Pollus Project Plan

Version 1.0
Date: October 1, 2024

In-Class Polling Application

Prepared by: Mark Ospina,
Jared Fuller,
Ohm Shah,
Diana Bayandina

Table of Contents

1. Introduction

1.1. Purpose and Objectives

The purpose of this document is to provide a comprehensive project plan for the development and implementation of the **“Pollus”** Web Application. This plan outlines the scope, objectives, timeline, resource allocation, and management strategies required to successfully deliver the application. It serves as a guide for the stakeholders, detailing the following:

1) project’s scheduling and resourcing,
2) quality assurance practices, and
3) risk management.

Additionally, the document ensures alignment among all team members and stakeholders by establishing clear expectations for each phase of the project.

1.2. Scope

The scope of the **“Pollus”** Web Application project encompasses the design, development, testing, and deployment of a real-time polling platform for use in academic environments. The application will allow lecturers to create, manage, and deploy multiple-choice quizzes during classroom sessions, with students responding via mobile or desktop devices.

Key features included within this scope are:

- **User Account Management**: A system for lecturers to create and manage accounts, including authentication and password recovery.
- **Poll Creation and Management**: Functionality for lecturers to design custom polls, set question timers, and launch polls in real time.

- **Student Access**: Students will be able to join polls using access codes or QR codes and submit responses via their mobile devices.
- **Real-Time Feedback**: Display of real-time quiz results, showing correct and incorrect answers, as well as a score summary at the end of the quiz.
- **Analytics and Reporting**: Detailed performance analytics for lecturers, including class averages, most common mistakes, and top performers, with options to export data in CSV or Excel format.
- **Mobile Compatibility**: The platform will be optimized for mobile devices to ensure ease of use for students during polls.

2. Project Scheduling and Resourcing

In this section there is the detailed daily plan from **October 1, 2024** to **December 3, 2024**, covering team members' tasks, requirements, and the expected output for each day. The plan follows the CMM(i) Methodology and ensures that each requirement from the Software Requirements Specification (version 2.0) is addressed. We have assigned tasks based on team member roles.

2.1. Project Schedule

October 1-7, 2024 (Week 1: Kick-off and Initial Planning)

Goal: Requirement analysis, initial system architecture, project setup.

Date	Task	Assigned to	Related Requirement
October 1, 2024	Kick-off meeting, discuss requirements and project objectives	All team members	General
October 2, 2024	Define architecture for frontend and backend	All team members	Overall SRS

October 3, 2024	Identify main functional and non-functional requirements	All team members	Requirement Traceability
October 4, 2024	Define database structure	Mark, Ohm	System requirements (Database)
October 5, 2024	Set up version control (Git) and task management (Trello or Jira)	Diana, Jared	Configuration Management Strategy
October 6, 2024	UI Mockups revision for homepage, instructor login, examinee login	All team members	Instructor Features (R-1.1, R-9.1)
October 7, 2024	Review and finalize SRS	All team members	Final SRS (traceability to all features)

October 8-14, 2024 (Week 2: UI and Backend Setup)

Goal: Finalize UI mockups, start backend development, define testing strategy.

Date	Task	Assigned to	Related Requirement
October 8, 2024	Finalize UI mockups for Instructor and Examinee login pages	Diana	R-1.1, R-9.1, R-2.1
October 9, 2024	Set up frontend skeleton (HTML/CSS for login pages)	Mark	R-2.1, R-10.1
October 10, 2024	Start backend server setup (Node.js or Python setup)	Ohm	System requirements (Backend)
October 11, 2024	Set up the initial database structure (PostgreSQL/MySQL)	Mark, Ohm	System requirements (Database)
October 12, 2024	Set up authentication module for instructor	Diana, Jared	R-2.2, Security Requirements

	login		
October 13, 2024	Define test cases for login, poll creation	All team members	Quality Assurance (Testing)
October 14, 2024	Review progress and refine tasks for the next phase	All team members	Review all requirements

October 15-21, 2024 (Week 3: Instructor Features - Poll Uploading)

Goal: Implement instructor login, poll upload functionality.

