
Ticket 7 SDD

CS5213 Group B

Mar 09, 2024

CONTENTS

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Overview | 1 |
| 1.2 | Stakeholders | 1 |
| 1.3 | Goals and Context | 2 |
| 2 | Solution | 4 |
| 2.1 | System Architecture | 4 |
| 2.2 | UI | 5 |
| 2.3 | Reduced Scope | 8 |
| 3 | Sprint Structure | 9 |
| 3.1 | Sprint Duration | 9 |
| 3.2 | Monday: Sprint Planning Meeting | 9 |
| 3.3 | Daily: Standup Calls | 9 |
| 3.4 | Thursday: Code Cutoff | 10 |
| 3.5 | Friday: Sprint Review Session | 10 |
| 3.6 | Deployments | 10 |
| 3.7 | Release Planning | 10 |
| 4 | Milestone and Timeline: | 11 |
| 4.1 | Milestone M1: Project Setup | 11 |
| 4.2 | Milestone M2: Production Environment Setup | 12 |
| 4.3 | Milestone M3: Test Environment Setup Report | 14 |
| 4.4 | Milestone M4: ORM Objects Definition Report | 15 |
| 4.5 | Milestone M5: Automated CI/CD Deployment Report | 17 |
| 4.6 | Milestone M6: Login and Dashboard Accessibility Report | 18 |
| 4.7 | Milestone M7: Team Formation System Report | 20 |
| 4.8 | Milestone M8: Calendar Integration Report | 21 |
| 4.9 | Milestone M9: Submission Page Report | 23 |
| 4.10 | Milestone M10: Communication System Integration Report | 25 |
| 4.11 | Milestone M11: Review Page with Grade Integration Report | 26 |
| 5 | Sprint Timelines | 29 |
| 5.1 | Completion | 30 |
| 5.2 | Work Distribution | 30 |

INTRODUCTION

1.1 Overview

The Capstone Management System (CMS) stands as a sophisticated web-based platform meticulously developed to optimize the management of capstone projects within educational institutions. Designed to cater to the diverse needs of students, faculty, and administrators, the CMS seamlessly integrates fundamental features like user authentication, project repositories, communication tools, and analytics. The overarching objective is to simplify and enhance the various facets of capstone project management, providing a unified and accessible solution that promotes collaboration and operational efficiency.

At its core, the CMS focuses on user engagement, seeking to empower students with tools for project success while facilitating faculty and administration through comprehensive monitoring and evaluation features. The system goes beyond mere project management, aiming to contribute to the educational journey by fostering a more streamlined and efficient process. With an emphasis on simplicity and effectiveness, the CMS envisions transforming the traditional approach to capstone project management in educational settings.

Through its user-centric design, the CMS addresses the unique requirements of each stakeholder involved in capstone projects. The integration of advanced reporting and analytics capabilities further enables data-driven decision-making, providing valuable insights for refining project guidelines and support structures. In summary, the Capstone Management System represents a comprehensive and accessible solution, poised to redefine the educational experience by simplifying project management and promoting collaboration within the capstone project landscape.

1.2 Stakeholders

The stakeholders in the Capstone Management System development:

Users (Students, Faculty, Administrators):

- **Students:** Primary end-users relying on the CMS for managing capstone projects. They prioritize user-friendly interfaces, secure project data, and streamlined project management processes.
- **Faculty:** Stakeholders utilizing the CMS for project oversight, collaboration, and assessment. They seek features that enhance communication, feedback, and evaluation of student projects, emphasizing functionality and ease of use.
- **Administrators:** Overseeing departmental goals, administrators expect the CMS to align with organizational objectives. They prioritize scalability, data security, and alignment with the department's strategic vision in the CMS.

Investors (Department Decision-Makers):

- **Department Decision-Makers:** As investors, decision-makers within the department rely on the Software Development Document (SDD) to understand development strategies, project timelines, and CMS alignment with

departmental goals. The SDD assists in making informed decisions regarding resource allocation and investment in CMS development.

Maintainers (Developers):

- **Developers:** Serving as maintainers, developers are responsible for ongoing development, optimization, and maintenance of the CMS. The SDD is a crucial guide for developers, offering insights into the testing approach, expectations, and specific areas of focus for development. It ensures that the software aligns with intended testing scenarios, contributing to the system's long-term reliability and functionality.

The stakeholders in the CMS development process, including end-users (students, faculty, administrators), department decision-makers, and developers, collectively shape the trajectory of the Capstone Management System. Their collaboration and engagement are pivotal in ensuring that the CMS becomes a seamlessly integrated solution, meeting the distinct needs of each stakeholder group. By actively participating in the development journey, these stakeholders contribute to the creation of a user-friendly, functional, and strategically aligned platform that enhances the efficiency of capstone project management within educational institutions. Clear communication and collaboration among these stakeholders will continue to be fundamental as the CMS progresses, ensuring its successful implementation and sustained impact on the educational landscape.

1.3 Goals and Context

Enhanced Capstone Project Management:

- **Goal:** To develop the Capstone Management System (CMS) to significantly improve the efficiency and effectiveness of capstone project management for students, faculty, and administrators.
- **Context:** The primary aim of the CMS is to provide a platform that simplifies project-related processes, fostering collaboration, and contributing to a streamlined educational experience.

User-Friendly Interface:

- **Goal:** To design an intuitive and user-friendly interface for the CMS that ensures ease of navigation and accessibility for all stakeholders.
- **Context:** A user-friendly interface enhances the overall user experience, encouraging active engagement and participation in the capstone project management process.

Scalability for Future Growth:

- **Goal:** To build a scalable CMS architecture that can accommodate future growth, additional features, and evolving educational requirements.
- **Context:** Anticipating the need for expansion and adaptation, the CMS aims to provide a robust foundation that can scale seamlessly with the growth of capstone projects and user needs.

Robust Security Measures:

- **Goal:** To implement stringent security measures within the CMS to safeguard sensitive project data, ensuring compliance with industry standards and regulations.
- **Context:** Security is a top priority to protect the confidentiality and integrity of capstone project information, fostering trust among users and meeting regulatory requirements.

Effective Collaboration Tools:

- **Goal:** To integrate effective collaboration tools within the CMS, facilitating seamless communication, feedback, and interaction among students, faculty, and administrators.
- **Context:** The CMS aims to enhance collaboration, ensuring that project teams can communicate efficiently and receive timely feedback, fostering a supportive learning environment.

Data-Driven Decision-Making:

- Goal: To incorporate robust reporting and analytics features into the CMS to enable data-driven decision-making for administrators and faculty.
- Context: Analytics tools provide valuable insights into project outcomes, student performance, and overall system effectiveness, empowering educational institutions to make informed decisions.

Continuous Improvement and Adaptability:

- Goal: To design the CMS with a flexible architecture to allow for continuous improvement, updates, and adaptations based on user feedback and evolving educational needs.
- Context: The CMS seeks to be a dynamic platform that evolves over time, ensuring its relevance and effectiveness in meeting the changing requirements of capstone project management.

These goals collectively shape the vision of the Capstone Management System, aiming to deliver a secure, user-friendly, and adaptable platform that enhances capstone project management within educational institutions.

SOLUTION

The proposed solution will incorporate approaches and technologies that are appropriate for an application with a small number of users. A small number of users in this context constitutes less than one hundred thousand. Most modern server appliances with a sufficiently fast ethernet connection can support thousands of concurrent web application users before the system's resources become strained. Designing the application to be deployed on a standard web server system will help ensure that the deployment of the application is straight forward and compatible with most standard hosting options. Standard hosting options include: a server appliance owned by the customer, a virtual private server (VPS) hosted in the cloud (Azure, AWS, Google Cloud, etc.), or even a container deployed into Kubernetes or other container orchestration environments.

