CLIENT NAME
Computer Science Department
Swinburne Vietnam Alliance Program HCMC

PROJECT

Secure CI/CD Pipeline for Application Development

Version 1.0
Date: 02/06/2024

Prepared by: GROUP 1

- Dang Vi Luan - 103802759
- Nguyen Linh Dan – 103488557
- Nguyen Duy Khang - 104056476
- Tran Bao Huy - 103505799

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY 1
2. BACKGROUND 1
3. OBJECTIVES 1
4. SCOPE 2
4.1. Functional Requirements 2
4.2. Non-Functional Requirements 3
5. OUT OF SCOPE 4
6. CONSTRAINTS AND DEPENDENCIES 4
7. ASSUMPTIONS 5
8. RISKS 5
9. DELIVERABLES 7
10. ISSUES 8
11. PROJECT TEAM STRUCTURE 9
11.1. Project Chart 9
11.2. Roles And Responsibilities 9
12. PROJECT SIGNOFF 10
APPENDIX A – GANTT CHART 11

LIST OF FIGURES

Figure 1: Secured CI/CD Pipeline Diagram 7
Figure 2: Project Team Structure 9

LIST OF TABLES

Table 1: Role and Responsibilities matrix 10

REVISION HISTORY

Version	Date Released	Comments/Changes
1.0	02/06/2024	First Draft

1. EXECUTIVE SUMMARY

In the current technological world, automatic CI/CD pipelines has become widely popular due to its superiority in ensuring efficiency, automation, collaboration, and security. A CI/CD pipeline, which stands for Continuous Integration and Continuous Deployment, can integrate code changes from multiple developers within a team into a shared repository to foster better teamwork in large development projects. This project proposal aims to develop a CI/CD pipeline solution for the Computer Science department at Swinburne Vietnam Alliance Program at HCMC. The CI/CD solution is strongly believed to enhance the learning experience and prepare the students for real-world practices. The report proposal provides information related to the background, goals and objectives, scope, constraints and dependencies, assumptions, risks, issues, deliverables, and team structure for the CI/CD project.

2. BACKGROUND

Computer Science department at Swinburne Vietnam Alliance Program at HCMC is a renowned institution in teaching computer science courses in the country. The study curriculum at Swinburne Vietnam Alliance Program provides students with the opportunities to acquire real working experience by developing and deploying multiple application projects within the study program.

Currently, the Computer Science department is using conventional software development lifecycle methodologies for students to develop their projects. In other words, the development, testing, and deployment phases are separated and conducted consecutively. With this system, the students are having difficulties working together in a large group. They are unable to combine each other's codes or debug safely and effectively after code merging. Additionally, it is very difficult to keep track of the changes made and the personnel responsible for those changes. The students will not be able to receive valuable feedback from lecturers constantly enough to foster continuous improvement. Overall, the conventional process of application development offers little to no flexibility, collaboration, and invulnerability from incidents. Additionally, as the student base is growing, the lecturers are also having a hard time managing all student projects in an effective and secured manner.

As a solution to the mentioned problems, the Computer Science department has hired our team of developers and technicians to design and implement an automatic CI/CP pipeline to facilitate the development and deployment of students' application projects.

3. OBJECTIVES

Per discussion with the client, we determined the following objectives for the CI/CD pipeline project:

- **Design the complete CI/CD pipeline**
  Using the team's technical expertise and detailed discussion with the client to produce a CI/CD pipeline design that satisfies all client's requirements.
- **Prepare infrastructure for the CI/CD pipeline:**
  Prepare and configure the required tools and infrastructure such as GitHub, Terraform Registry, DockerHub, ArgoCD, Kubernetes, and AWS services to fully support the CI/CD pipeline.
- **Implement the CI/CD pipeline:**
  Perform development and implementation of the real solution into the prepared infrastructure to ensure a smooth operation as intended in the planning.
- **Perform testing after implementation:**
  Perform technical testing and user acceptance testing (UAT) to ensure the system works as intended and the user experience quality is ensured.
- **Prepare User Guide and Training Document:**

Prepare comprehensive user guide and training materials to help the students get familiar with the Agile application development methodology and technical aspects of the CI/CD pipeline.

- **Feedback gathering from client:**
  Gather feedback from lecturers and professors to evaluate the effectiveness of the system to implement any additional change if needed.

4. SCOPE

4.1. Functional Requirements

A. DevOps CI/CD Pipeline:

A1. Version Control Integration

- Integrate the codebase with a version control system using Git to manage source code changes and enable collaboration among developers.
- Set up repositories for the main application, associated services, and any necessary infrastructure code using GitHub, Terraform Registry, and DockerHub.

