Project Plan

Team Name: Git Guardians

Team Members:

* Tyler Short
* Steven Dawe
* Jakub Sepiolo
* Owen Bradstreet
* Luke Wood

Project Name: COM619-AE1 Assessment

Project Link: [jakubsepiolo2001/COM619-AE1: COM619 Dev Ops Project](https://github.com/jakubsepiolo2001/COM619-AE1)

Stakeholder: Craig Gallen

Project Duration: 9th October 2023 to 8th December 2023

**Sprint Schedule:**

Sprint 1: 9th October 2023 - 29th October 2023

Focus: Laying the Foundations and Basics

Sprint 2: 30th October 2023 - 19th November 2023

Focus: Completing Core Functionality

Sprint 3: 20th November 2023 - 8th December 2023

Focus: Finalizing Core Functionality and Nice-to-Have Features

**Work Meetings:**

Weekly Scrum Meetings (Virtual): Virtual Scrum meetings on Mondays provide an opportunity to align the team, discuss progress, identify issues, and plan the work for the upcoming week. These meetings allow us to follow an agile project management workflow and remain responsive to changing requirements/challenges.

In-Person Scrum Meetings: In-person Scrum meetings on Thursdays offer a schedule for more in-depth discussions. This allows the team to dive into specific challenges, brainstorm solutions, and address any blockers effectively. Face-to-face meetings engage the team more and help gain a deeper understanding of the project status.

**Planning and Structure:**

Using kanban boards provides a visual representation of our project and the on-going sprint. It allows team members to quickly identify the status of tasks and helps in workload management, reducing potential burnout among team members and create a more adaptable and flexible work environment. The transparency of the current workload incorporated collaboration and increases communication.

A standardized workflow ensures consistency in how we handle tasks. It minimizes confusion and miscommunication within the team. This structure will streamline our work process and prevent issues from falling through the cracks.

**Workflow:**

Kanban Board - Issues will be created and managed using a Kanban board, which will include the following statues:

* To-Do: Tasks that need to be addressed.
* In Progress: Tasks that team members are actively working on.
* In Review: Tasks that are completed and need to be reviewed.
* Blocked: Tasks that are temporarily halted due to dependencies or issues.
* Done: Tasks that are completed and approved.

Roadmap - Issues will be tracked and time managed using a roadmap to provide a holistic overview of the projects timeline and when each feature was attempted and completed or left incomplete. The roadmap will include the following:

1. Date created
2. Issue started (in-progress)
3. Issue in review
4. Issue closed

Standardized Workflow - Each issue will be managed through the following workflow steps:

1. Testing: Issues must be tested before moving forward.
2. Review: Issues are reviewed by team members.
3. Implementation of Comments: Any review comments are addressed.
4. Second Testing: Issues must have final testing before moving forward.
5. Closed as Done.

**Communication Plan:**

Effective communication is vital to ensure that the project stays on track, team members are aligned, and stakeholders are informed about project progress. The following communication plan outlines the key elements of communication:

1. Collaboration Tools:

* Tools: Collaboration tools, such as team chat platforms, document sharing, and issue tracking systems, will be used to facilitate real-time communication and information sharing among team members.
* Justification: Collaborative tools streamline communication and information exchange, making it easier for team members to work together, even in virtual settings.

2. Stakeholder Updates:

* Regular Updates: Project stakeholders will receive regular updates on project progress, achieved milestones, and any significant changes or delays.
* Justification: Keeping stakeholders informed fosters transparency, helps manage expectations, and ensures that the project aligns with stakeholder requirements.

1. Clear Task Assignments:

* Task Management: Team members will receive clear task assignments, including responsibilities and deadlines, to minimize confusion and miscommunication.
* Justification: Clear task assignments reduce the risk of overlap, ensure accountability, and promote a structured approach to project management.

**Risk Management:**

Throughout the project we will need to be mindful of risk management. Identifying potential risks and having strategies and plans to mitigate these risks are essential to the project being a success. The following risks have been mentioned, along with potential impact of the risks and strategies for mitigation:

1. Technical Challenges:

* Impact: Complex technical challenges may arise during the development of the REST APIs, database integration, or geolocation functionality, potentially leading to project delays.
* Mitigation: To address this, the team will conduct thorough research and planning during Sprint 1. Additionally, we will assign experienced team members to critical technical tasks and be prepared to seek external expertise if needed.

1. Resource Constraints:

* Impact: Limited availability of team members or resource constraints may hinder project progress, particularly during intense development sprints.
* Mitigation: We will establish a clear project plan and allocate tasks based on team members' expertise and availability. Regular communication and task prioritization will help us navigate resource constraints.