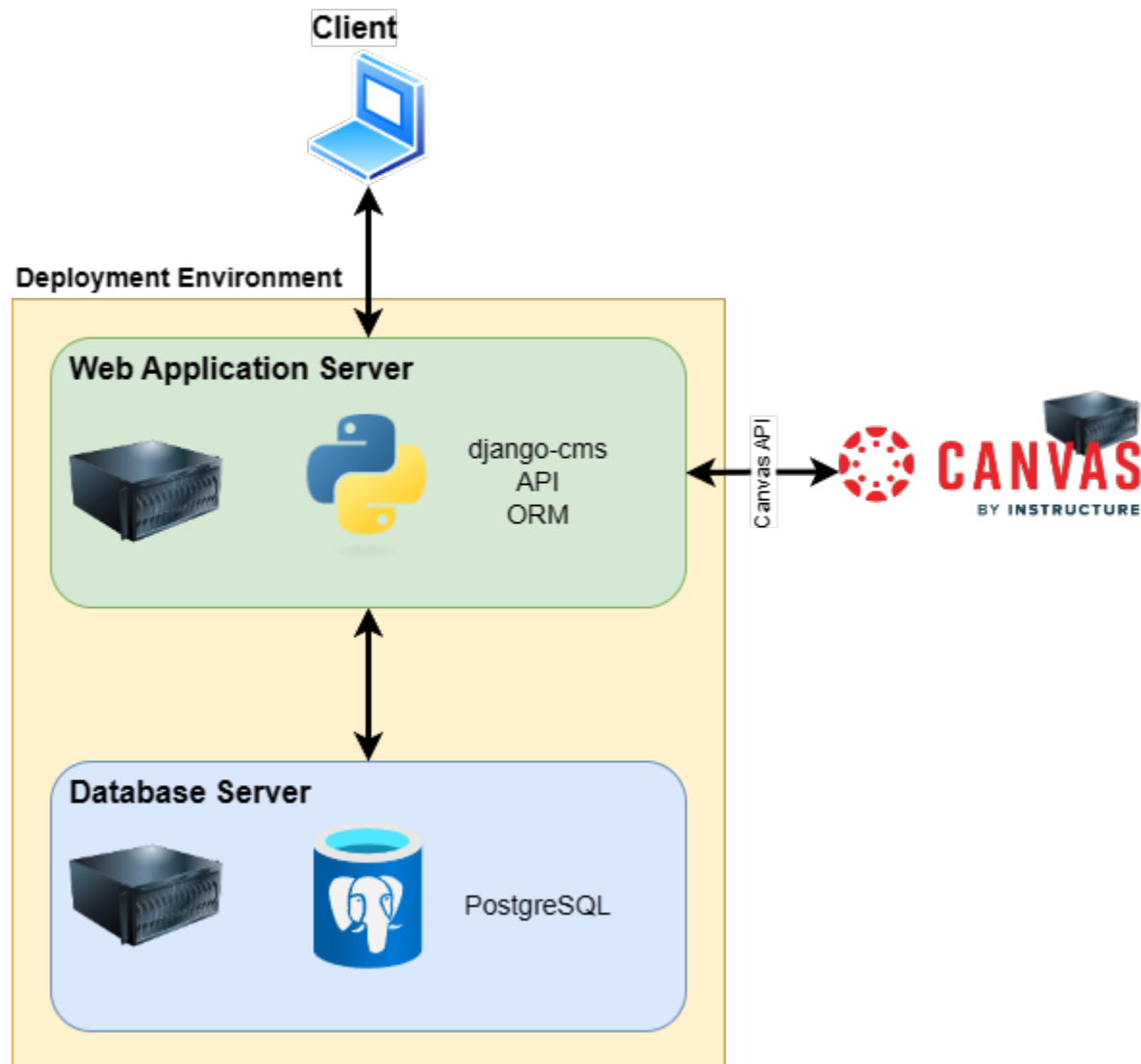
2.1 System Architecture

The linux web server is a tried and true web application deployment environment. Much hype surrounds designing applications with microservice architectures. These cloud native designs are meant for scalability, but there is no point in designing an application for scalability -incurring the extra complexity and deployment environment restrictions- when there are no guarantees that the application will ever need to scale in the first place.

The proposed application will be elegant in the sense that deployment will consist of archiving a certain state of the git repository by bundling the application code (using django deployment mechanisms), copying the archive to a web server, applying database migrations to the database, and restarting the web application service. The architecture of the deployment environment will be structured with the following components:

- Web Application Server
 - This server will host the Capstone Management System django application. It can be any sufficiently powerful modern server or VPS appliance.
- PostgreSQL Database Server
 - This server will host the database for the CMS application. It can be the same server appliance as the web application server, but may also be separated for to allow other instances of the web application to access it.
- Canvas
 - The deployment environment must be able to access `canvas.ou.edu` in order to make canvas API integration possible.


Architecture diagram




2.2 UI

Django CMS is simply Django, so the content of the site can be completely customized by the development team. The basic Django CMS deployment looks like the following screenshots:

django **CMS** [Log in](#)

 DEMO

BlogKiller Features ▾About us ▾

Welcome to the django CMS demo

This is your playground! Feel **free** to **play** around on the **website**, move elements, change content, add pages etc.

In order to edit this page press the "Log in" button in the top left corner.

Login email and password are the same:

`demo@django-cms.org`

`demo@django-cms.org`

The demo will be expire after 60 minutes

The source code of the demo is available on [Github](#)

If have any questions, please reach out to: info@django-cms.org

Please note: This demo can only give you a small impression about the features and capabilities of django CMS. If you want to learn more about django CMS, please contact us. We will be happy to schedule an individual demo with you.

Two columns example


Those columns are responsive and transform naturally in full width on lower resolutions.