A2. Continuous Integration (CI)

- Implement automated build and testing processes to ensure the integrity of the codebase.
- Integrate ArgoCD as a tool to automatically trigger build upon code commits or merges
- Implement unit tests, integration tests, and any necessary static code analysis to validate the application's functionality and code quality.
- Provide feedback to developers on the success or failure of the build and test processes.

A3. Deployment

- Integrate the CI/CD pipeline with Docker to build and push container images to a registry, using versioning schemes to maintain traceability.
- Implement Dockerfiles for services and components and configure the CI process to automatically build and publish the Docker images.

A4. Operation

- Leverage Terraform to provision the required cloud infrastructure, integrating the Terraform code into the CI/CD pipeline for consistent provisioning across environments.
- Deploy the containerized application to a Kubernetes cluster, automating the deployment of Kubernetes manifests as part of the CD process.

B. DevSecOps Integration

B1. Version Control Integration

- Implement pre-commit hooks using git-secrets to prevent accidental commit of sensitive information such as credentials or access keys.
- Integrate AWS Secrets Manager to control access to passwords and other credentials across services

B2. Continuous Integration

- Implement Sonarqube as Static Application Security Testing (SAST) to scan source code for security vulnerabilities, leverage tools that provide incremental testing and minimize false positives.
- GitHub can be used again for Source Composition Analysis (SCA) to scan for vulnerable libraries.
- Integrate OWASP ZAP to identify vulnerabilities in running applications

B3. Deployment:

- Scan Docker images for vulnerabilities using tools like Clair to ensure the security of the containerized infrastructure.

B4. Monitoring and alerting

- Use Prometheus for collecting and storing time-series metrics from the application and infrastructure components.
- Integrate Grafana to create custom dashboards and visualizations for monitoring the overall system health and performance.
- Configure Prometheus to set up alerts based on predefined thresholds or anomalous behavior.
- Integrate the alerting system with Telegram to send notifications to the appropriate team members when alerts are triggered, enabling timely incident response.

4.2. Non-Functional Requirements

4.2.1. Performance

- Scalability: The CI/CD pipeline must handle increasing workloads as the number of developers and the size of the codebase grows. It should support horizontal scaling to accommodate additional resources as needed.
- Throughput: The pipeline should process a high volume of builds, tests, and deployments efficiently, minimizing delays and bottlenecks.
- Latency: Ensure minimal latency in the build, test, and deployment stages to provide quick feedback to developers and reduce the overall development cycle time.

4.2.2. Reliability

- Availability: The CI/CD pipeline should be highly available with minimal downtime. Implement redundancy and failover mechanisms to ensure continuous operation.
- Fault Tolerance: Design the pipeline to tolerate failures and recover gracefully without data loss or significant interruption to service.

4.2.3. Integrability

- Tool Compatibility: Ensure compatibility with various development tools and platforms, including version control systems (e.g., GitHub), CI tools (e.g., ArgoCD), and container registries (e.g., DockerHub).

4.2.4. Observability

- Metrics Collection: Use Prometheus to collect detailed metrics on pipeline performance, application health, and infrastructure status.
- Visualization: Integrate Grafana for creating visual dashboards to monitor metrics and system health in real time.

- Alerting: Configure Prometheus and integrate with Telegram for real-time alerting on predefined thresholds and anomalous behaviors to ensure timely response to incidents.

By adhering to these functional requirements and nonfunctional requirements, the CI/CD pipeline will be robust, secure, efficient, and user-friendly, ensuring that it meets the needs of the Swinburne's Computer Science Department.

5. OUT OF SCOPE

The following elements are explicitly out of scope for this project:

- **Extensive Training for Cloud Services**: Our team will provide knowledge transfer and documentation on how to use the CI/CD pipeline. However, extensive training related to cloud services and additional features on integrated services (e.g., advanced AWS features, in-depth Terraform training) will not be within our scope of work.
- **Marketing and User Adoption Plans**: Developing and executing marketing strategies or user adoption plans to promote the new CI/CD pipeline within the organization is not included.
- **Third-Party Tools and Services**: While we will integrate various third-party tools (e.g., ArgoCD, GitHub, Docker, Terraform), the ongoing management, maintenance, and support for these tools are not within our scope. This includes troubleshooting issues directly related to these tools and their updates.
- **Bug Fixing and Additional Functionality for Applications**: The project is focused solely on the integration and deployment of applications via the CI/CD pipeline. Any bug fixing or development of additional functionalities for the applications themselves are out of scope. This includes addressing bugs that arise in the application code and adding new features to the applications.