Date	Task	Assigned to	Related Requirement
October 15, 2024	Complete backend integration for instructor login	All team members	R-2.2
October 16, 2024	Work on CSV poll upload functionality (backend logic)	Ohm	R-4.1
October 17, 2024	Develop the front-end functionality for instructor poll upload	Diana, Jared	R-4.1
October 18, 2024	Test instructor login and poll upload	Mark, Ohm	R-4.1, Testing
October 19, 2024	Set up database schema for polls (questions, options, correct answers)	Mark, Ohm	Database for Polls (R-4.1, R-4.2)
October 20, 2024	Backend logic for time-per-question setup	Diana, Jared	R-4.2
October 21, 2024	Complete CSV upload testing, error handling	All team members	R-4.1, R-4.2

October 22-28, 2024 (Week 4: Instructor Control Panel and Lobby)

Goal: Complete instructor control panel, real-time lobby display.

Date	Task	Assigned to	Related Requirement
October 22, 2024	Implement "Time Per Question" field	Jared	R-4.2
October 23, 2024	Develop lobby page for real-time examinee display	Ohm	R-5.1, R-5.2
October 24, 2024	Work on backend connection for real-time lobby data	Mark, Jared	R-5.1, R-5.2
October 25, 2024	Review and test lobby functionality	All team members	Testing (Lobby)
October 26, 2024	UI refinement and feedback session for instructor control panel	Diana	Instructor Features
October 27, 2024	Work on real-time integration testing for examinees joining the poll	Diana, Jared	R-5.1, R-11.1
October 28, 2024	Final testing for lobby and control panel features	All team members	Testing

October 29 - November 4, 2024 (Week 5: Poll Session Development)

Goal: Develop and test poll session pages.

Date	Task	Assigned to	Related Requirement
October 29, 2024	Implement poll session page layout for instructor (classroom display)	Jared	R-6.1, R-6.2

October 30, 2024	Work on countdown timer and answer options display	Diana	R-6.1, R-6.3
October 31, 2024	Implement auto-progression between questions (backend)	Mark, Jared	R-6.4
November 1, 2024	Test poll session question/answer flow	Diana, Jared	R-6.1, R-6.4, Testing
November 2, 2024	Develop examinee poll session pages (answer locking)	Ohm	R-12.2
November 3, 2024	Backend validation of examinee responses	Mark, Jared	R-12.3
November 4, 2024	Test and validate poll session for both instructor and examinee	All team members	Testing (Poll Sessions)

November 5 - 11, 2024 (Week 6: Poll Results Development)

Goal: Develop and test poll results for both instructor and examinee.

Date	Task	Assigned to	Related Requirement
November 5, 2024	Develop results page for instructor (bar chart, correct answer indicator)	Ohm	R-7.1, R-7.2
November 6, 2024	Implement review timer for results page	Jared	R-7.3
November 7, 2024	Work on poll results export functionality	Mark, Jared	R-8.2
November 8, 2024	Develop examinee's poll result pages (correct/wrong answers) functionality	Diana	R-13.1, R-14.1

November 9, 2024	Backend validation of results for each examinee	Mark, Ohm	R-13.1
November 10, 2024	UI refinement for poll results and feedback session	Diana, Jared	Testing
November 11, 2024	Finalize and test result exporting and finish page	All team members	R-8.1, R-8.2

November 12 - 18, 2024 (Week 7: Full System Testing)

Goal: Test the complete instructor and examinee workflow.

Date	Task	Assigned to	Related Requirement
November 12, 2024	Test end-to-end poll session flow with multiple users	All team members	Full System Testing
November 13, 2024	Bug fixing and adjustments from test cases	All team members	Quality Assurance
November 14, 2024	Review and verify database integrity	Mark, Jared	System Integrity Testing
November 15, 2024	Test stress on real-time examinee count and poll progression	Diana, Ohm	Performance Testing
November 16, 2024	Test login, poll upload, session, results in multiple environments	Mark, Ohm	Compatibility Testing
November 17, 2024	Conduct User Acceptance Testing (UAT)	All team members	User Testing
November 18, 2024	Collect feedback and finalize bug fixes	All team members	Testing

November 19 - December 2, 2024 (Week 8: Final Testing and Presentation Preparation)

Goal: Finalize the project, fix bugs, and prepare for the presentation.