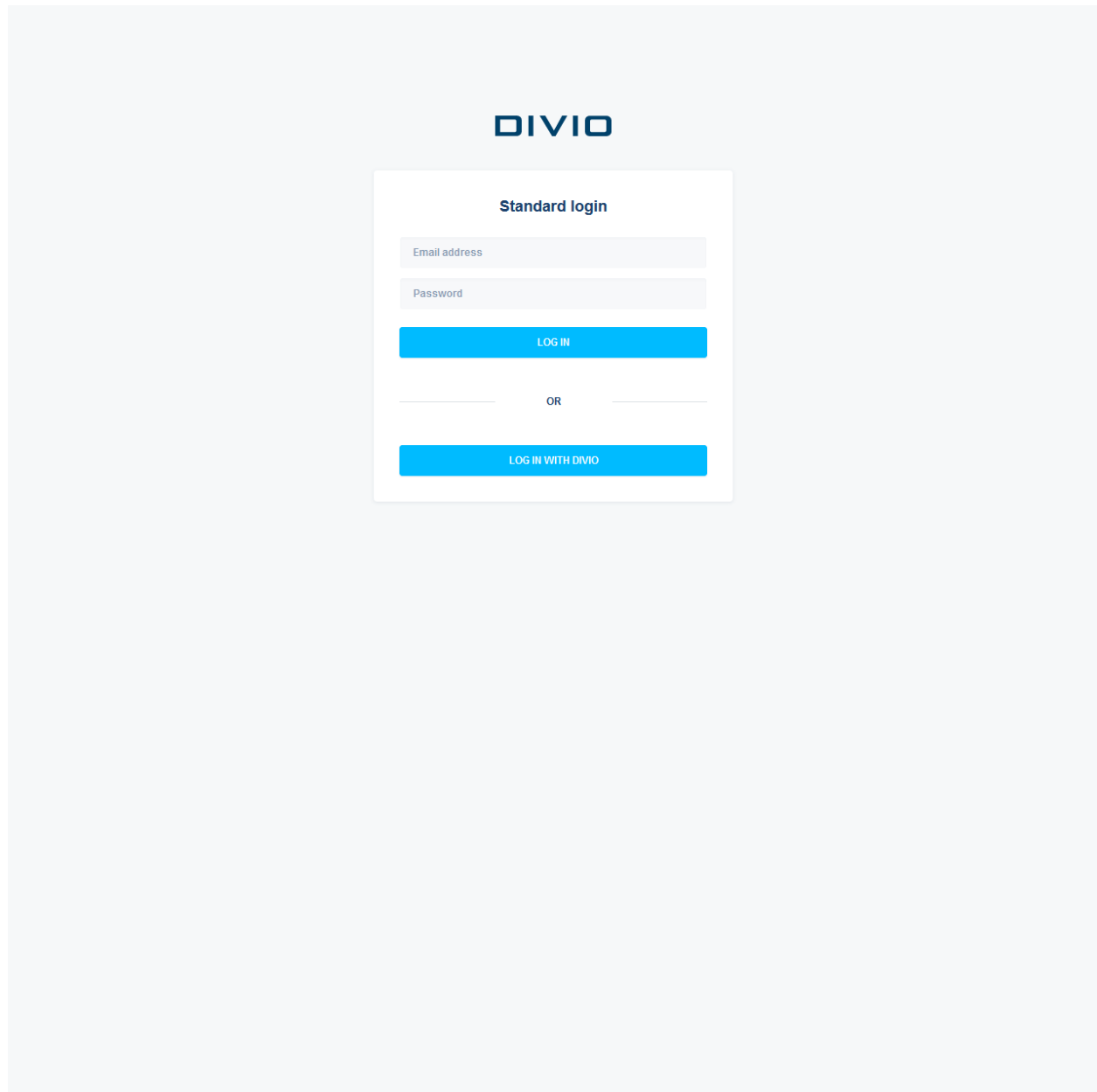
In order to change the column structure, go to *Page* and click on *Templates*. You can choose from a variety of different page structures.

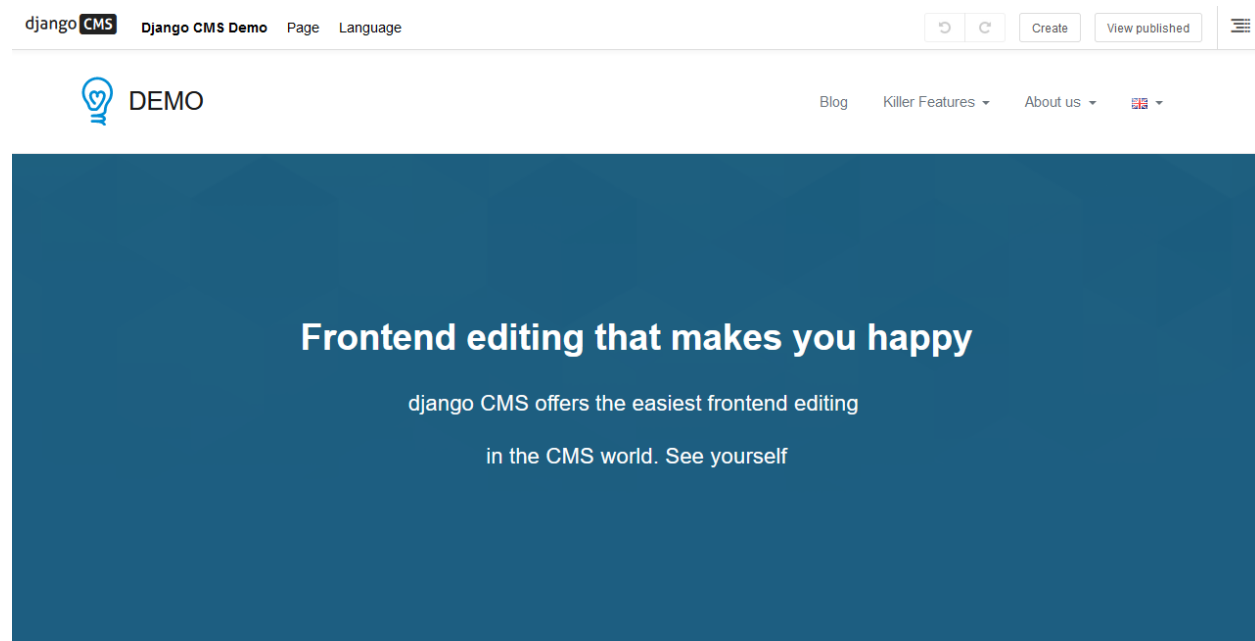
Below you can also see an example of a spoiler that contains a zoomable image:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Gallery example



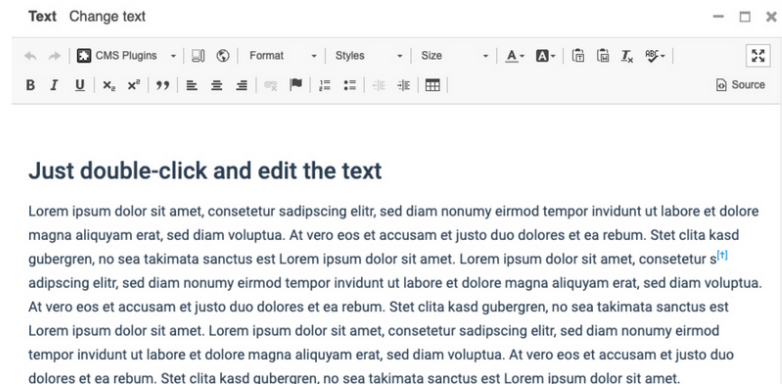




Frontend editing done the right way

Rich Content Editor

Simple and easy editing with a Rich Content Editor that allows to integrate Plugins. Everything an editor's heart desires.



Frontend editing done the right way

Rich Content Editor

Just double-click and edit the text

CMS Plugins: Reference popup with mouseover effect

CMS Plugins: Links & Buttons

2.3 Reduced Scope

Due to the constrained development timeline of 4 1-week sprints, the scope of the development of the proposed solution for the Capstone Management System has been reduced to 11 milestones. These 11 milestones cover the basic setup, deployment, and critical automations and support around the application as well as the Login, Team Formation, Calendar, Submission, Communications, and Review features of the CMS application.

An additional justification for the reduction in scope is that the development teams have not worked together to develop an application previously. Many of the development team members have not developed an application of with the scope and size of the original system requirements specification. In order to ensure the success of the project and maintain a high level of quality for the application, it is best that the scope be limited to the proposed milestones

SPRINT STRUCTURE

The Sprint Structure outlined represents a structured and systematic approach to project management, particularly within the Agile framework. Here's a detailed breakdown and elaboration of each component of the Sprint Structure:

3.1 Sprint Duration

- **Timeframe:** The sprint lasts for 5 working days, from Monday to Friday. This concise period promotes focus and efficiency, ensuring the team remains agile and can quickly adapt to changes.

3.2 Monday: Sprint Planning Meeting

- **Participants:** Scrum masters and stakeholders.
- **Duration:** 1 to 2 hours.
- **Platform:** Conducted via Zoom meetings.
- **Activities:**
 - Milestones are reviewed, and tasks are extracted and assigned to team members.
 - The team collaborates to identify the objectives for the sprint, ensuring everyone is aligned and understands their responsibilities.
 - This meeting sets the tone and direction for the week, ensuring all team members are focused and understand their goals.

