easy access.

4.3.1 Build Backend Image

In this section, we use the following command to build the Docker image for the backend service. The command builds an image based on the Dockerfile located in the '/home/eeshu/Osdag-web' directory.

```
docker build -t eeshuyadav/osdag-web-backend:v1.1 /home/eeshu/
   Osdag-web
```

Explanation: - 'docker build': This command tells Docker to start the process of building an image. - '-t eeshuyadav/osdag-web-backend:v1.1': The '-t' flag allows us to specify the name and tag for the image. In this case, the image is named 'eeshuyadav/osdag-web-backend', and the version tag is 'v1.1'. This helps to identify different versions of the image over time. - '/home/eeshu/Osdag-web': This is the directory that contains the 'Dockerfile' and all the necessary files for building the backend image. Docker will look into this directory to create the image based on the instructions provided in the Dockerfile.

4.3.2 Build Frontend Image

Similarly, the following command builds the Docker image for the frontend service. The command is similar to the backend build command but targets the frontend directory.

```
docker build -t eeshuyadav/osdag-web-frontend:v1.1 /home/eeshu/
   Osdag-web
```

Explanation: - 'docker build': Initiates the process of building a Docker image. - '-t eeshuyadav/osdag-web-frontend:v1.1': Assigns the name 'eeshuyadav/osdag-web-frontend' and tag 'v1.1' to the image being built. This tagging ensures we can distinguish the frontend image from the backend image and helps in managing different versions of the images. - '/home/eeshu/Osdag-web': This is the directory that contains the 'Dockerfile' for the frontend, along with all required files to build the frontend Docker image.

4.3.3 Push Backend Image

Once the backend image is built successfully, the next step is to push it to Docker Hub so that it can be accessed by others. Docker Hub serves as a cloud-based registry where Docker images can be stored and shared. Use the following command to push the backend image to Docker Hub:

```
docker push eeshuyadav/osdag-web-backend:v1.1
```

Explanation: - ‘docker push‘: This command uploads a locally built image to a remote Docker registry such as Docker Hub. - ‘eeshuyadav/osdag-web-backend:v1.1‘: This specifies the name and tag of the image to push. The image should already be built and tagged with the specified version ‘v1.1‘ as done in the previous build step. - Pushing the image to Docker Hub makes it available publicly (if set to public) or privately (if a private repository is used), allowing for easy deployment on other machines or cloud platforms.

4.3.4 Push Frontend Image

Similarly, the following command pushes the frontend image to Docker Hub. This ensures that the frontend Docker image is stored remotely and can be pulled by anyone with the correct permissions.

```
docker push eeshuyadav/osdag-web-frontend:v1.1
```

Explanation: - ‘docker push‘: This command is used to upload the frontend image to Docker Hub. - ‘eeshuyadav/osdag-web-frontend:v1.1‘: This specifies the image name and version tag to be pushed. It should match the name and tag given during the build step. - By pushing the frontend image, we ensure that the frontend service can be easily deployed and scaled in environments that support Docker.

4.4 Conclusion

Building and pushing Docker images for both the backend and frontend services ensures that the Osdag application can be deployed consistently across multiple environments. By using Docker containers, we eliminate many of the challenges associated with dependency

management and system configurations. Pushing the images to Docker Hub allows us to share and distribute these containers easily, ensuring quick and reliable deployment.

In future updates, it might be useful to consider automating these steps using a Continuous Integration/Continuous Deployment (CI/CD) pipeline. This would allow for automated builds and deployments every time new code is pushed to the repository, further streamlining the development process.

Chapter 5

Using Docker Images in the Project

5.1 Introduction

This section describes the process of integrating Docker images into the Osdag project by updating the `docker-compose.yml` file. Docker images ensure consistency across different environments, simplify deployment, and facilitate easier scaling of services.