Date	Task	Assigned to	Related Requirement
November 19, 2024	Final bug fixes and performance improvement	All team members	Final Testing
November 20, 2024	Prepare project presentation materials	All team members	Presentation Preparation
November 21, 2024	Refine and finalize system documentation	Diana	Project Documentation
November 22, 2024	Ensure system configuration and deployment	All team members	System Deployment
November 23-2, 2024	Full team rehearses presentation	All team members	Presentation Preparation
December 3, 2024	Final review and project presentation	All team members	Project Presentation

This schedule guarantees that every team member is assigned particular responsibilities throughout the development process. It focuses on ongoing testing, managing risks, and ensuring timely completion to meet the project presentation deadline on December 3, 2024.

2.2. Activity Diagram

Activity: Each major task or phase of the project, such as "Requirements Analysis," "UI Mockups," and "Backend Setup." These activities break down the project into manageable segments.

Duration (weeks): The time allocated to complete each activity, measured in weeks. The durations are derived from the project plan and reflect the estimated effort for each phase.

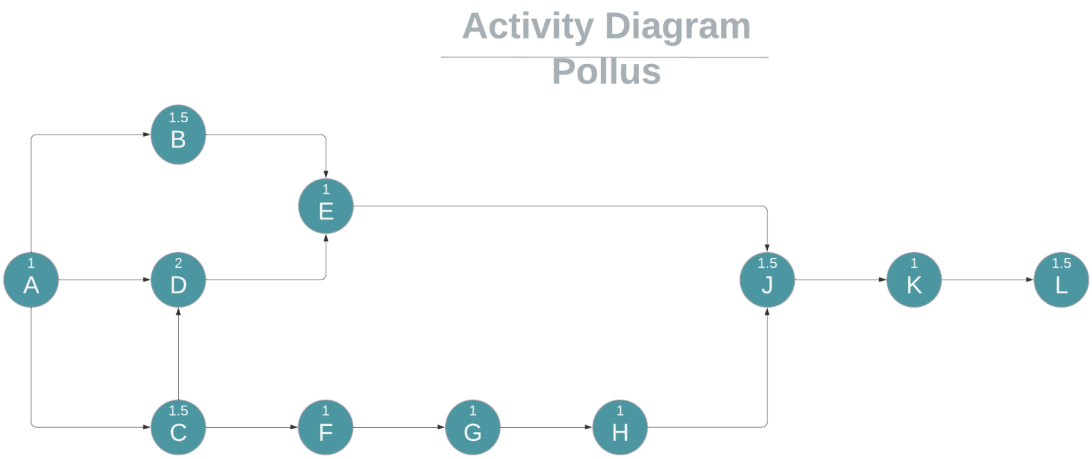
Dependency: The preceding activities that must be completed before starting the current one. This ensures that the project follows a logical flow and identifies critical paths that may impact the timeline.

Resources (team members): The number of people allocated to each task, along with their roles. It shows which team members are responsible for specific activities and how many are working on each task.

Activity	Duration (weeks)	Dependency	Resources
A: Requirements Analysis	1	None	Diana, Jared, Mark, Ohm (4)
B: UI Mockups	1.5	A	Ohm (1)
C: Database Design	1.5	A	Mark (1)
D: Backend Setup	2	A, C	Jared, Mark (2)
E: Instructor Login Development	1	B, D	Jared, Ohm (2)
F: Poll Upload Functionality	1	C	Diana (1)
G: Real-Time Lobby Development	1	F	Mark, Ohm (2)
H: Poll Session Development	1	G	Diana, Jared (2)

J: Poll Results Development	1.5	E, H	Mark, Ohm (2)
K: System Testing	1	J	Diana, Jared, Ohm, Mark (4)
L: Presentation Preparation	1.5	K	Diana, Jared, Ohm, Mark (4)

Below is the activity diagram for the "**Pollus"** project. It illustrates the workflow and dependencies between tasks, with each activity's duration shown inside the circles. The arrows indicate the order in which activities must be completed, providing a clear view of the project's progression.



We have the following **two critical paths**:

1. **A → C → D → E → J → K → L** = 9.5 weeks.
2. **A → C → F → G → H → J → K → L** = 9.5 weeks.

By highlighting the critical paths, the team can prioritize essential activities that are vital for meeting project deadlines, ensuring effective resource allocation and minimizing any potential delays.