3.3 Daily: Standup Calls

- **Duration:** 15 minutes.
- **Purpose:**
 - A quick, daily check-in allows team members to report progress, share updates, and identify any roadblocks.
 - The scrum master facilitates the discussion, ensuring it remains focused and productive.
 - Any identified blockers or issues are addressed, with strategies or individuals assigned to resolve them, ensuring momentum is maintained.

3.4 Thursday: Code Cutoff

- **Policy:** No more merge/pull requests are allowed; the codebase is frozen.
- **Purpose:**
 - This cutoff ensures stability and allows for a thorough review and testing of the existing code.
 - It helps in mitigating risks associated with last-minute changes, ensuring the quality and reliability of the product.

3.5 Friday: Sprint Review Session

- **Duration:** 1 hour.
- **Platform:** Conducted via Zoom meetings.
- **Participants:** Scrum masters and stakeholders.
- **Activities:**
 - The team presents a demo of the work accomplished, allowing stakeholders to verify that the outputs meet the predefined requirements.
 - This session provides a platform for feedback, enabling continuous improvement.
 - The scrum master evaluates whether the sprint's milestones were achieved, ensuring accountability and progress tracking.

3.6 Deployments

- **Frequency:** As required within the sprint.
- **Context:** Deployments to test and development environments are executed whenever necessary, ensuring continuous integration and allowing for ongoing testing and validation.

3.7 Release Planning

- **Timing:** Planned on Fridays after successful demos.
- **Purpose:**
 - Once it's confirmed that all milestones are accomplished and stakeholders are satisfied with the outcomes, the release is planned.
 - This ensures that the product or feature is ready for broader testing or deployment, marking the culmination of the sprint's efforts.

By adhering to this structured approach, teams can maintain focus, adapt quickly to changes, and ensure a high level of productivity and quality in their deliverables. The consistent checkpoints and reviews facilitate transparency, enable timely feedback, and ensure that the team is always aligned with the project's goals and deadlines.

MILESTONE AND TIMELINE:

4.1 Milestone M1: Project Setup

The Milestone M1 for the Capstone Management System, aimed at establishing the foundational development infrastructure. The setup of a Django-based environment marks the first concrete step toward developing a functional and robust system. Both development and production environments have been configured to support various development stages, ensuring a smooth transition from coding to deployment. Milestone M1's completion provides the Capstone Management System project with a robust and scalable foundation for all subsequent development activities. The environments are configured to uphold best practices in development, security, and deployment, ensuring that the project is built on a reliable and efficient infrastructure.

- **Purpose:** To set up the initial development infrastructure of the Capstone Management System using Django and to configure separate environments for development and production purposes.
- **Scope:** Establish the basic project structure and ensure that both development and production environments are operational and properly segregated.

4.1.1 Process

The setup involved the following key steps:

1. Django Project Initialization

- Execution of `django-admin startproject` to generate the base project structure.
- Configuration of initial project settings, including database connections, security configurations, and static asset management.
- Creation of a `requirements.txt` file to document project dependencies.

2. Development Environment Configuration

- Installation of Python, Django, and other dependencies on development machines.
- Setup of a virtual environment for each developer to maintain project-specific dependencies separately from the global environment.
- Establishment of a version control system with an initialized repository and properly configured `.gitignore` file.

3. Production Environment Configuration

- Selection of hosting services and configuration of production-grade web and database servers.
- Implementation of security measures such as environment variables for managing sensitive configurations.
- Planning of a CI/CD pipeline for automated testing and deployment to the production server.

4.1.2 Deliverables

1. Django Project Created

- A new Django project with a default landing page accessible on local development servers.

2. Development Environment Established

- Fully operational development environments on all developers' machines, complete with IDE setup and access to the version control repository.

3. Production Environment Established

- A live, initial version of the project deployed to the production environment, ready for future updates and feature deployments.

4.1.3 Challenges and Solutions

During the setup process, several challenges were encountered:

- **Compatibility Issues:** Differing operating system versions among development team members required careful coordination. Virtual environments and Docker containers were used to standardize development settings.
- **Security Concerns:** To prevent sensitive information from being stored in version control, environment variables and secret management tools were introduced.

4.2 Milestone M2: Production Environment Setup

4.2.1 Executive Summary

The Milestone M2, focusing on the setup of the production environment for the Capstone Management System. The objective was to establish a secure and reliable production environment using cloud services. This milestone is a key step in ensuring that the application's deployment is robust and capable of handling real-world usage.

- **Purpose:** To configure and deploy the production environment of the Capstone Management System on a cloud platform.
- **Scope:** The scope of this milestone includes selecting an appropriate cloud service provider, setting up the server infrastructure, and ensuring the system's readiness for deployment and live operation.

4.2.2 Production Environment Setup

Cloud/Server Configuration

1. Cloud Service Provider Selection

- Evaluation and selection of a cloud service provider based on the system requirements, cost, scalability, and reliability. Providers considered include AWS, GCP, and Azure.

2. Server Setup

- Provisioning of cloud server instances (virtual machines) to host the web application and associated services.
- Configuration of server specifications to match the expected load and performance requirements.

3. Server Configuration

- Manual installation of the operating system and necessary packages including web server software (e.g., Nginx, Apache), a database server (e.g., PostgreSQL, MySQL), and other dependencies.
- Configuration of the network settings, including firewalls and security groups, to secure the server from unauthorized access.

4. Application Deployment

- Deployment of the Capstone Management System codebase to the server.
- Setup of the web server to serve the application and manage traffic.

5. Database Configuration

- Installation and configuration of the production database system.
- Implementation of security measures such as strong passwords, encrypted connections, and restricted access to the database.

6. Security Measures

- Implementation of security best practices, including SSL/TLS encryption for secure HTTP connections.
- Regular updates and security patches applied to the server and software to protect against known vulnerabilities.

7. Backup and Recovery Strategy

- Establishment of regular backup procedures for the server and database.
- Creation of a disaster recovery plan to minimize downtime and data loss in case of a server failure.

4.2.3 Deliverables

1. Production Server Ready

- A fully configured and operational server ready to host the Capstone Management System in a production capacity.

2. Application Deployed

- The latest stable version of the application deployed to the production server and accessible via a secure connection.

3. Backup and Recovery

- Backup systems tested and confirmed, ensuring data integrity and system resilience.

4.2.4 Challenges and Solutions

- **Resource Estimation:** Determining the correct server specifications required careful consideration of the anticipated user load. Initial estimations were made based on projected usage, with the understanding that the system's scalability allows for adjustments as actual usage patterns emerge.
- **Security Configuration:** Ensuring the security of the production environment was paramount. A combination of network security measures and regular security audits was employed to maintain a high standard of protection.

With Milestone M2 completed, the Capstone Management System now has a robust production environment in place. The system is primed for live operation, with measures taken to secure, backup, and reliably host the application for end-users.

4.3 Milestone M3: Test Environment Setup Report

4.3.1 Executive Summary

Milestone M3 centers on establishing an automated testing environment for the Capstone Management System. This milestone's successful completion has resulted in a fully automated, cloud-based server setup for testing purposes. The automation of the test environment setup ensures consistent, repeatable configurations and efficient on-demand creation of test instances.

- **Purpose:** Automate the creation and configuration of a cloud-based test environment that mirrors the production setup to facilitate continuous integration and consistent testing procedures.
- **Scope:** The scope includes automated provisioning of server resources, configuration of services, deployment of the application, and implementation of a testing suite in a cloud environment.

4.3.2 Test Environment Setup

Automated Cloud/Server Configuration

1. Automation Tool Selection

- Selection of appropriate Infrastructure as Code (IaC) tools, such as Terraform or AWS CloudFormation, to script the cloud infrastructure setup.
- Use of configuration management tools like Ansible, Chef, or Puppet to automate server configurations and deployments.

