# Continuous Integration and Deployment Pipeline

Sprint Number 3
Date 11/3/2024


| **Name** | **Email Address** |
|---|---|
| John Gahagan | john.gahagan398@topper.wku.edu |
| Jay Mistry | jay.mistry627@topper.wku.edu |


CS 396
Fall 2024
Project Organization Documentation

# Contents

# List of Figures

# 1 Project Team's Organizational Approach

Our team divided responsibilities based on individual strengths to ensure both technical progress and thorough documentation. By integrating version control, automated testing and deployment tools, we aimed to reduce manual work and improve development efficiency, ensuring system's consistency and security at every stage. John Gahagan focused on the technical aspects, including back end and front end development, setting up the CI/CD pipeline, and configuring Docker environments. Jay Mistry handled organizational tasks, including documentation, progress tracking, and ensuring that project requirements were met. Jay also organized communication during meetings and kept a record of action items to ensure smooth project flow. We met twice a week using Discord and held in-person meetings when needed. These meetings allowed us to discuss progress, address problems, and realign project milestones if issues were to happen. Meetings followed a structured plan to help maintain focus and reduce miscommunication. The CI/CD pipeline itself was structured to automate development life cycle comprehensively. The application was then built using Docker to ensure consistent environments across different deployment stages. In addition to meetings, we used a dedicated Discord channel for updates and quick questions, which allowed us to fix easy problems or questions without having to meet in person. Our key decisions and action items were documented. This shows us maintaining a clear project history for reference. In case, we must go back through our history and find issues there. This process makes it nice and organized when an error occurs we can pinpoint it. Our setup relied on several key tools. GitHub managed version control, GitHub Actions allowed use to build the CI/CD pipeline, Docker enabled use to maintain builds, Jest allowed us to automate test cases, etc. Overall, our approach of structured meetings, deciding which tasks each member will do, and consistent communication through Discord, phone, and biweekly meetings, allowed us to address technical and organizational aspects efficiently, ensuring successful CI/CD pipeline implementation. By assigning roles based on strengths and establishing clear communication channels, we addressed both technical and organizational challenges effectively and were able to build a project that fulfilled all the requirements.

# 2 Schedule Organization

The project schedule was visualized using a Gantt chart, updated at the start of each sprint to reflect current objectives and deadlines. The Gantt chart is located in the project's root directory it is named Sprint 3 Gantt Chart.xslx as well as a image of the Gantt Chart in the root directory GanntChart3.png.

## 2.1 Sprint Focus and Task Breakdown

This sprint's primary focus was CI/CD pipeline implementation. Key tasks included:

- **Setting Up Version Control Integration**: Making sure that the CI/CD pipeline integrates with Git to trigger builds on new code commits or pull requests. This step maintained version control and supported collaborative and automated. development. Version control allowed quick rollbacks if issues arose, minimizing downtime and ensuring and reliability. This continuous integration approach provided immediate feedback, streamlined development and upheld stability for healthcare information system.

- **Automating Builds and Tests**: Configuring automated build processes for back end and front end applications. These processes included unit, integration, and end-to-end tests to validate functionality. By automating these tests, we could quickly identify and resolve any bugs, guaranteeing that new features or fixes didn't compromise the system's overall performance and reliability. This setup not only enhanced code quality but also accelerated feedback loop, enabling faster, more reliable deployments.

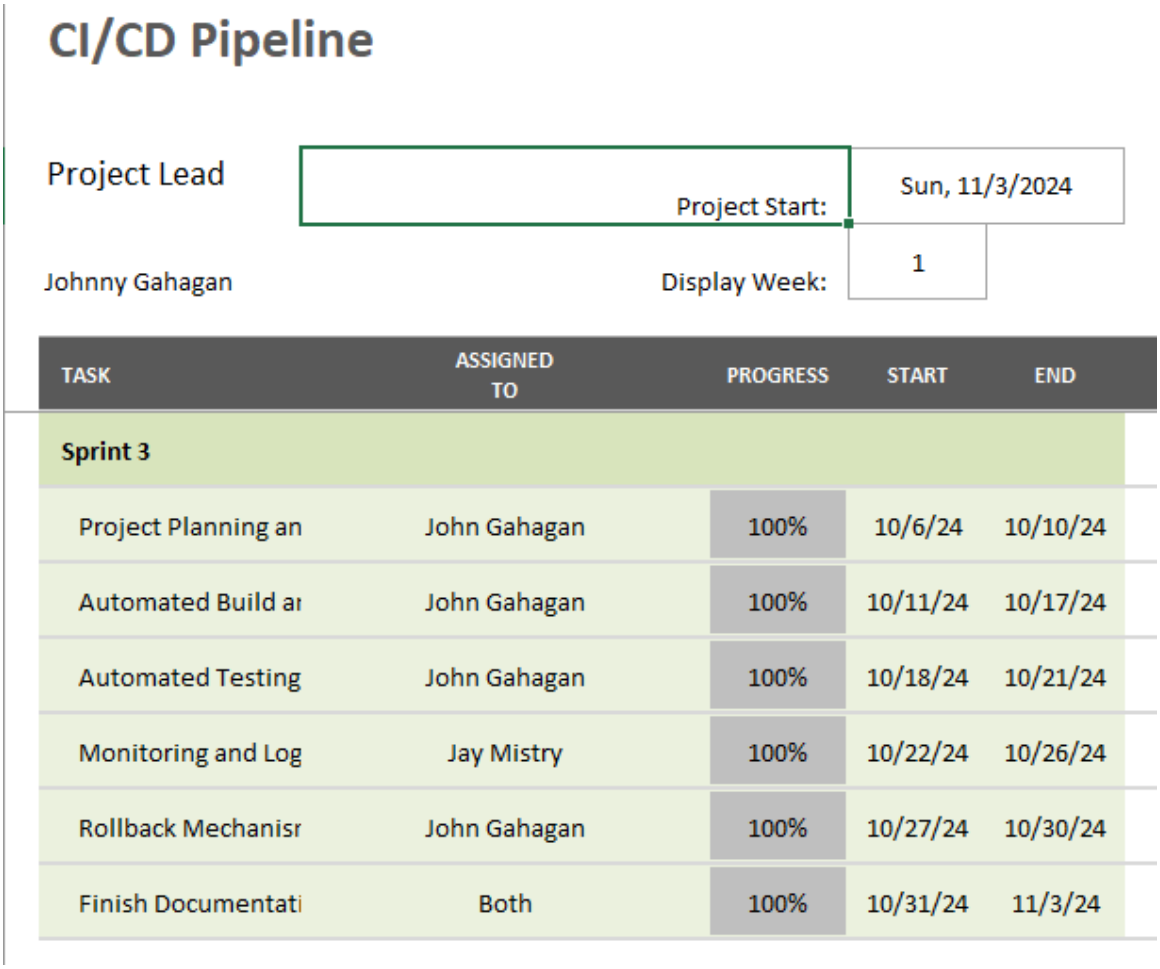
- **Configuring Deployment Environments**: Setting up deployment configurations and environment variables for staging and production. Docker containerization enabled consistent deployment across environments. This setup enabled smooth transitions from staging to production, minimized configuration errors and ensured a stable deployment process, critical for maintaining the readability of healthcare information system.