5.2 Updating `docker-compose.yml`

To utilize Docker images in the project, the `docker-compose.yml` file was updated to define and manage multiple services, including the backend, frontend, and database. Below is the updated configuration:

```
version: '3.8'

services:
  backend:
    image: eeshuyadav/osdag-web-backend:v1.1
    ports:
      - "8000:8000"
    volumes:
      - .:/app
    depends_on:
      - db
```

```
  frontend:
    image: eeshuyadav/osdag-web-frontend:v1.1
    ports:
      - "3000:3000"
    volumes:
      - ./osdagclient:/app
    environment:
      - CHOKIDAR_USEPOLLING=true
    command: npm run dev
    depends_on:
      - backend

  db:
    image: postgres:14
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
      POSTGRES_DB: mydb
    ports:
      - "5434:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

5.3 Explanation of Configuration

5.3.1 Backend Service

- **Image:** The Docker image `eeshuyadav/osdag-web-backend:v1.1` is used for the backend service.

- **Ports:** Port 8000 on the host is mapped to port 8000 in the container.
- **Volumes:** The current directory is mounted to the `/app` directory in the container to enable real-time code synchronization.
- **Depends On:** The backend service depends on the database service, ensuring the database is running before the backend starts.

5.3.2 Frontend Service

- **Image:** The Docker image `eeshuyadav/osdag-web-frontend:v1.1` is used for the frontend service.
- **Ports:** Port 3000 on the host is mapped to port 3000 in the container.
- **Volumes:** The `./osdagclient` directory is mounted to `/app` in the container.
- **Environment:** The environment variable `CHOKIDAR_USEPOLLING=true` is set to enable hot-reloading in development environments.
- **Command:** The development server is started using the command `npm run dev`.
- **Depends On:** The frontend service depends on the backend service, ensuring the backend is running before the frontend starts.

5.3.3 Database Service (PostgreSQL)

- **Image:** The official PostgreSQL Docker image (version 14) is used for the database service.
- **Environment:** Environment variables are configured for the database user (`myuser`), password (`mypassword`), and database name (`mydb`).
- **Ports:** Port 5434 on the host is mapped to port 5432 in the container.
- **Volumes:** A named volume, `postgres_data`, is used to persist database data, ensuring data is not lost when the container is stopped.

5.4 Benefits of Using Docker Images

- **Consistency:** Docker ensures that the application runs in the same environment across different machines, reducing the "it works on my machine" problem.
- **Scalability:** Docker simplifies scaling services up or down as needed, making it easier to handle varying workloads.
- **Efficiency:** Docker reduces setup time for new developers and deployment environments, streamlining the development process.
- **Isolation:** Each service runs in an isolated environment, minimizing conflicts between dependencies and configurations.

Chapter 6

Detailed Example of a CI/CD Pipeline

6.1 Step-by-Step Breakdown

- Triggering the Workflow: The workflow is triggered on pushes and pull requests to the develop1 branch
- Setting Up the Job: The job runs on the latest version of Ubuntu
- Checkout Code: The actions/checkout@v2 action checks out the repository code
- Triggering the Workflow: The workflow is triggered on pushes and pull requests to the develop1 branch
- Set Up Docker Compose: This step installs Docker Compose on the runner
- Pull Docker Images: These steps pull the backend and frontend Docker images from Docker Hub
- Start Services: This step starts the backend and frontend services using Docker Compose
- Wait for Services to Be Ready: This step waits for 30 seconds to ensure the services are up and running
- Install Frontend Dependencies: This step installs the frontend dependencies using npm install

- Run Backend Migrations: This step runs the backend migrations to set up the database schema
- Populate Database: This step populates the database with initial data
- Update Sequences: This step updates the database sequences
- Stop Services: This step stops the Docker Compose services

6.2 Full ci-cd.yml Code

Listing 6.1: Building Docker Images

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - develop1
  pull_request:
    branches:
      - develop1

jobs:
  build:
    runs-on: ubuntu-latest

    services:
      db:
        image: eeshuyadav/postgres:v1.1
        ports:
          - 5432:5432
        env:
          POSTGRES_USER: myuser
          POSTGRES_PASSWORD: mypassword
          POSTGRES_DB: mydb
```

```
steps:
  - name: Checkout code
    uses: actions/checkout@v2

  - name: Set up Docker Compose
    run: |
      sudo curl -SL https://github.com/docker/compose/
         releases/latest/download/docker-compose-$(uname -s)-
         $(uname -m) -o /usr/local/bin/docker-compose
      sudo chmod +x /usr/local/bin/docker-compose
      docker-compose --version

  - name: Pull backend Docker image
    run: docker pull eeshuyadav/osdag-web-backend:v1.1

  - name: Pull frontend Docker image
    run: docker pull eeshuyadav/osdag-web-frontend:v1.1

  - name: Start backend and frontend services
    run: docker-compose up -d

  - name: Wait for services to be ready
    run: sleep 30

  - name: Install frontend dependencies
    run: docker-compose run frontend npm install

  - name: Run backend migrations
    run: docker-compose exec backend bash -c "source /opt/
       miniconda/etc/profile.d/conda.sh && conda activate
       myenv && python manage.py migrate"

  - name: Populate database
```

```
      run: docker-compose exec backend bash -c "source /opt/
        miniconda/etc/profile.d/conda.sh && conda activate
        myenv && python populate_database.py"

    - name: Update sequences
      run: docker-compose exec backend bash -c "source /opt/
        miniconda/etc/profile.d/conda.sh && conda activate
        myenv && python update_sequences.py"

    - name: Run tests
      run: docker-compose exec backend bash -c "source /opt/
        miniconda/etc/profile.d/conda.sh && conda activate
        myenv && pytest"

    - name: Stop services
      run: docker-compose down
```