3. Risk Management

Effective risk management is crucial to the success of the "**Pollus"** project. Identifying, classifying, and mitigating risks ensures that potential challenges are addressed before they can significantly impact the project timeline, cost, or quality.

3.1. Risk Identification

The area of risk where is best to focus for Pollus is in the sign up page/homepage, backend, and security vulnerability.

The key areas of risk for the "**Pollus"** project include:

Homepage and Sign-up Page Risks:

- **Risk:** Errors or performance issues in the sign-up and login processes could prevent instructors from accessing the system.
- **Impact:** This could cause major delays in user onboarding and disrupt poll creation, preventing instructors from using the system effectively.

Backend Integration Risks:

- **Risk:** Backend issues, such as database connectivity problems or API failures, could hinder the functionality of critical features like poll creation, real-time response tracking, and result generation.
- **Impact:** Failure in the backend could render the application non-functional, affecting user experience and data reliability.

Security Vulnerabilities:

- **Risk:** The system could be vulnerable to unauthorized access, data breaches, or loss of sensitive information such as instructor credentials and poll results.

- **Impact:** Any security breach could lead to significant data loss, legal issues, or loss of trust from users.

3.2. Risk Classification

Each identified risk is classified based on its probability of occurrence and potential impact on the project.

Risk Area	Probability	Impact	Classification
Homepage/Sign-up Page Failures	Medium	High	High-Risk
Backend Integration Failures	High	High	Critical-Risk
Security Vulnerabilities	Low	Very High	Critical-Risk

High-Risk: Medium probability and high impact; needs significant attention but is less likely to occur compared to critical risks.

Critical-Risk: High probability or very high impact; requires immediate focus and strong mitigation strategies to ensure project success.

3.3. Mitigation Strategies

To minimize the likelihood and impact of these risks, the following mitigation strategies will be implemented:

- **Homepage and Sign-up Page Failures:**
    1. Conduct thorough unit testing and integration testing on the sign-up and login modules to identify and fix bugs early.

2. Implement redundancy measures such as fallback mechanisms for errors and downtime.
3. Perform load testing to ensure the homepage and login page can handle peak traffic.

Expected Outcome: Ensure stable access to the platform for instructors and examinees, minimizing disruptions in user onboarding.

- **Backend Integration Failures:**
    1. Conduct system-wide integration tests to ensure that all backend APIs, databases, and third-party services are properly connected.
    2. Implement a robust error-handling framework to capture and resolve API failures or connectivity issues quickly.
    3. Use continuous monitoring tools to detect backend issues in real-time, allowing for prompt fixes.

    Expected Outcome: Ensure smooth data flow between the frontend and backend, with minimal downtime and interruptions.

- **Security Vulnerabilities:**
    1. Implement SSL/TLS encryption for all data transmissions and ensure proper encryption of sensitive data, such as passwords, using hashing algorithms like bcrypt.
    2. Perform regular security audits and penetration testing to identify and fix vulnerabilities in the system.
    3. Enforce role-based access control (RBAC) to limit access to critical features based on user roles.

    Expected Outcome: Maintain data integrity and security, protecting user credentials and sensitive poll results from unauthorized access.

4. Quality Assurance Planning

Effective quality assurance (QA) is vital to ensure that the "**Pollus"** project meets the desired standards of functionality, usability, and performance. This section outlines the quality standards, testing strategy, roles, defect management approach, and quality metrics that will guide the development and delivery of the project.

4.1. Quality Standards Definition

The following quality standards will be applied to the Pollus project:

- **Functional Completeness**: The system must fulfill all functional requirements as outlined in the Software Requirements Specification.
- **Performance**: The system should handle real-time interaction between instructors and at least 50 examinees without any significant lag or performance degradation.
- **Usability**: The interface will be intuitive and user-friendly, ensuring smooth navigation for both instructors and examinees, following best practices for web application usability.
- **Security**: The system must protect sensitive data, including encrypted credentials for instructors and secure handling of poll results.
- **Cross-browser Compatibility**: The application will be compatible with modern web browsers, including Chrome, Firefox, Edge, and Safari.
- **Accessibility**: The system will follow accessibility guidelines to ensure that users with disabilities can interact with the application effectively.