2. Resource Provisioning

- Automated scripting of virtual machine instances with the required computing resources (CPU, memory, storage).
- Configuration of auto-scaling to handle variable loads during different testing phases.

3. Server and Services Automation

- Automation of the operating system installations and updates.
- Automated setup of web servers, database servers, and other dependencies required for the test environment.

4. Application Deployment

- Deployment scripts to automatically fetch the latest version of the application from the version control system and deploy it to the test environment.

5. Database Configuration

- Automation of the database setup, including provisioning, configuration, and seeding with test data.

6. Test Suite Integration

- Integration with CI tools to trigger automated tests upon code commits.
- Configuration of test runners and environments to execute unit tests, integration tests, and other automated tests.

4.3.3 Deliverables

1. Automated Test Environment

- A fully automated infrastructure and server setup process for the test environment ready for deployment and testing.

2. Test Suite Deployment

- Deployment and integration of the automated test suite, with tests ready to run against new builds.

3. Continuous Integration Pipeline

- A CI pipeline that automatically triggers environment setup, application deployment, and test execution on code commit.

4.3.4 Challenges and Solutions

- **Environment Parity:** Ensuring the test environment accurately reflects the production environment. This was addressed by using the same IaC scripts for provisioning both environments, with differences managed through configuration files.
- **Automated Testing Coverage:** Developing a comprehensive automated test suite that covers all features. Continuous collaboration with the development team was necessary to write and update tests in tandem with new feature development.

The establishment of an automated test environment under Milestone M3 has provided the Capstone Management System with robust tools for continuous testing and quality assurance. This setup paves the way for a reliable development workflow, facilitating regular releases with confidence in the application's stability and performance.

4.4 Milestone M4: ORM Objects Definition Report

Milestone M4 has been achieved in the development of the Capstone Management System, focusing on the definition and creation of ORM (Object-Relational Mapping) objects. This milestone involves the translation of system requirements into a set of Python classes which interact with the database through Django's ORM, followed by the necessary database migrations to reflect these new models in the system's database schema.

- **Purpose:** To define the data models of the Capstone Management System within the Django framework using Python classes, and to carry out the associated database migrations to set up the underlying database schema.
- **Scope:** The work involved in this milestone includes the creation of Python classes that represent the system's data entities, the generation of migrations, and the application of these migrations to the database to create the required tables and relationships.

4.4.1 ORM Objects Definition

Python Class Development

1. Data Model Design

- Translation of the application's requirements into a comprehensive data model.
- Identification of entities such as Users, Projects, Teams, Submissions, and Evaluations, and their respective attributes and relationships.

2. Python Class Creation

- Creation of Python classes that extend Django's `models.Model` class, with fields and behaviors of the data stored in the system.
- Use of Django field types such as `CharField`, `IntegerField`, `ForeignKey`, etc., to define the properties of each model.

3. Relationship Mapping

- Definition of relationships between models using ORM relationships such as `ForeignKey`, `ManyToManyField`, and `OneToOneField`.
- Ensuring referential integrity and the efficient retrieval of related data through Django's query syntax.

Database Migration Generation

1. Migration Scripts

- Use of Django's `makemigrations` command to automatically generate migration scripts based on changes to the models.
- Review and, if necessary, manual editing of migration scripts to fine-tune database schema changes.

2. Database Schema Creation

- Application of migrations to the development database using Django's `migrate` command to create the necessary database tables and relationships.
- Verification of the resulting database schema to ensure it accurately represents the data models.

4.4.2 Deliverables

1. ORM Classes Defined

- A complete set of Python classes representing the application's data model, committed to the version control system.

2. Database Migrations Completed

- Migration scripts generated, applied, and verified, resulting in an updated database schema ready for use by the application.

4.4.3 Challenges and Solutions

- **Data Model Complexity:** The complexity of the system's data relationships required careful analysis and design. The use of ER (Entity-Relationship) diagrams and collaborative design sessions ensured a comprehensive data model was achieved.
- **Migration Management:** Handling database migrations, especially when altering existing tables with data. Development included versioning of migrations and thorough testing to ensure data integrity.

The completion of Milestone M4 marks a significant step in the backend development of the Capstone Management System. The defined ORM objects form the framework upon which database interactions are built and play a crucial role in the system's operation and data management.

4.5 Milestone M5: Automated CI/CD Deployment Report

This report provides a detailed account of achieving Milestone M5, which was focused on implementing an automated Continuous Integration/Continuous Deployment (CI/CD) pipeline for the Capstone Management System (CMS). With the use of GitHub Actions, the pipeline has been configured to streamline and automate the testing, building, and deployment processes. This ensures that code changes are reliably integrated and deployed to the test environment and, following approval, to the production environment.

- **Purpose:** To create a robust CI/CD pipeline that automates the process from code commit to deployment in the test environment and ultimately to the production environment upon successful reviews and tests.
- **Scope:** The scope included the setup of GitHub Actions to handle automation tasks, integration with the existing GitHub repository, and ensuring the proper deployment flow of the CMS to designated environments.

4.5.1 CI/CD Pipeline Implementation

Configuration of GitHub Actions

1. Workflow Files Creation

- Creation of `.github/workflows` YAML files to define the CI/CD process, specifying jobs, steps, triggers, and environments.
- Setup of different workflows for pull requests and deployment to handle tests and merges into the main branch.

2. Automated Testing

- Implementation of automated tests that run every time a new pull request is created or updated, ensuring that all tests must pass before the code can be merged.
- Use of service containers when necessary to simulate the production environment for accurate testing.

Automated Deployment

1. Test Environment Deployment

- Configuration of a deployment job that automatically deploys the CMS to the test environment after a successful merge into the main branch.
- Setup of environment secrets and variables to manage credentials and configuration settings securely.

2. Production Deployment

- Setup of manual triggers for deployment to the production environment after thorough testing and approval of changes in the test environment.
- Implementation of deployment strategies to minimize downtime and ensure rollback capabilities in case of deployment failures.

Continuous Integration Best Practices

1. Code Quality Checks

- Integration of code linters and formatters to maintain code quality and consistency across contributions.
- Enforcing coding standards and running security vulnerability scans on each pull request.

2. Review Apps

- Utilization of GitHub Actions to create temporary review apps for each pull request, allowing stakeholders to view and interact with changes in a live environment.

4.5.2 Deliverables

1. Automated CI/CD Pipeline

- A fully functional CI/CD pipeline using GitHub Actions that automates testing, building, and deploying the CMS.

2. Documentation

- Comprehensive documentation detailing the CI/CD process, including workflow files, environment setup, and deployment strategies.

4.5.3 Challenges and Solutions

- **Pipeline Complexity:** The complex requirements of the deployment pipeline necessitated iterative development and fine-tuning of the workflow files.
- **Secrets Management:** Ensuring sensitive information was securely managed within GitHub Actions required the use of encrypted secrets and adherence to best practices for secrets storage.

Milestone M5 has been successfully completed, resulting in a seamless and automated workflow that enhances the development process of the CMS. This automated CI/CD pipeline establishes a strong foundation for ongoing development, allowing the team to deliver features and fixes with higher efficiency and reliability.

4.6 Milestone M6: Login and Dashboard Accessibility Report

This report documents the completion of Milestone M6 for the Capstone Management System (CMS), where the focus was on developing a secure and functional login mechanism along with an accessible, user-friendly dashboard. The milestone also introduced a 4x4 login feature, an added security measure ensuring a robust authentication process.

- **Purpose:** To deliver a secure login system and an operational dashboard that serves as the main interface for users interacting with the CMS.
- **Scope:** The scope included the implementation of the login functionality, complete with a 4x4 matrix verification process, and the development of a dashboard that aggregates and displays relevant project data.