Chapter 7

Conclusions

7.1 Tasks Accomplished

Throughout the course of this project, several critical tasks were successfully completed, contributing to the development and deployment of the Osdag project. The key accomplishments are summarized below:

- **CI/CD Pipeline Implementation:** Designed and implemented a robust Continuous Integration and Continuous Deployment (CI/CD) pipeline using GitHub Actions and Docker. This pipeline automates the build, test, and deployment processes, ensuring code integrity and rapid deployment cycles.
- **Docker Integration:** Created Dockerfiles for the backend, frontend, and PostgreSQL database services, and built Docker images to standardize development and deployment environments.
- **Docker Compose Configuration:** Updated the `docker-compose.yml` file to manage multi-container Docker applications efficiently, linking backend, frontend, and database services seamlessly.
- **Version Control Management:** Established workflows in GitHub Actions to automate code quality checks, run unit tests, and deploy updates to production environments automatically.
- **Documentation:** Provided detailed documentation on project architecture, CI/CD processes, and Docker configurations to ensure knowledge transfer and ease of future

maintenance.

7.2 Skills Developed

During the fellowship, both technical and professional skills were significantly enhanced. The key skills developed include:

7.2.1 Technical Skills

- **CI/CD Tools:** Gained hands-on experience with GitHub Actions for automating software workflows and enhancing deployment pipelines.
- **Containerization:** Developed expertise in Docker, including creating Dockerfiles, managing Docker images, and orchestrating containers using Docker Compose.
- **Version Control:** Improved proficiency in Git and GitHub, including branching strategies, pull requests, and integrating CI workflows.
- **Scripting and Automation:** Enhanced skills in writing scripts to automate repetitive tasks, optimize build processes, and improve deployment efficiency.
- **Database Management:** Worked with PostgreSQL, configuring it within Docker containers and managing data persistence effectively.

7.2.2 Professional Skills

- **Project Management:** Learned to plan, execute, and monitor project tasks effectively, ensuring timely completion of deliverables.
- **Collaboration:** Strengthened teamwork and communication skills through active collaboration with project stakeholders and team members.
- **Problem-Solving:** Developed critical thinking and problem-solving abilities by troubleshooting CI/CD pipeline issues and optimizing workflows.
- **Documentation and Reporting:** Improved technical writing skills by documenting processes, configurations, and project summaries clearly and concisely.

- **Adaptability:** Adapted to new tools and technologies quickly, demonstrating flexibility in handling project challenges and changes.

This project not only enhanced technical competencies but also fostered a professional growth mindset, preparing for future challenges in software development and DevOps environments.