6. CONSTRAINTS AND DEPENDENCIES

The project to design and deploy a secure CI/CD pipeline for application development faces the constraints of:

- **Budget Limitations:** The project is not initially supported with any budget, necessitating the prioritization of free, open-source tools. This may pose compliance challenges later in the deployment stages if regulations require specific tools or licenses.
- **Timeframe:** Completion of this project is limited within a 12-week timeframe, requiring efficient project management and resource allocation to meet deadlines.
- **Resource Availability:** Although our team possesses skills in solution architecture, DevOps, security, and auditing, we lack dedicated developers for all sections, such as web or desktop applications. This will require additional human resources consultancy should it be necessary
- **Infrastructure Limitations:** Currently, Swinburne's Computer Science department lacks infrastructure, regulations, and documentation for student projects. Therefore, we rely heavily on our assumptions and experience to guide the project.

Apart from key constraints, we also acknowledge some important dependencies that need to be accomplished for the success of this project:

- **Development Team Cooperation:** The successful implementation of the project hinges upon active collaboration among team members and stakeholders.

- **Tool and Technology Integration:** The project's success is contingent upon the seamless integration of selected CI/CD tools with various DevSecOps tools.
- **Knowledge Transfer and Documentation:** Given the introduction of new concepts such as DevSecOps, it is imperative that knowledge transfer for key services and comprehensive documentation for all services receive meticulous attention for post-project utilization.
- **Stakeholder Engagement:** Continuous engagement with stakeholders, including mentors, students, and lecturers from Swinburne's Computer Science department, is indispensable for gathering requirements, making informed decisions, and ensuring alignment with project goals.

7. ASSUMPTIONS

- Creation of a new code pipeline is not part of scope for this engagement. Project team 1 would assist the customer in modifying the existing pipeline to deploy changes to the AWS environment.
- Customer’s Application vendor or internal application support team will be responsible for the application testing on AWS. Project team 1 will provide the necessary infrastructure support to enable the testing.
- Any issues related to Infra will be resolved through AWS Support via the customer’s support plan.
- The authentication services of the Application will be managed by the customer.
- Project team 1 will provide Knowledge Transfer (KT) to the Customer's team on the provisioned Infra. However, Training related to AWS Services are not part of current scope of work
- Any changes to requirements or scope of work will go through the change management process. Such changes will be allowed only after a mutual agreement between the Parties in writing (emails permitted)
- Security is a shared responsibility between Customer, Project team 1 and the Cloud Provider, in this case AWS, Github, Sonarcube, and other 3rd party tools.
- Regarding credentials log in, Project team 1 mandates implementation of MFA for the root account credentials. If the root account is owned by Project team 1, MFA will be enabled by default. If the root credentials are owned by the Customer, Project team 1 recommends implementing MFA as one of the first set of activities before commencing the project.
- Project team 1 will not ask for or retain the root access to the Customer’s servers and/or databases.
- Project team 1 and Customer will discuss the need for different users from Customer and Customer’s Application vendor (if any) side. These users will be created as AWS IAM users with least privileges needed to perform the respective actions.

8. RISKS

Below we outline the possible risks and their mitigation:

Infrastructure Provisioning Failures

- Risk: Errors or inconsistencies in the Terraform configuration could lead to failed infrastructure deployments or unintended changes to the environment.
- Mitigation:
    - Implement thorough testing of Terraform configurations, including unit tests and integration tests, to catch issues before deployment.

    - Leverage Terraform's state management capabilities to maintain a consistent view of the deployed infrastructure.
    - Implement a review process for all Terraform changes to ensure they align with the desired state.

Security Vulnerabilities

- Risk: The introduction of security vulnerabilities in the codebase or deployed infrastructure could lead to potential data breaches or system compromises.
- Mitigation:
    - Integrate SonarQube into the CI/CD pipeline to continuously scan the codebase for security vulnerabilities and code quality issues.
    - Ensure timely patching and updates of all software components, including the underlying infrastructure.
    - Implement secure coding practices and regular code reviews to identify and address security concerns.

Deployment Failures

- Risk: Errors or issues during the deployment process could result in downtime or the introduction of bugs in the production environment.
- Mitigation:
    - Implement a robust CI/CD pipeline with well-defined stages (e.g., build, test, staging, production) to catch issues early in the process.
    - Utilize Argo CD's capabilities for maintaining the desired state of the Kubernetes cluster and rolling back deployments in case of issues.
    - Implement canary or blue-green deployment strategies to minimize the impact of deployment failures.

Operational Complexity

- Risk: The complexity of managing multiple tools and technologies (GitHub, SonarQube, Terraform, Argo CD) could lead to administrative overhead and increased maintenance burden.
- Mitigation:
    - Provide comprehensive documentation and handover session for the CI/CD pipeline and the related tools and processes.
    - Designate a dedicated team or individuals responsible for maintaining and updating the CI/CD infrastructure.
    - Regularly review and optimize the CI/CD pipeline to ensure efficiency and simplicity.