4.6.1 Login System Implementation

Security Features

1. Authentication Process

- Development of a login system using Django's built-in authentication frameworks.
- Integration of a 4x4 matrix code verification as a two-factor authentication step to enhance security measures during login.

2. 4x4 Matrix Code Verification

- Creation of a user-specific 4x4 matrix displayed at login, with users required to input correct values from specific coordinates within their unique matrix.
- The matrix coordinates are randomly generated and sent to the user's registered email address upon each login attempt to verify the user's identity.

Dashboard Development

1. User Interface

- Design and implementation of a dashboard that offers a comprehensive view of the user's activities, project status, deadlines, and other relevant information.
- Ensuring responsive design for accessibility across different devices and screen sizes.

2. Functionality

- Integration of real-time data feeds to provide users with up-to-date information.
- Customization options allowing users to personalize their dashboard view according to their preferences or roles within the CMS.

User Experience

1. Testing and Feedback

- Conducting user experience testing to ensure intuitive navigation and accessibility.
- Gathering user feedback to make iterative improvements to the dashboard design and functionality.

2. Documentation

- Providing clear documentation and tooltips within the dashboard to assist users in understanding how to use different features.

4.6.2 Deliverables

1. Operational Login System

- A fully functional login system with enhanced security via a 4x4 matrix code verification process.

2. Accessible Dashboard

- A live dashboard that provides a central point for users to interact with the CMS and access important information.

4.6.3 Challenges and Solutions

- **User Authentication Balance:** Balancing ease of use with security in the authentication process. The solution was rigorous user testing to streamline the 4x4 matrix verification for a balance of security and user-friendliness.
- **Dashboard Information Hierarchy:** Organizing the dashboard to display information efficiently. Feedback from users was crucial in designing a layout that prioritizes critical information and reduces clutter.

With the completion of Milestone M6, the CMS now boasts a secure login mechanism and a comprehensive dashboard that provides users with a central hub for managing and reviewing their projects. The additional layer of security implemented through the 4x4 matrix and the personalized user dashboard experience represent significant advancements in the system's usability and security.

4.7 Milestone M7: Team Formation System Report

This report announces the successful completion of Milestone M7 for the Capstone Management System (CMS), which encompasses the development of the Team Formation feature. This crucial functionality allows users to create groups, manage team membership, and control access to various project-related resources within the CMS.

- **Purpose:** To establish a robust Team Formation feature that enables users to form groups and manage access controls effectively, facilitating collaborative project management and organization within the CMS.
- **Scope:** The scope of this milestone included designing a system to create and manage teams, defining roles within these teams, and implementing access control mechanisms to secure project resources.

4.7.1 Team Formation System Development

Group Creation and Management

1. User Interface for Team Setup

- Development of an intuitive user interface allowing users to create new teams, invite members, and assign roles within the CMS.
- Implementation of functionality for team leaders to manage team member roles and permissions.

2. Team Database Schema

- Design of the underlying database schema to support team entities, member relationships, and associated permissions.
- Ensuring referential integrity and efficient data retrieval through proper database indexing and query optimization.

Access Control Implementation

1. Role-Based Access Controls (RBAC)

- Development of an RBAC system to define different levels of access for team members, from read-only participation to full administrative rights.
- Integration of the RBAC system with the team formation feature to ensure dynamic access control based on team structure and user roles.

2. Security Policies and Permissions

- Establishment of security policies to govern who can create teams, invite members, and access specific project components.
- Application of these policies through permission checks at various points within the CMS to prevent unauthorized access to sensitive information.

4.7.2 Deliverables

1. Team Formation Feature

- A complete team formation system with an easy-to-use interface for creating and managing teams and their members.

2. Access Control System

- An operational access control system that provides secure and role-specific access to CMS resources, based on team membership and assigned permissions.

4.7.3 Challenges and Solutions

- **Scalability of Team Structures:** Ensuring the system could accommodate teams of varying sizes and complexity. The solution was to design a flexible database schema and write scalable code that could handle different team dynamics and sizes.
- **Complexity of Access Controls:** Creating an access control system that was both secure and user-friendly. The development team achieved this by iteratively refining RBAC policies and conducting extensive user testing to ensure ease of use without compromising security.

Milestone M7's completion has significantly enhanced the CMS's collaborative capabilities. The Team Formation system now enables efficient and secure group collaboration, which is fundamental for managing capstone projects within the platform. The integration of sophisticated access controls ensures that project resources are adequately secured, maintaining the system's integrity and users' trust.

4.8 Milestone M8: Calendar Integration Report

Milestone M8 marks the successful integration of a calendar feature within the Capstone Management System (CMS), leveraging the Canvas API. This integration provides users with a comprehensive view of project timelines, deadlines, and events, facilitating better planning and time management.

- **Purpose:** To enhance the CMS with a dynamic calendar feature, integrating with the Canvas API for real-time updates and synchronization of project milestones, deadlines, and events.
- **Scope:** The development and integration of a calendar feature that allows users to visualize and manage their project timelines effectively within the CMS environment.

4.8.1 Calendar Feature Development

Canvas API Integration

1. API Configuration

- Setup and configuration of the Canvas API within the CMS, ensuring secure authentication and data exchange.
- Establishment of a communication protocol for fetching and updating calendar events from the Canvas platform.

2. Calendar Functionality

- Implementation of calendar views (monthly, weekly, daily) to display project-related events, deadlines, and milestones.
- Development of user interface components that allow users to interact with the calendar, including adding, editing, and removing events.

User Interaction and Experience

1. Event Management

- Creation of a seamless user experience for managing calendar events, including integration with project milestones and deadlines within the CMS.
- Enabling notifications and reminders for upcoming events to ensure users are well informed and prepared.

2. Synchronization Capabilities

- Implementation of real-time synchronization between the CMS calendar and the Canvas platform, ensuring that updates in one system are immediately reflected in the other.
- Development of conflict resolution mechanisms for handling discrepancies between the CMS and Canvas calendar data.

4.8.2 Deliverables

1. Integrated Calendar Feature

- A fully functional calendar feature within the CMS, powered by the Canvas API, providing users with an intuitive interface for managing their project schedules.

2. Documentation and User Guides

- Comprehensive documentation detailing the calendar feature's functionality, API integration, and user guides for navigating and utilizing the calendar within the CMS.

4.8.3 Challenges and Solutions

- **API Limitations and Data Mapping:** Encountering limitations with the Canvas API that required creative data mapping and integration strategies to ensure comprehensive calendar functionality. The solution involved developing custom data processing layers to bridge gaps between the CMS data structures and the Canvas API.
- **User Interface Design:** Creating an intuitive and user-friendly calendar interface that meets the diverse needs of CMS users. This was addressed through iterative design processes and user feedback sessions to refine the UI/UX.

The integration of the Canvas API-based calendar feature into the CMS represents a significant enhancement in project management capabilities. It offers users a powerful tool for visualizing project timelines and deadlines, facilitating improved organization and productivity. The successful deployment of this feature underscores the CMS's commitment to providing comprehensive and user-friendly project management solutions.

4.9 Milestone M9: Submission Page Report

Milestone M9 marks the successful implementation of the Submission Page feature within the Capstone Management System (CMS). This key functionality allows students and team members to submit their project proposals, documents, and other required materials efficiently. The submission page is designed to streamline the process of managing project submissions, ensuring a user-friendly and effective system for all participants.

- **Purpose:** To develop a centralized submission page that facilitates the easy and organized submission of project materials by students and teams.
- **Scope:** The development covered the design and implementation of the submission interface, backend processing of submissions, and integration with the CMS's project management functionalities.

