GradeFast
Initial Project Plan

VERSION 1.0 | 10/23/2023

Celine Chiou, Will Eand, Mihai Siia, Aravind Sripada

1. Project Overview

Project Team

Will Eand - Scrum Master (rotating role every 2 months)
Aravind Sripada - Full stack developer
Celine Chiou - Full stack developer
Mihai Siia - Full stack developer

Project Description

This project has been specifically devised for use within introductory computer science courses at Bellevue College, such as CS 209, CS 210, and CS 211. The primary objective of this team is to develop a web application intended for instructors during code review. The overarching goal of this initiative is to alleviate the teaching burden by making the grading process more seamless while supplying real-time data to professors without imposing additional work demands. Simultaneously, it seeks to offer support optional resources for students for easy access into self-improvement.

Project Objective

The objective of this project is to develop a comprehensive software engineering code review tool tailored specifically for college instructors, enabling them to seamlessly assess and provide feedback on student code submissions, ultimately enhancing the education experience and facilitating better learning outcomes within the academic environment.

2. Scope

Inclusions

Minimum Viable Product (MVP):

1. Submission Management
    - Design a submission management system that allows instructors to upload individual assignment submission batches as a single zip file
    - Separate single zip file to each student file
    - Feature extracts and organize student files from the zip for teacher to grade

2. Applying Comments
    - Enable instructors access to a tool kit to seamlessly edit and provide comments
        - Allow instructors to review these automated changes in a clear manner

3. Exporting graded submissions
    - Teacher is able to export into a folder with graded assignments and associated names

Core Features/Additional Functionalities:

4. Store comment bank system
    - Allow professors to reuse previous comments on the same assignment to reduce redundancies
    - Allow professor to use shortcuts or drag-drop comments

5. Code Review Interface
    - Provide an intuitive and user-friendly interface for instructors to review and grade student submissions

- Display code files in a readable format (syntax highlighting)

6. Report generation
    - Generate a report with code and comments added by an instructor (the file will be shared with the student)
    - Generate summary reports on class performance for professor review

Exclusions

1. Automatic student generation from text files
2. Mobile compatibility for easy grading with mobile device
3. Smooth transition

3. Project Schedule

2023

- October: Project Start, Project Plan; SRS Plan, PRFAQ
- November: Prototyping, System Design, MVP
- December: End of Quarter 1 Presentation

2024

- January: Prioritize key feature to develop
- February: Code review for features, documentation, and add to main codebase
- March: Prioritize new key features to develop
- April: Code Review for features, documentation, and add to main code base
- May: Update features, make improvements, run testing for sample
- June: Project End, Final Presentation

4. Milestones

- November 21: Develop easy-to-use Web App UI
- February: Completion of MVP
- April: Add 3 additional features
- May: Overall testing with same and usage cases

5. Project Deliverables

- Easy to use user-interface
- Submission feature
- Commenting Feature
- Generate a report that gathers comment/critique data
- Exporting graded assignment feature

6. Resources

- Human Resources: Developers - Will, Celine, Aravind, Mihai
- Tools and Software: Tools: MacBooks, Windows Computers | Software: MongoDB, Express.js, React, Node.js, Typescript, React component libraries (e.g., Chakra UI, Grommet), Code-specialized LLMs (such as Starcoder or Code Llama)
- Hardware: Functioning computers Mac / Windows
- Budget: $0 Trials only for software

7. Quality Assurance and Testing

Testing Plan

1. Unit tests for method functionality.
2. Integration tests to make sure functionality between different services are properly communicating with each other.
3. Functional tests for making sure that output from integration values is correct.
4. End-to-end testing for making sure that users (instructors) can use the application in an overall setting in June with functional exports.

Quality Standards

1. Code review by (1) team member before merging with main branch
2. Document each feature and method with clearly defined purpose and usage implications.

8. Risk Management

Identified Risks

- Scope Creep
- Labor shortages
    - Busy professional life
- Unfamiliar technologies
    - Using tech stack unfamiliar to us
- Quality Assurance
    - Ex. Poor testing and quality control can lead to defects and issues

Mitigation Plan

- Addressing Scope Creep
    - Strong, detailed documentation and following through with plans.
- Addressing labor shortages
    - Maintain a flexible work arrangement (work from home)
- Addressing unfamiliar technologies
    - Collaborative learning by sharing knowledge and supporting each other
    - Maintaining a shared knowledge bank to use as a reference
- Quality Assurance
    - Peer reviews to prevent quality issues

9. Communication Plan

- Stakeholders: Fatma
- Communication Channels: Communicate with stakeholder through Microsoft Teams, Discord for team-related collaborations, in-person meetings, virtual meetings on discord
- Meeting Schedule: Once a week (To be decided)

10. Change Control

Change Request Process

Step 1: Project Team members or stakeholder submit a change request
Step 2: Scrum Master reviews change request and puts it to a vote with all team members.
Step 3: Impact Assessment: Team discusses feasibility, impact on scope, schedule, and resource limitations.
Step 4: Documentation: Document change, details, assessments, and outcome.
Step 5: Team votes on proposed change.

11. Appendices

Code Review Interface

1. Automatic Comment Generation
    - Generate a report on code conventions for Java programmers.
    - Implement an AI-powered code analysis tool to generate automatic comments based on code quality, style, and object-oriented design
    - Programming language is java, and target courses are CS 209, CS 210, CS211
    - Allow instructors to customize or edit these automatic comments

2. Archive Access
    - Allow instructors to access previous submissions and code reviews

3. Handling multiple class files
    - Allows professor to check multiple class files from a single student seamlessly

4. UI allows for tracking of how many students to left to grade
    - A UI sidebar that keeps track of students left

5. Assignment grade tracker (Allows teacher to keep track of grades for assignment for easy conversion into canvas later)
    - Student order can be adjusted based on class list manually for inputting grades
6. Automatic student generation from text files

7. Provide an intuitive and user-friendly interface for instructors to review and grade student submissions
    - Display code files in a readable format (syntax highlighting)