Vendor Lock-in

- Risk: Heavy reliance on a specific cloud provider (AWS) or tool (GitHub, SonarQube, Argo CD) could potentially lead to vendor lock-in and limit future flexibility.
- Mitigation:
    - Design the CI/CD pipeline to be as cloud agnostic as possible, using technologies and tools that can be easily integrated with other platforms.

- Periodically evaluate alternative tools and providers to ensure the solution remains competitive and flexible.
- Implement a clear migration plan or exit strategy in case the need to switch providers or tools arises in the future.

9. DELIVERABLES

The project aims to deliver several key outcomes and deliverables, ensuring the successful implementation of a secured CI/CD Pipeline:

Secured CI/CD Pipeline Design: A comprehensive design encompassing Version Control Integration, Continuous Integration, Deployment, and DevSecOps Integration. This design will integrate various tools and technologies to establish a robust and secure development and deployment pipeline. The tools utilized for these components are summarized in the diagram below.

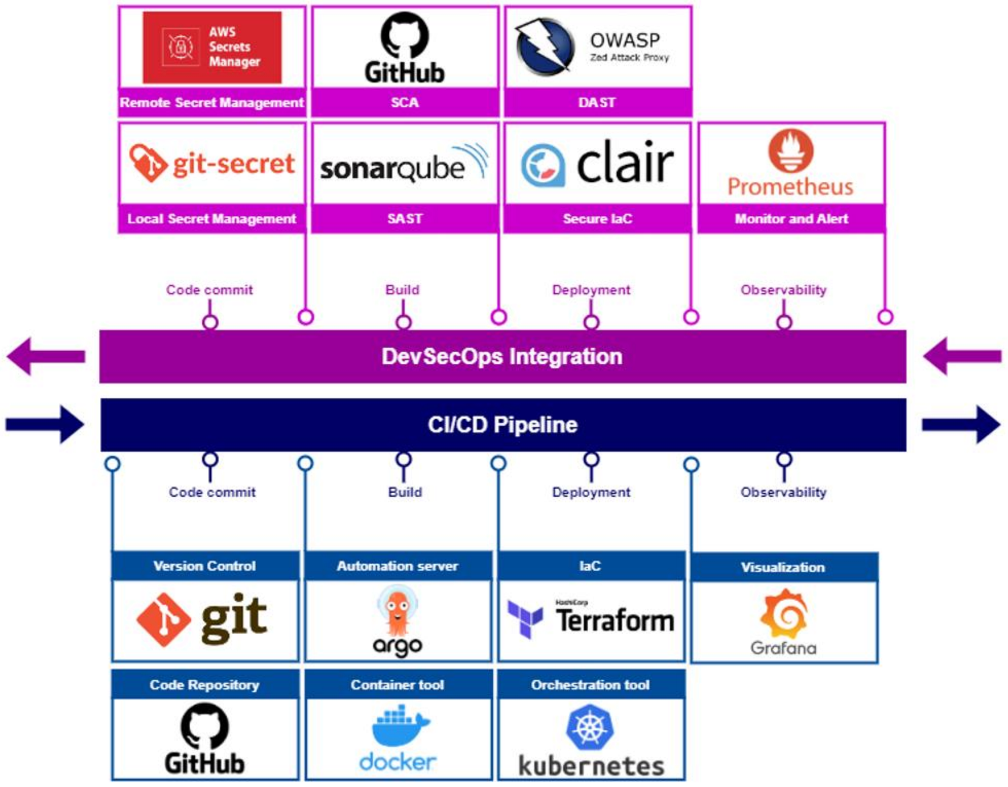


Figure 1: Secured CI/CD Pipeline Diagram

Test Report: A detailed report documenting the testing procedures and outcomes conducted on the CI/CD pipeline providing insights into the effectiveness and reliability of the pipeline in ensuring code integrity and security.

Project Completion Report: A comprehensive report summarizing the project's objectives, activities, achievements, and outcomes serving as a formal documentation of the project's completion and will include insights into the challenges faced, lessons learned, and recommendations for future improvements.

Setup Documentation and User Knowledge Transfer: Documentation outlining the setup procedures and configurations for key services involved in the CI/CD pipeline. Additionally, user knowledge transfer documentation will be provided to facilitate efficient utilization of the pipeline by relevant stakeholders.

10. ISSUES

Integration Complexity

- Issue: Integrating the various tools and technologies (GitHub, SonarQube, Terraform, Argo CD) within the CI/CD pipeline may introduce complexity and require significant effort to configure and maintain.
- Mitigation:
  - Allocate sufficient time and resources for thorough planning and testing of the tool integrations.
  - Provide comprehensive documentation and training for the development and operations teams to ensure they understand the integration process.
  - Regularly review and optimize the integration points to maintain efficiency and reliability.