1. GDPR Compliance Challenges:

* Impact: Failing to meet GDPR guidelines could lead to legal and privacy issues, damaging the project's reputation and user trust.
* Mitigation: To ensure compliance, the team will conduct a comprehensive GDPR review during Sprint 2, addressing any potential issues or gaps in data protection. We will also seek legal counsel if needed.

1. User Authentication and Security Risks:

* Impact: Inadequate user authentication and security measures could expose the application to data breaches and user privacy violations.
* Mitigation: The team will implement robust user authentication and security measures during Sprint 1. Additionally, external security assessments will be conducted to identify and address vulnerabilities.

1. Cloud Hosting Service Downtime:

* Impact: Downtime or service interruptions from the chosen cloud hosting provider could affect application availability.
* Mitigation: We will select a reputable cloud hosting provider with a strong track record for uptime.

1. Scope Creep:

* Impact: Expanding project scope beyond the original requirements can lead to schedule overruns and budget issues.
* Mitigation: We will closely adhere to the defined project requirements and continuously assess whether new features align with project objectives. Changes will be evaluated and approved through a change control process.

7. Communication Challenges:

* Impact: Inadequate communication among team members or with stakeholders can lead to misunderstandings, missed deadlines, and project delays.
* Mitigation: The team will maintain regular communication through virtual Scrum meetings (Mondays) and in-person meetings (Thursdays). Collaborative tools like GitHub and Microsoft Teams will be used for real-time updates and issue tracking.

8. Unforeseen Dependencies:

* Impact: The project may encounter unforeseen dependencies, which could impact timelines and cause project delays.
* Mitigation: We will maintain clear documentation and regularly update the Kanban board and roadmap to identify and address dependencies as they arise.

**Documentation Standards:**

Clear and consistent documentation is essential to ensure that team members and stakeholders have access to accurate and reliable information for the project:

* Documentation Tools: The project will utilize a documentation platform (e.g., Wiki, Markdown, or a designated static indexing system) for project documentation. This platform will serve as the central repository for all project-related documentation.
* Document Types: Documentation will encompass various types, including code documentation, user guides, and project documentation. Each type will adhere to defined standards.
* Version Control: All documentation will be version-controlled, ensuring that changes and updates are tracked, reviewed, and approved by relevant team members.
* Accessibility: Documentation will be made accessible to all team members, and guidelines will be provided for accessing, editing, and updating documentation.

**Deployment Plan:**

Effective deployment planning is crucial to ensure a smooth transition from development to production, maintaining the integrity of the application while adhering to the specified requirements. This plan addresses the key aspects of the deployment process for the project:

1. Branch Management:

* Individual Branches: Each issue will be assigned to an individual branch. This approach ensures that each task is isolated, making it easier to track progress, test, and review.
* Testing and Review: Before merging any branch into the main branch, comprehensive testing and peer review will be conducted to verify that the code is functioning correctly and adheres to the project's standards.
* Automated Testing with GitHub Actions: Commits to the main branch will trigger an automated GitHub Action that checks if the Java Maven application can be built using the .pom file. This automated testing process guarantees code quality and consistency before changes are integrated into the main branch.

2. Deployment Process:

* Incremental Updates: We will follow a strategy of incremental updates to the hosted application. Instead of infrequent major releases, this approach ensures that changes are deployed in smaller, manageable increments, making the application more agile and responsive to user needs.
* Continuous Integration and Continuous Deployment (CI/CD): CI/CD principles will be employed to automate the deployment process. This includes the integration of automated tests and deployment pipelines. CI/CD helps in maintaining consistency and reliability throughout the deployment process.

3. Deployment Environment:

* Cloud Hosting: The project specifies the use of a cloud hosting environment to host the application. As such, we will configure and deploy the application in the selected cloud environment, ensuring scalability and reliability.

4. Deployment Monitoring:

* Monitoring Environment: A monitoring environment will be created to continuously assess the operation of the hosted application. This environment will capture and record metrics and logs for the running application.
* Real-Time Monitoring: Real-time monitoring helps detect and address issues promptly, ensuring a seamless user experience and reducing the impact of potential problems.

**Testing Strategy:**

A well-defined testing strategy is crucial to ensure the reliability and functionality of the application. The testing strategy is designed to align with project requirements while optimizing the use of resources:

1. Unit Testing: Whenever possible, unit tests will be implemented during development to verify the correctness of individual code components. These tests serve as an initial quality assurance measure to detect and rectify errors early in the development process.
2. User-Centric Manual Testing: The primary testing approach for the Geo-Point Uploader will involve manual testing by users. This real-world testing will evaluate the application's usability, functionality, and performance. Users will interact with the application to ensure that geographic points can be uploaded, descriptions can be added, and photos can be displayed accurately.
3. Bug Tracking: A bug tracking system will be in place to record and manage identified issues, prioritizing them based on severity and impact.