4.2. Testing Strategy

A multi-layered testing strategy will be employed to ensure that the system functions as expected and meets the defined quality standards. The testing process will include:

- **Unit Testing**: Each individual component (such as login, poll creation, and response submission) will be tested to ensure it performs as intended. This will be automated where possible.
- **Integration Testing**: After individual components are developed, they will be tested together to ensure smooth communication and data flow between different modules (e.g., the instructor's control panel and the backend database).
- **System Testing**: End-to-end testing will be performed to simulate real-world use cases, such as an entire poll session from creation to result export.
- **User Acceptance Testing (UAT)**: Instructors and examinees will be selected to test the system in a real-world environment, ensuring that the user experience aligns with expectations.
- **Performance Testing**: Load testing will be conducted to evaluate how the system performs under various stress conditions (e.g., a large number of examinees joining simultaneously).
- **Security Testing**: Penetration testing and vulnerability scans will be performed to ensure the system is secure against unauthorized access and data breaches.

4.3. Defect Management

A robust defect management process will be in place to track and resolve any issues encountered during the development and testing phases:

- **Defect Reporting**: All bugs, defects, or issues will be reported using a centralized tool like Jira or Trello, categorized by severity (e.g., critical, high, medium, low).

- **Resolution and Retesting**: Developers will resolve the reported defects, after which the team will perform retesting to confirm the issue has been fixed.
- **Defect Closure**: Once a defect has been resolved and retested, it will be closed in the tracking system.

4.4. Quality Metrics

To ensure the project's quality objectives are met, the following metrics will be monitored and reported:

- **Test Coverage**: The percentage of the code that is covered by automated tests, ensuring that a majority of the code has been verified by unit tests.
- **Defect Resolution Time**: The average time taken to resolve reported defects, providing insights into the efficiency of the development.
- **UAT Feedback**: The satisfaction level of end users based on feedback from the UAT process. This metric will focus on ease of use, performance, and overall user experience.
- **Performance Metrics**: System response time and load-handling capacity during performance tests, ensuring that the system meets real-time performance standards.

5. Configuration Management Strategy

5.1. Version Control System

The version control system that we are using is GitHub. GitHub allows us to branch off from each other’s code, test in isolation, as well as, integration testing through merging branches. It also helps the team keep track of the last working version of the code that we can go to in case of catastrophic

failure. We keep our code in a repository so that we will not lose all progress if a team member's computer breaks. Team members will regularly commit to the GitHub repository to ensure that no code base is left untracked.

Team Member	GitHub
Diana Bayandina	github.com/DianaBay
Mark Ospina	github.com/mospina1
Jared Fuller	github.com/JaredZX
Ohm Shah	github.com/Ohm-Shah

5.2. Build and Release Management

We will upload our components to separate branches. Once components are finished, we will slowly integrate each component into a combined branch for those two components. Following this, we will provide integration testing to make sure the combined branch still works on all machines. Once all components are integrated, we will begin system testing. After a build has been system tested, we will restart the cycle of working on components to merge those components into system testing. We will also add regression testing to ensure that previous features still work. The builds will be working prototypes, meaning they will have a limited number of features for each build until the release version is complete.

5.3. Configuration Item Identification

- Frontend (Website Pages)
    - Homepage
    - Sign In Page
    - Sign Up Page

    - Examinee Page
    - Instructor Page
    - Waiting Room Page
        - Instructor's View
        - Examinee's View
    - Polling page
        - Instructor's View
        - Examinee's View
    - Per Question Results Page
        - Instructor's View
        - Examinee's View
    - Per Poll Results Page
        - Instructor's View
        - Examinee's View
- Backend (Database)
    - Import CSV Files For Polls
    - Read CSV File
    - Construct CSV File into Poll
        - Poll Class
        - Question Class
            - Option members
        - Master Question Array
    - Store Individual Examinees' Answers

- Export Results as TXT File
- Progression Saving

5.4. Change Management Process

Changes must be approved by the team member(s) in charge of the changed component. When two components are being merged, it must be approved by the team members who designed and developed the code. When a merge is completed, a team member other than the one who started the merge must begin integration testing.

5.5. Audit and Documentation

There will be regular audits and documentation updates throughout our timeline. Audits will make sure that the team is on track and that we are progressing. Documentation allows others, including team members, to add to the code base and understand what the code does. Documentation will be updated as the code evolves in each component and will be edited if the combined features cause any changes.