Chapter A

Appendix

A.1 Work Reports

Name	Eeshu		
Project	Osdag		
Internship	FOSSEE Winter Fellowship 2024		
Date	**Day**	**Task Description**	**Hours Spent**
13 Nov 2024	Wednesday	Overview of GitHub Actions	7
14 Nov 2024	Thursday	Overview of Docker and containerization	6
15 Nov 2024	Friday	Setting up GitHub repository for CI/CD pipeline	5
16 Nov 2024	Saturday	Creating workflow directory and initial YAML file	6
17 Nov 2024	Sunday	Exploring GitHub Actions Marketplace for reusable actions	5
18 Nov 2024	Monday	Defining CI workflow triggers and environment setup	7
19 Nov 2024	Tuesday	Creating Dockerfiles for Osdag backend and frontend	6
20 Nov 2024	Wednesday	Building and testing Docker images locally	7
21 Nov 2024	Thursday	Setting up Docker Hub account and repositories	5
22 Nov 2024	Friday	Pushing Docker images to Docker Hub	6
23 Nov 2024	Saturday	Automating Docker image builds with GitHub Actions	7
24 Nov 2024	Sunday	Debugging GitHub Action failures during image builds	6
25 Nov 2024	Monday	Integrating Docker with docker-compose.yml	7
26 Nov 2024	Tuesday	Implementing automated testing in CI/CD pipeline	6
27 Nov 2024	Wednesday	Running tests and debugging CI workflow failures	8
28 Nov 2024	Thursday	Optimizing workflow with caching and parallel jobs	6
29 Nov 2024	Friday	Managing GitHub Secrets for secure environment variables	7
30 Nov 2024	Saturday	Adding database migration steps in CI pipeline	6
01 Dec 2024	Sunday	Conducting performance tests of the CI/CD workflow	7
02 Dec 2024	Monday	Troubleshooting Docker build and deployment issues	6
03 Dec 2024	Tuesday	Refining Docker Compose services for production readiness	7
04 Dec 2024	Wednesday	Writing comprehensive CI/CD pipeline documentation	6
05 Dec 2024	Thursday	Internal review of CI/CD implementation with the team	7
06 Dec 2024	Friday	Implementing feedback and fixing identified issues	6
07 Dec 2024	Saturday	Adding monitoring and logging for CI workflows	7
08 Dec 2024	Sunday	Running integration tests with external services	6
09 Dec 2024	Monday	Final round of automated deployment tests	7
10 Dec 2024	Tuesday	Preparing final presentation of CI/CD pipeline	6

11 Dec 2024	Wednesday	Presenting CI/CD workflow outcomes to the team	7
12 Dec 2024	Thursday	Collecting feedback and documenting improvements	6
13 Dec 2024	Friday	Making final adjustments to the pipeline	7
14 Dec 2024	Saturday	Reviewing Docker image management strategies	6
15 Dec 2024	Sunday	Conducting end-to-end deployment tests	7
16 Dec 2024	Monday	Final cleanup of GitHub Actions workflows	5
17 Dec 2024	Tuesday	Preparing final documentation and reports	6
18 Dec 2024	Wednesday	Submission of CI/CD pipeline report	7
19 Dec 2024	Thursday	Reviewing project outcomes with the mentor	6
20 Dec 2024	Friday	Addressing final feedback and polishing the CI/CD setup	7
21 Dec 2024	Saturday	Reviewing best practices for CI/CD workflows	5
22 Dec 2024	Sunday	Documentation review and last-minute refinements	6
23 Dec 2024	Monday	Preparing a case study report on CI/CD implementation	7
24 Dec 2024	Tuesday	Final meeting with mentor to discuss the project outcome	6
25 Dec 2024	Wednesday	General project review, learning, or support activities	6
26 Dec 2024	Thursday	Wrap-up of any pending issues in CI/CD deployment	5
27 Dec 2024	Friday	Final documentation submission	6
28 Dec 2024	Saturday	Project completion review and feedback session	7
29 Dec 2024	Sunday	General project review, learning, or support activities	6
30 Dec 2024	Monday	General project review, learning, or support activities	6

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta. Osdag: A Software for Structural Steel Design Using IS 800:2007. In Sondipon Adhikari, Anjan Dutta, and Satyabrata Choudhury, editors, *Advances in Structural Technologies*, volume 81 of *Lecture Notes in Civil Engineering*, pages 219–231, Singapore, 2021. Springer Singapore.

[2] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.

[3] FOSSEE Project. Osdag website. Accessed: 2024-12-05.