**Continuous Improvement:**

Continuous improvement is a fundamental aspect of the Geo-Point Uploader project, ensuring that the application evolves and meets user needs:

1. User Feedback: User feedback will be actively collected, analyzed, and addressed to refine the application. Feedback on usability, functionality, and feature requests will be considered.
2. Sprint Retrospectives: At the end of each sprint, a retrospective session will be held to review the progress, identify areas for improvement, and adjust strategies for subsequent sprints.
3. Bug Resolution: Identified bugs and issues will be addressed promptly, with clear processes for resolution. This includes setting priorities and timelines for fixes.
4. Feature Enhancements: As part of continuous improvement, the team will consider feature enhancements or additions based on user feedback and evolving requirements.
5. Documentation Updates: Documentation, including user guides, will be updated to reflect any changes or improvements to the application.

**Final Presentation Plan:**

The final presentation plan outlines the structure and key components of the assessment presentation for the project:

1. Presentation Format: The presentation will be in the form of a screencast with a duration not exceeding 15 minutes. This concise format ensures that all team members have an opportunity to present their contributions within the allotted time.
2. Content: The presentation will include the following components:

* Overall Description: An introduction to the project, its objectives, and the team's approach.
* Workflow Demonstration: A brief demonstration of the project's workflow, highlighting key processes and strategies.
* DevOps Choices: An overview of the DevOps choices made during the project, including deployment and testing strategies.
* Features Completed: An overview of the features successfully completed during the project, emphasizing their alignment with project requirements.
* Features Not Completed: An honest reflection on any features that were not fully implemented or any challenges encountered.
* Reflection and Future Work: Team members' reflections on the project, lessons learned, and ideas for future enhancements.

1. Contributor Allocation: The presentation will ensure that each team member has an opportunity to contribute and share their insights, with an equal distribution of presentation time among team members.

**Sprint 1: Laying the Foundations and Basics (9th October - 29th October)**

Issues:

1. Set up GitHub repository:

* Create a GitHub repository to host the project's source code and documentation. Initialize the repository with a README.md file.
* Justification: This serves as a centralized platform for version control and collaboration, ensuring project management and code tracking.

1. Establish the project structure:

* Define the project's directory structure, including folders for frontend, backend, documentation, and testing.
* Justification: A well-organized project structure ensures team members can work effectively within a structured environment.

1. Implement automated testing using GitHub Actions (CI/CD):

* Set up automated testing with GitHub Actions to ensure code quality and reliability.
* Justification: Automated build testing improves reliability.

1. Create a monitoring environment for testing:

* Build a monitoring environment to test and ensure the operation of the hosted application, recording metrics and logs.
* Justification: Monitoring helps identify and address issues in real-time.

1. Define REST API endpoints:

* Plan and document the REST API endpoints for uploading geographic points and retrieving information.
* Justification: Clear API design prevents misunderstandings and errors during development, promoting smooth collaboration.

1. Configure HTTPS with correct certificates:

* Set up HTTPS security with valid SSL/TLS certificates to secure data in transit.
* Justification: Security is a fundamental requirement to protect user data and ensure trust.

1. Integrate Swagger for API documentation:

* Integrate Swagger to generate API documentation, making it easier for developers to understand and use the APIs.
* Justification: Well-documented APIs improve collaboration and development efficiency.

1. Choose a database and set up schema:

* Select the appropriate database technology (e.g., PostgreSQL) and design the database schema for storing geo-points and associated data.
* Justification: A robust and efficient database is essential for data storage and retrieval.

1. Implement basic user authentication:

* Create a basic user authentication system to allow users to access the application.
* Justification: User authentication is critical for securing user accounts and data.

1. Begin internationalization (i18n):

* Lay the foundation for internationalization by setting up translation tools and preparing the codebase for localization.
* Justification: Internationalization ensures the application can be used by a global audience.

1. Research and select cloud hosting provider:

* Research and choose a cloud hosting provider (e.g., AWS) for deploying the application.
* Justification: Cloud hosting offers scalability, reliability, and infrastructure management.

1. Licence the project to Apache Licensing Standards:

* Apply an Apache License to the Geo-Point Uploader project to align with open-source standards.
* Justification: Licensing to Apache standards promotes open collaboration, legal compliance, and community trust in the project.

Role Assignments:

Tyler - Documentation/Planning

Owen -

Jakub -

Luke -

Steve - Server

**Sprint 2: Completing Core Functionality (30th October - 19th November)**

Issues:

1. Implement REST API for uploading geo-points:

* Develop the REST API endpoints for users to upload geographic points with descriptions.
* Justification: This is a core functionality of the application, and it's essential for users to submit their data.