Skill Gaps

- Issue: The development and operations teams may lack familiarity or expertise with some of the technologies (e.g., Terraform, Argo CD) used in the CI/CD pipeline, which could impact the implementation and ongoing maintenance.
- Mitigation:
  - Provide targeted training and knowledge-sharing sessions to upskill the team members on the relevant technologies.
  - Hire or partner with subject matter experts to assist with the initial setup and knowledge transfer.
  - Establish a continuous learning and development program to ensure the team stays up to date with the latest tools and best practices.

Data Privacy and Compliance

- Issue: The CI/CD pipeline may need to handle sensitive data or operate in a highly regulated environment, requiring additional considerations for data privacy and compliance.
- Mitigation:
  - Conduct a thorough review of the data flows and processing within the CI/CD pipeline to identify any potential privacy or compliance concerns.
  - Implement appropriate security controls, such as data encryption, access management, and audit logging, to ensure the protection of sensitive information.
  - Align the CI/CD solution with the relevant data privacy and compliance regulations (e.g., GDPR, HIPAA, PCI-DSS) and obtain necessary certifications or approvals.

Scalability and Performance

- Issue: As the application and infrastructure grow, the CI/CD pipeline may need to handle increasing volumes of code, tests, and deployments, which could impact its scalability and performance.
- Mitigation:
  - Design the CI/CD pipeline with scalability in mind, leveraging features and capabilities of the chosen tools (e.g., GitHub Actions, SonarQube, Argo CD) to handle increased workloads.
  - Implement monitoring and alerting mechanisms to proactively identify performance bottlenecks or resource constraints.

    - Establish a plan for scaling the CI/CD infrastructure (e.g., adding more build agents, expanding storage) as the project requirements evolve.

Change Management

- Issue: Introducing changes to the CI/CD pipeline, such as updating tools, modifying configurations, or introducing new workflows, could disrupt the existing processes and potentially impact the development and deployment workflows.
- Mitigation:
    - Establish a structured change management process to review, test, and approve changes to the CI/CD pipeline.
    - Implement version control and rollback capabilities to enable easy reversal of changes in case of issues.
    - Communicate changes to the development and operations teams, providing clear documentation and training to ensure a smooth transition.

11. PROJECT TEAM STRUCTURE

11.1. Project Chart

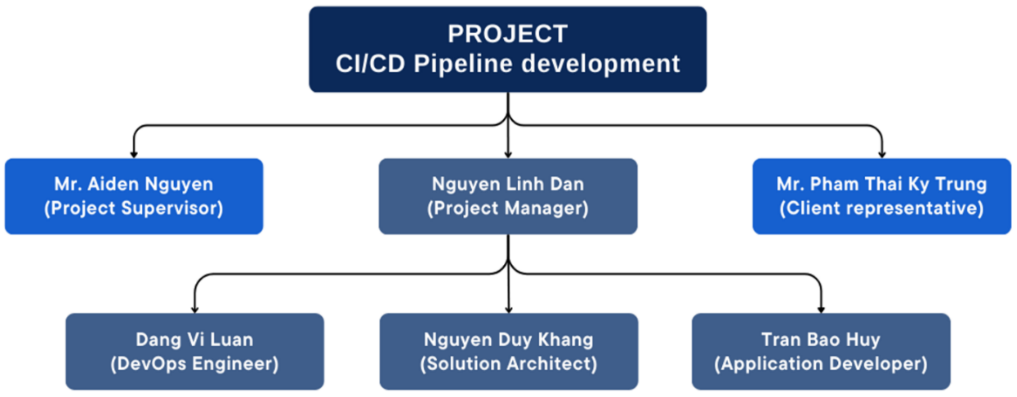

PROJECT
CI/CD Pipeline development
Mr. Aiden Nguyen (Project Supervisor)
Nguyen Linh Dan (Project Manager)
Mr. Pham Thai Ky Trung (Client representative)
Dang Vi Luan (DevOps Engineer)
Nguyen Duy Khang (Solution Architect)
Tran Bao Huy (Application Developer)

Figure 2: Project Team Structure

11.2. Roles And Responsibilities

Role	**Responsibilities**
Project Supervisor	Provides guidance and oversight to the project manager and development team. Ensures the project aligns with organizational goals and standards defined by the clients. Support the project team in identifying and resolving any problems or risks.
Client Representative	The primary contact point between the client and the project team. Communicates and conveys the client requirements, feedback to the project team. Helps project team to answer questions related to their organizational concerns or changes

Project Manager	Responsible for project planning, management, coordination, and stakeholder management. Ensures the project can deliver the final product on time and within budget.
DevOps Engineer	Designs and implements the CI/CD pipeline, including automation of build, test, and deployment processes. Integrates the pipeline with security measures to ensure the the pipeline safety.
Solution Architect	Defines the application architecture and oversees the development of secure, scalable, and maintainable software components.
Application Developer	Implements the application features and functionality, following secure coding practices and integrating with the CI/CD pipeline.