- **Implementing Rollback Mechanisms**: Adding a rollback feature to revert to a stable version if a deployment fails, minimizing downtime. This setup was particularly important for maintaining the reliability

and availability of health care information system, ensuring that any issues in new releases could be swiftly addressed without interrupting service or compromising patient data integrity.

### 2.2 Gantt Chart:

This sprint included critical tasks such as project planning, setting up an automated build process, implementing automated testing, and establishing continuous monitoring and logging. Additionally, a rollback mechanism was introduced to ensure system stability in case of deployment failures. The sprint concluded with final documentation, consolidating all efforts and ensuring comprehensive coverage of the pipeline's design and functionality. Each task was assigned to team members and tracked for progress, achieving full completion by the end date. The Gantt Chart for Sprint 3 is located in the root of the project directory as GanntChart3.png. It is also in excel format in the root of the project directory to as Sprint 3 Gantt Chart.xlsx.



CI/CD Pipeline

Project Lead

Johnny Gahagan

Project Start: Sun, 11/3/2024

Display Week: 1

| TASK | ASSIGNED TO | PROGRESS | START | END |
|---|---|---|---|---|
| **Sprint 3** | | | | |
| Project Planning an | John Gahagan | 100% | 10/6/24 | 10/10/24 |
| Automated Build ar | John Gahagan | 100% | 10/11/24 | 10/17/24 |
| Automated Testing | John Gahagan | 100% | 10/18/24 | 10/21/24 |
| Monitoring and Log | Jay Mistry | 100% | 10/22/24 | 10/26/24 |
| Rollback Mechanisr | John Gahagan | 100% | 10/27/24 | 10/30/24 |
| Finish Documentati | Both | 100% | 10/31/24 | 11/3/24 |

Figure 1: Sprint 3 Gantt Chart - CI/CD Pipeline Implementation

## 3 Progress Visibility

Each team member updated the group on their progress through Discord and shared completed work for review. This enabled us to have quick support for any challenges. Team members frequently committed code updates to the repository and conducted informal code reviews to ensure quality and alignment with requirements. Our team wished to have good consistency and quality as enabling swift adjustments would help us align with the given project requirements. This setup promoted accountability and facilitated timely support for any challenges, ensuring smooth, collaborative progress. Weekly meetings allowed us to review completed tasks, address challenges, and adjust priorities. This setup kept us organized and aligned with project goals.

# 4 Risk Management

Our main risk involved achieving seamless deployment across diverse environments. To address this, we incorporated compatibility testing, backup strategies, continuous monitoring, and tackled communication risks. We used Docker to test the pipeline across multiple environments, identifying potential compatibility issues. A rollback mechanism acted as a safeguard against deployment failures, helping maintain application availability. Real-time monitoring with health checks and logging allowed prompt response to any issues, and the logs offered valuable data for troubleshooting and optimization. A big issue for a lot of teams are communication risks. This could lead to duplicated work or missed deadlines. To lower the odds of this happening, we used a structured communication plan featuring regular meetings and documented updates this would help keep the team aligned. We built buffer time into our Gantt chart to accommodate any unexpected delays, and the project timeline was reviewed periodically to adjust deadlines as needed. After each task, we assessed any challenges to refine our approach, showcasing the pipeline's resilience and adaptability. Our risk management approach combined technical safeguards, organized communication, and proactive planning to support successful project completion.