1. Develop mobile or desktop responsive web pages:

* Create responsive web pages to ensure the application is accessible and user-friendly on either mobile and desktop devices.
* Justification: Accessibility is key to reaching a wide user base.

1. Enable geolocation functionality:

* Implement geolocation functionality on mobile devices to enable users to provide accurate geographic information.
* Justification: Geolocation is a key feature for the application.

1. Set up Docker containers for the application:

* Containerize the application using Docker to enable consistent deployment and scaling.
* Justification: Containerization simplifies deployment and management.

1. Integrate GDPR guidelines:

* Incorporate GDPR guidelines to ensure the application respects user data privacy and complies with data protection regulations.
* Justification: Data protection is crucial for user trust and compliance.

1. Define roles and user sign-up functionality:

* Implement user roles and sign-up functionality to differentiate between standard users and administrators.
* Justification: Role-based access control is essential for user management.

1. Complete i18n support:

* Finalize internationalization support to make the application usable by a global audience.
* Justification: Internationalization enhances the application's reach.

Role Assignments:

Tyler -

Owen -

Jakub -

Luke -

Steve -

**Sprint 3: Finalizing Core Functionality (20th November - 8th December)**

Issues:

1. Add user and administrator roles:

* Enhance user and administrator roles, adding or refining permissions as needed.
* Justification: Role-based access control is crucial for user management and security.

1. Complete user documentation on the hosted site:

* Provide comprehensive user documentation on the hosted site to guide users on how to interact with the application.
* Justification: User documentation improves the user experience.

1. Address any pending GDPR issues:

* Ensure GDPR compliance and address any outstanding issues to safeguard user data and privacy.
* Justification: GDPR compliance is essential for user trust and legal compliance.

1. Plan for assessment presentation:

* Allocate times and areas of discussion for each member of the team.
* Justification: The assignment is mostly assessed on the end presentation. All members must contribute to the presentation for an equal amount of time.

Role Assignments:

Tyler -

Owen -

Jakub -

Luke -

Steve -

**Milestones and Deliverables:**

Milestones and deliverables are crucial aspects of project management that help track progress, ensure accountability, and align the team with project objectives. The following section outlines the key milestones and deliverables for each sprint, ensuring that they are in line with project requirements:

**Sprint 1: Laying the Foundations and Basics (9th October - 29th October)**

Milestone 1: GitHub Repository and Project Structure Established -

* Deliverable 1: GitHub repository created and initialized with a README.md file.
* Deliverable 2: Project structure defined, including directories for frontend, backend, documentation, and testing.

Milestone 2: Automated Testing and HTTPS Security -

* Deliverable 3: Automated testing set up using GitHub Actions (CI/CD) to ensure code quality.
* Deliverable 4: HTTPS security configuration with valid SSL/TLS certificates.

Milestone 3: Database and User Authentication Prepared -

* Deliverable 5: Database technology (e.g., PostgreSQL) chosen, and the database schema designed.
* Deliverable 6: Basic user authentication system created.

Final Milestone: Internationalization Framework Initiated -

* Deliverable 7: Framework for internationalization (i18n) established, including translation tools and codebase preparation.

**Sprint 2: Completing Core Functionality (30th October - 19th November)**

Milestone 1: REST API and User Interface Development -

* Deliverable 8: REST API endpoints for uploading geographic points and descriptions completed.
* Deliverable 9: Responsive web pages for mobile and desktop devices developed.

Milestone 2: Docker Containerization and GDPR Integration -

* Deliverable 10: Application containerized using Docker for consistent deployment and scaling.
* Deliverable 11: GDPR guidelines integrated into the application to ensure data privacy and compliance.

Final milestone: Role-Based Access Control and Internationalization -

* Deliverable 12: Role-based access control (user roles and sign-up functionality) implemented.
* Deliverable 13: Internationalization (i18n) support finalized for global accessibility.

**Sprint 3: Finalizing Core Functionality (20th November - 8th December)**

Milestone 1: User Documentation and GDPR Compliance -

* Deliverable 14: Comprehensive user documentation for the hosted site provided.
* Deliverable 15: Any pending GDPR compliance issues addressed and resolved.

Milestone 2: User and Administrator Roles Enhanced -

* Deliverable 16: User and administrator roles enhanced with added or refined permissions.
* Deliverable 17: Assessment presentation plan established, including time allocation for each team member.

Final milestone: Presentation and Reflection -

* Deliverable 18: Assessment presentation recorded in a screencast format, not exceeding 15 minutes.
* Deliverable 19: Presentation components include an introduction to the project, workflow demonstration, DevOps choices, features completed, features not completed, reflection, and future work. Each team member contributes equitably to the presentation.