Table 1: Role and Responsibilities matrix

12. PROJECT SIGNOFF

For and on behalf of Computer Science Department – Swinburne Vietnam Alliance Program HCMC:

Signed: [illegible] Digitally signed by Pham Thai Ky Trung DN: cn=Pham Thai Ky Trung, o=FEHCM, ou=FPT-Swinburne, email=trungptk@fe.edu.vn, c=VN Date: 2024.05.24 11:48:10 +07'00' Date: 02/06/2024

(Mr. Pham Thai Ky Trung - CS Department Representative)

For and on behalf of the Project Team:

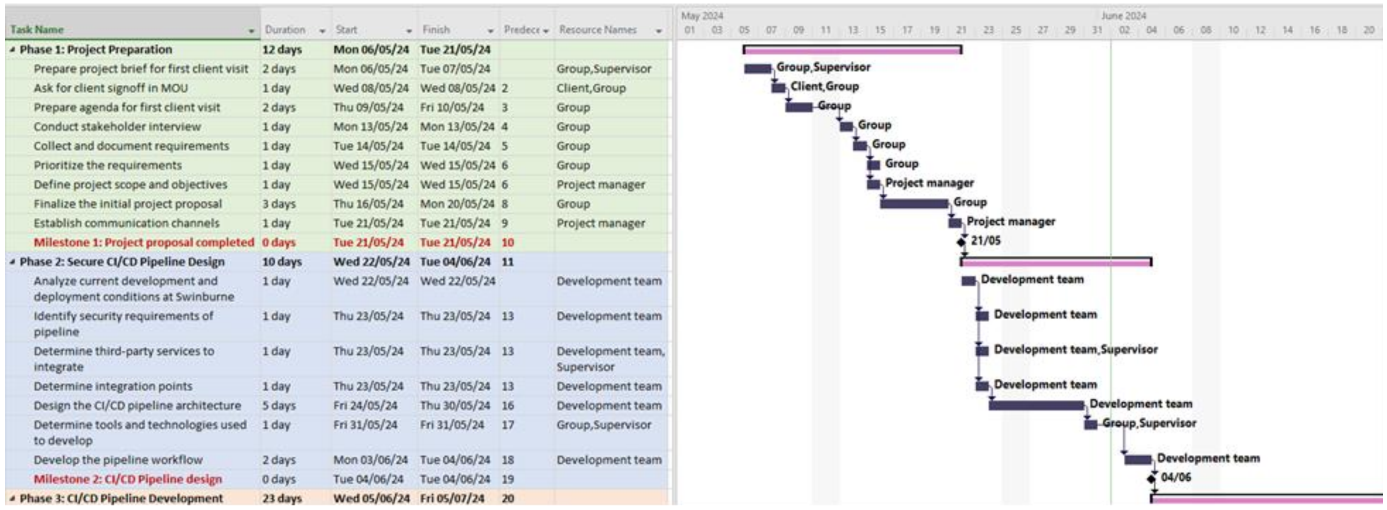
Signed: [illegible] Digitally signed by Nguyen Linh Dan Location: Swinburne University of Technology Date: 2024.06.02 17:51:14+07'00' Date: 02/06/2024

(Ms. Nguyen Linh Dan, Project Manager)