4.9.1 Submission Page Development

Interface Design

1. User Experience

- Crafting an intuitive and responsive submission page design that accommodates various types of submissions, including text entries, file uploads, and links.
- Implementation of form validation to guide users in submitting correctly formatted and complete information.

2. Accessibility Considerations

- Ensuring the submission page is accessible to all users, including those with disabilities, by following web accessibility guidelines and best practices.

Backend Integration

1. Submission Handling

- Development of backend logic to process and store submissions in the system's database securely.
- Implementation of timestamping submissions to maintain an accurate record of when materials were submitted.

2. Feedback and Status Updates

- Integration of a mechanism for instructors or system administrators to provide feedback directly on the submission page or linked to individual submissions.
- Development of a status tracking feature, allowing submitters to view the review status of their submissions in real time.

Testing and Security

1. Quality Assurance

- Comprehensive testing of the submission page for functionality, usability, and responsiveness across various devices and browsers.
- Implementation of security measures to protect against common web vulnerabilities, ensuring that user data and submission materials are securely handled.

2. User Feedback Collection

- Collection of initial user feedback to identify areas for improvement and ensure the submission process meets user needs and expectations.

4.9.2 Deliverables

1. Operational Submission Page

- A fully functional submission page, integrated into the CMS, allowing for the efficient submission of project materials.

2. Documentation

- Detailed documentation on how to use the submission page, including guidelines for different types of submissions and troubleshooting common issues.

4.9.3 Challenges and Solutions

- **Handling Diverse Submission Types:** Developing a flexible submission system capable of handling various file types and sizes. The solution involved implementing dynamic form elements and server-side processing to accommodate a wide range of submission requirements.
- **User Navigation:** Ensuring users could easily navigate the submission process. Iterative design improvements and user testing were conducted to refine the user interface and experience.

The completion of Milestone M9 significantly enhances the CMS's functionality by providing a streamlined and secure method for project submissions. The submission page is a critical component in managing the workflow of capstone projects, facilitating clear communication and efficient processing of project materials.

4.10 Milestone M10: Communication System Integration Report

Milestone M10 represents a pivotal development in the Capstone Management System (CMS), focusing on the integration of a comprehensive Communication System utilizing the Canvas API. This enhancement significantly bolsters the platform's collaborative capabilities, facilitating seamless interaction among students, faculty, and project stakeholders. The implementation of the Communication System via the Canvas API enables real-time discussions, feedback exchanges, and centralized communication channels within the CMS environment.

- **Purpose:** To integrate a robust Communication System within the CMS to support effective and efficient communication between users through the use of the Canvas API.
- **Scope:** The scope included the integration of messaging features, discussion forums, and notification systems into the CMS, leveraging the Canvas API for enhanced collaboration and user engagement.

4.10.1 Communication System Integration

Canvas API Utilization

1. API Integration

- Establishing a secure connection between the CMS and the Canvas API to access communication and collaboration features.
- Implementing OAuth2 for secure API requests, ensuring user data protection and system integrity.

2. Messaging Capabilities

- Integration of direct messaging and group messaging functionalities, allowing users to communicate privately or within project teams.

Discussion Forums

1. Forum Setup

- Creation of project-specific discussion forums within the CMS, leveraging the Canvas API to facilitate topic-based discussions among project participants.
- Implementation of threading and reply functionalities to maintain organized and coherent discussions.

2. User Engagement Features

- Introduction of features such as post upvoting, tagging, and notifications to enhance user interaction and engagement within forums.

Notifications System

1. Real-Time Notifications

- Development of a notification system to alert users about new messages, forum replies, project updates, and other relevant activities.
- Integration with the Canvas API to synchronize notifications across the CMS and Canvas platforms, ensuring users receive timely updates.

2. Customization Options

- Allowing users to customize notification settings, including the frequency and types of notifications they receive, to tailor the communication experience to individual preferences.

4.10.2 Testing and Deployment

1. Functionality Testing

- Rigorous testing of communication features for functionality, security, and user experience across various devices and platforms.
- Verification of API integration stability and performance under different load conditions.

2. User Feedback and Iteration

- Collection of user feedback on the usability and effectiveness of the communication tools.
- Implementation of iterative improvements based on feedback to enhance the overall user experience.

4.10.3 Deliverables

1. Integrated Communication System

- A fully functional communication system embedded within the CMS, powered by the Canvas API, offering messaging, discussion forums, and notifications.

2. Documentation

- Comprehensive documentation covering the use of the communication features, API integration details, and customization options.

4.10.4 Challenges and Solutions

- **API Integration Complexity:** Navigating the complexities of integrating with the Canvas API was managed through detailed documentation review and incremental testing.
- **User Adoption:** Encouraging active user participation in the new communication system required clear communication of benefits and training materials to demonstrate the system's value.

The integration of the Communication System via the Canvas API marks a significant milestone in enhancing the collaborative and interactive capabilities of the CMS. By providing a suite of communication tools directly within the platform, the CMS now facilitates a more engaged and cohesive project management experience for all users.

4.11 Milestone M11: Review Page with Grade Integration Report

Milestone M11 has been successfully achieved with the development and implementation of the Review Page feature, incorporating grade integration via the Canvas API within the Capstone Management System (CMS). This milestone enhances the system's evaluation capabilities, allowing faculty and project evaluators to provide feedback and grades directly on the CMS platform, which are then seamlessly integrated into Canvas's grading system.

- **Purpose:** To create a Review Page in the CMS that enables efficient assessment and grading of student projects, with an integrated system for synchronizing these grades with Canvas, thereby streamlining the grading process and ensuring consistency across platforms.
- **Scope:** The scope involved designing a user-friendly review interface, developing the backend logic for grade submission and retrieval, and integrating this system with the Canvas API for grade synchronization.

4.11.1 Review Page Development

Interface Design

1. User-Friendly Evaluation Interface

- Development of a clean and intuitive interface on the Review Page, enabling evaluators to easily navigate through student submissions, provide feedback, and assign grades.
- Implementation of tools within the interface for annotating submissions and highlighting areas of interest or concern.

Grade Integration with Canvas

1. Canvas API Integration

- Utilization of the Canvas API to fetch student and course information, ensuring that the grading process aligns with existing course structures and student enrolments.
- Development of functionality to push grades from the CMS directly to Canvas, allowing for real-time grade updates within both systems.

2. Security and Privacy

- Implementation of secure authentication and authorization mechanisms to protect student data and ensure that only authorized personnel can access grading information.
- Adherence to FERPA guidelines and other relevant privacy regulations to maintain the confidentiality of student grades and feedback.

4.11.2 Testing and Feedback

1. Comprehensive Testing

- Execution of thorough testing protocols to validate the functionality of the review and grading features, including integration points with the Canvas API.
- Stress testing of the system under various scenarios to ensure reliability and performance stability.

2. User Acceptance Testing

- Engagement with faculty and evaluators to conduct user acceptance testing, gathering feedback on the usability and effectiveness of the Review Page and grade integration features.

4.11.3 Deliverables

1. Operational Review Page

- A fully functional Review Page within the CMS, equipped with grading and feedback capabilities, integrated seamlessly with Canvas.

2. Documentation

- Detailed documentation outlining the process of using the Review Page, integrating grades with Canvas, and troubleshooting common issues.

4.11.4 Challenges and Solutions

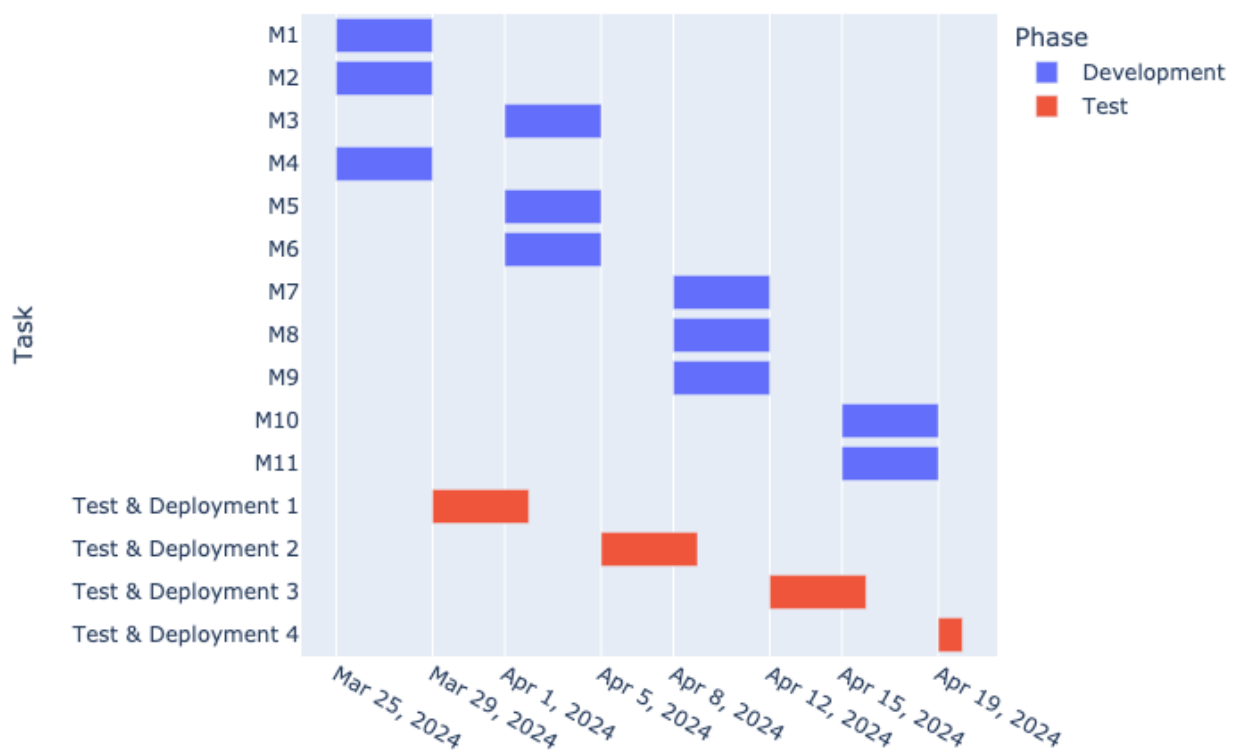
- **API Integration Complexity:** The integration with Canvas's API presented challenges in ensuring consistent and reliable data exchange. This was addressed through close collaboration with Canvas support and iterative testing of the integration.
- **User Training:** Ensuring that faculty and evaluators were comfortable using the new system required the development of comprehensive training materials and support resources.

The completion of Milestone M11 significantly enhances the CMS's functionality by providing a streamlined and integrated solution for project evaluation and grading. The Review Page with grade integration to Canvas simplifies the grading process, improves the efficiency of feedback delivery, and ensures a cohesive experience across educational platforms.

SPRINT TIMELINES

With the aforementioned milestones that have been proposed to be developed within this four-week sprint period, and with the structure and flow of each sprint discussed, the targets to be persued within each sprint must be conducted. The below Gantt chart illustrates the proposed development cycle of this project. After the code cutoff, the testing phases for each week will occur, where the primary task of integrating all the various merge requests into the deployed product will be pursued.

Propose Sprint Workflow

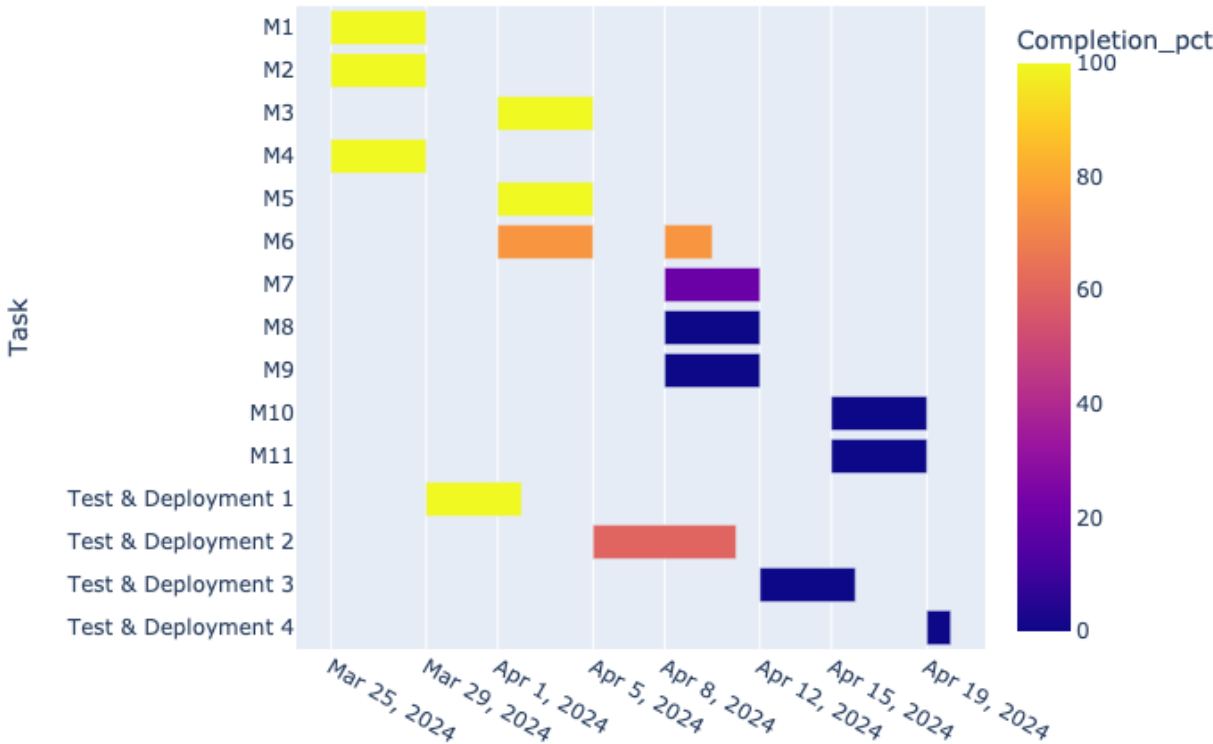


Since this project will be developed using the agile framework, tracking the completion of the various milestones will be imparative for the completion of the application. Utilizing gantt charts during sprint planning to convey the workflow of the given week will help the coherence of the team in distributing the workload properly. Here are a couple of examples:

5.1 Completion

In the development of the application, it is likely that the development teams will experience setbacks that will delay the release of a feature. Using the Gantt chart to display progress and the perscribed timeline for completion to the stakeholders and the development teams will allow for stakeholders to monitor the progress of the project, and the development teams to have new deadlines to be met for the various milestones. This example is what a potential setback at the week 3 sprint planning may look like:

Completion of Milestones vs Proposed Workflow



5.2 Work Distribution

Being able to monitor the productivity of the different groups and distributing the tasks for completion by the various groups is imparative for the success for the development of the application. Tailoring the work for a given groups strengths, and easing the load on other groups will help keep all the groups a peak productivity in the development of this project. Using the same example from above, this is another example of what a gantt chart produced after the sprint planning meeting may look like:

Distribution of work Sprint 3