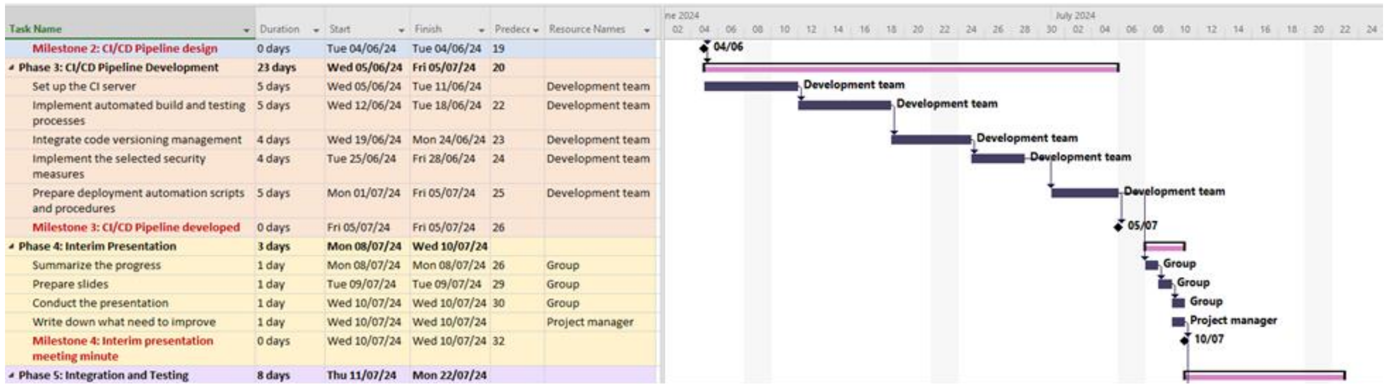
Signed: ____ Date: ______________

(Mr. Aiden Nguyen, Project Supervisor)

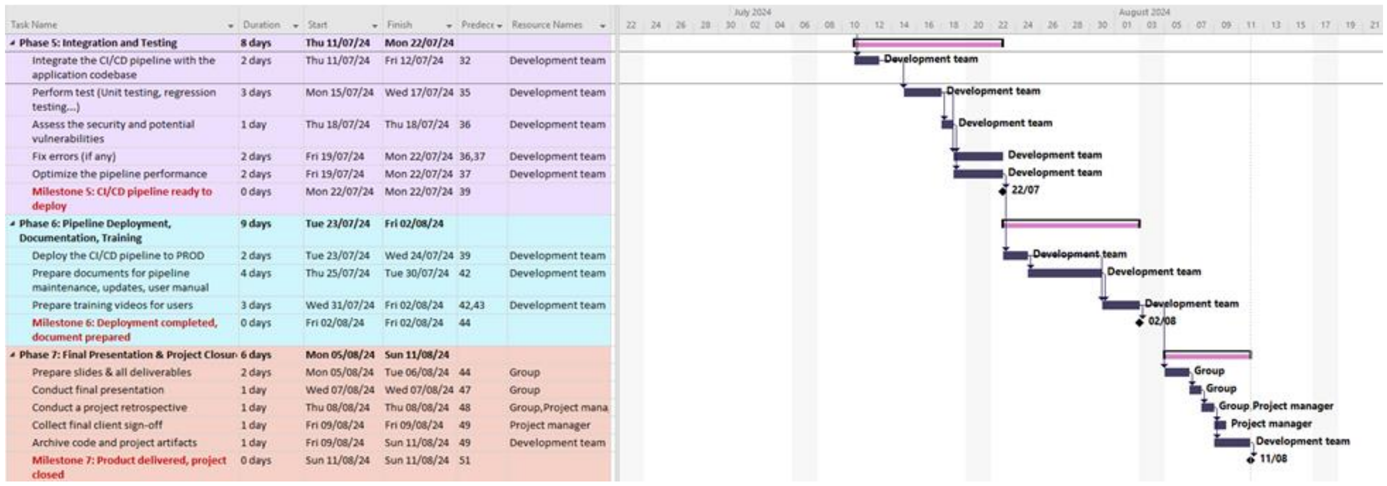
APPENDIX A – GANTT CHART



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Phase 1: Project Preparation	**12 days**	**Mon 06/05/24**	**Tue 21/05/24**		
Prepare project brief for first client visit	2 days	Mon 06/05/24	Tue 07/05/24		Group,Supervisor
Ask for client signoff in MOU	1 day	Wed 08/05/24	Wed 08/05/24	2	Client,Group
Prepare agenda for first client visit	2 days	Thu 09/05/24	Fri 10/05/24	3	Group
Conduct stakeholder interview	1 day	Mon 13/05/24	Mon 13/05/24	4	Group
Collect and document requirements	1 day	Tue 14/05/24	Tue 14/05/24	5	Group
Prioritize the requirements	1 day	Wed 15/05/24	Wed 15/05/24	6	Group
Define project scope and objectives	1 day	Wed 15/05/24	Wed 15/05/24	6	Project manager
Finalize the initial project proposal	3 days	Thu 16/05/24	Mon 20/05/24	8	Group
Establish communication channels	1 day	Tue 21/05/24	Tue 21/05/24	9	Project manager
Milestone 1: Project proposal completed	0 days	Tue 21/05/24	Tue 21/05/24	10	
Phase 2: Secure CI/CD Pipeline Design	**10 days**	**Wed 22/05/24**	**Tue 04/06/24**	**11**	
Analyze current development and deployment conditions at Swinburne	1 day	Wed 22/05/24	Wed 22/05/24		Development team
Identify security requirements of pipeline	1 day	Thu 23/05/24	Thu 23/05/24	13	Development team
Determine third-party services to integrate	1 day	Thu 23/05/24	Thu 23/05/24	13	Development team, Supervisor
Determine integration points	1 day	Thu 23/05/24	Thu 23/05/24	13	Development team
Design the CI/CD pipeline architecture	5 days	Fri 24/05/24	Thu 30/05/24	16	Development team
Determine tools and technologies used to develop	1 day	Fri 31/05/24	Fri 31/05/24	17	Group,Supervisor
Develop the pipeline workflow	2 days	Mon 03/06/24	Tue 04/06/24	18	Development team
Milestone 2: CI/CD Pipeline design	0 days	Tue 04/06/24	Tue 04/06/24	19	
Phase 3: CI/CD Pipeline Development	**23 days**	**Wed 05/06/24**	**Fri 05/07/24**	**20**	



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Milestone 2: CI/CD Pipeline design	0 days	Tue 04/06/24	Tue 04/06/24	19	
Phase 3: CI/CD Pipeline Development	**23 days**	**Wed 05/06/24**	**Fri 05/07/24**	**20**	
Set up the CI server	5 days	Wed 05/06/24	Tue 11/06/24		Development team
Implement automated build and testing processes	5 days	Wed 12/06/24	Tue 18/06/24	22	Development team
Integrate code versioning management	4 days	Wed 19/06/24	Mon 24/06/24	23	Development team
Implement the selected security measures	4 days	Tue 25/06/24	Fri 28/06/24	24	Development team
Prepare deployment automation scripts and procedures	5 days	Mon 01/07/24	Fri 05/07/24	25	Development team
Milestone 3: CI/CD Pipeline developed	0 days	Fri 05/07/24	Fri 05/07/24	26	
Phase 4: Interim Presentation	**3 days**	**Mon 08/07/24**	**Wed 10/07/24**		
Summarize the progress	1 day	Mon 08/07/24	Mon 08/07/24	26	Group
Prepare slides	1 day	Tue 09/07/24	Tue 09/07/24	29	Group
Conduct the presentation	1 day	Wed 10/07/24	Wed 10/07/24	30	Group
Write down what need to improve	1 day	Wed 10/07/24	Wed 10/07/24		Project manager
Milestone 4: Interim presentation meeting minute	0 days	Wed 10/07/24	Wed 10/07/24	32	
Phase 5: Integration and Testing	**8 days**	**Thu 11/07/24**	**Mon 22/07/24**		



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Phase 5: Integration and Testing	**8 days**	**Thu 11/07/24**	**Mon 22/07/24**		
Integrate the CI/CD pipeline with the application codebase	2 days	Thu 11/07/24	Fri 12/07/24	32	Development team
Perform test (Unit testing, regression testing...)	3 days	Mon 15/07/24	Wed 17/07/24	35	Development team
Assess the security and potential vulnerabilities	1 day	Thu 18/07/24	Thu 18/07/24	36	Development team
Fix errors (if any)	2 days	Fri 19/07/24	Mon 22/07/24	36,37	Development team
Optimize the pipeline performance	2 days	Fri 19/07/24	Mon 22/07/24	37	Development team
Milestone 5: CI/CD pipeline ready to deploy	0 days	Mon 22/07/24	Mon 22/07/24	39	
Phase 6: Pipeline Deployment, Documentation, Training	**9 days**	**Tue 23/07/24**	**Fri 02/08/24**		
Deploy the CI/CD pipeline to PROD	2 days	Tue 23/07/24	Wed 24/07/24	39	Development team
Prepare documents for pipeline maintenance, updates, user manual	4 days	Thu 25/07/24	Tue 30/07/24	42	Development team
Prepare training videos for users	3 days	Wed 31/07/24	Fri 02/08/24	42,43	Development team
Milestone 6: Deployment completed, document prepared	0 days	Fri 02/08/24	Fri 02/08/24	44	
Phase 7: Final Presentation & Project Closur	**6 days**	**Mon 05/08/24**	**Sun 11/08/24**		
Prepare slides & all deliverables	2 days	Mon 05/08/24	Tue 06/08/24	44	Group
Conduct final presentation	1 day	Wed 07/08/24	Wed 07/08/24	47	Group
Conduct a project retrospective	1 day	Thu 08/08/24	Thu 08/08/24	48	Group,Project mana
Collect final client sign-off	1 day	Fri 09/08/24	Fri 09/08/24	49	Project manager
Archive code and project artifacts	1 day	Fri 09/08/24	Sun 11/08/24	49	Development team
Milestone 7: Product delivered, project closed	0 days	Sun 11/08/24	Sun 11/08/24	51	

Project Management file (.mpp) and Gantt Chart exported .pdf file: Link