University of British Columbia Okanagan

COSC 499 – 201

Capstone Software Engineering Project

2023 Summer Term 1&2

Project Topic Number: #6

Title of project:

Gamified Coding Practice Platform

Assignment: Scope and Charter document

Draft: May 29th, 2023

Team Members:

Joe Gaspari

Archita Gattani

Gyumin Moon

Jason Ramos

Alrick Vincent

Project Name

Gamified Coding Practice platform

Owner, sponsor, project manager

Owner - Learnification Technologies
Sponsor - Scott Fazackerley
Project Manager - Scott Fazackerley

Stakeholder list

- Learnification Technologies - Parsa Rajabi
- The University of British Columbia
- Scott Fazackerley
- Primary users - students
- Educators and instructors
- Joe Gaspari
- Archita Gattani
- Gyumin Moon
- Jason Ramos
- Alrick Vincent

Problem Statement

Current e-learning platforms primarily focused around the computer science field lack focus around mastery learning. With the absence of repeatable content on these platforms, many learners both beginner and expert are left entering their next subject with only a fraction of the required knowledge to excel. The team's gamified learning platform aims to solve this problem by integrating AI components to address the work-intensive nature of unique repeatable content. By addressing this issue, the team creates an e-learning platform revolving around the idea of mastery learning to continuously practice content until students are sufficiently confident or competent in the subject matter.

About the project

The Gamified Coding Practice Application project, undertaken by Learnification Technologies, aims to help students learn and practice coding skills. By leveraging the power of modern technologies, personalization, and gamification, the application intends to create an engaging and efficient learning experience for students.

The primary objective of the project is to develop a dynamic web application that encourages students to practice coding by providing them with a gamified environment. The application will let students choose

their preferred programming language to focus on, allowing them to tailor their learning experience according to their individual needs and interests.

To achieve this, the project will utilize React JS, a powerful JavaScript library. It will form the foundation of the application's front-end, ensuring a seamless and user-friendly interface for students. The core functionality of the Gamified Coding Practice Application will be built around AI APIs. It will generate coding questions tailored to the selected programming language.

The gamified nature of the application will enhance student engagement by incorporating elements such as badges, leaderboards, and rewards. By making coding practice enjoyable and interactive, the application will encourage students to dedicate more time and effort to their learning journey.

Project purpose

To leverage the power of artificial intelligence (AI) in revolutionizing the way students learn to code. The project will create an interactive and immersive learning environment that empowers students to develop essential coding skills with ease and confidence. By harnessing the capabilities of AI, the project will provide personalized learning tailored to each student's unique needs, abilities and learning pace.

To enhance the efficiency and effectiveness of coding education by providing instructors with a comprehensive and customizable set of questions. The goal is to equip educators with the tools to seamlessly align coding assessments with their course curriculum, ensuring students are challenged appropriately while reinforcing key concepts. The project will harness AI to automate the process of generating coding questions, taking into account the desired learning outcomes, difficulty levels, and programming languages. The project aims to save instructors valuable time and effort by offering a diverse range of high-quality questions that cater to the specific needs of their students

Project Goals and Objectives

Project Goals:

- Develop a dynamic, gamified web application.
- Encourage students to practice coding skills.
- Provide a personalized learning experience.
- Utilize AI-assisted learning for generating coding questions.
- Foster continuous learning and improvement among users.
- Make coding practice enjoyable and engaging through gamification elements.
- Enhance student motivation and engagement in coding.
- Promote collaboration and competition among students.
- Improve coding proficiency and skills among users.
- Contribute to the advancement of education technology.

Project Objectives:

- Design and develop a user-friendly and visually appealing web application.
- Implement a selection mechanism for students to choose programming language and topics.
- Integrate AI technologies to generate coding questions based on language and topics selected by students.
- Develop an assessment system using AI to evaluate code submissions and provide feedback.
- Incorporate gamification elements such as achievements, leaderboards, and rewards.
- Ensure the application is accessible on various devices and browsers.
- Test and refine the application for usability, performance, and reliability.
- Gather user feedback to continuously improve and enhance the application.
- Monitor and analyze user engagement and progress within the application.
- Promote the application among educational institutions and coding communities.
- Provide documentation and support resources for users and administrators.
- Maintain data privacy and security of user information.
- Collaborate with AI technology providers for efficient integration and updates.
- Evaluate the impact and effectiveness of the application on students' coding skills.
- Stay updated with emerging technologies and educational trends to enhance the application's features and capabilities.

Project Objectives and Related Success Criteria

Project Objectives:

- Develop a dynamic web application that offers a gamified environment for students to practice coding skills.
- Provide students with the ability to choose their preferred programming language and specific topics for coding practice.
- Incorporate AI technology to generate coding questions based on the selected language and topics.
- Enhance student engagement through gamification elements such as badges, leaderboards, and rewards.
- Promote continuous learning and improvement among students by encouraging regular coding practice.

Success Criteria:

- Completion of the web application that provides an engaging and intuitive user interface.
- Successful integration of React JS for the front-end development, ensuring efficient data handling and powerful UI capabilities.
- Integration of AI APIs, to generate coding questions based on the selected language and topics.
- Implementation of gamification elements that motivate students to practice coding regularly and achieve milestones.
- Positive feedback and high user satisfaction from students, indicating an enjoyable and efficient learning experience.

- Increased student participation and time spent on the platform, demonstrating the effectiveness of the gamified approach.
- Improved coding skills and proficiency among students, measured through pre- and post-assessments or user surveys.
- Recognition and adoption of the Gamified Coding Practice Application by educational institutions or coding communities.
- Continued usage and engagement of the application by students over an extended period, indicating sustained interest and value.

High-level Requirements

- Development of an accessible, interactive web-based platform for learning different programming languages.
- Supported programming languages include JavaScript, Python, HTML/CSS, and potentially Java, C, C++.
- Integration of AI APIs to generate practice questions based on user-selected programming language, topics and proficiency.
- Implementation of rewards systems to incentivize user engagement and improvement
- Implementation of user progress tracking and optional display, including course milestones, points, badges.
- User roles include learners, instructors, and potentially admins.
- Secure user registration, login, logout, and password retrieval features.
- Classroom support for instructors to choose topics, manage students and view class analytics.
- Accessible system with responsive design for non-mobile devices.
- Development of the system shall be made using good software development practices, including code documentation, testing before committing, and using stable, useful libraries.
- Development of the system in React, NodeJS, SQL.
- Dockerization of the system for deployment.
- Client-side and server-side security.

Assumptions and Constraints

Assumptions:

- All potential users will have access to a stable internet connection to use the web-platform.
- Users may have little to no knowledge of programming concepts prior to using the web-platform
- The selected AI API will be running and fully functional throughout the project.
- The developers will be able to program in the selected technology stack (React, NodeJS, and SQL).
- Future expansion or maintenance of the system will not significantly affect the architecture of the system.
- All stakeholders will be available at all times.
- The main requirements for the project will be completed within the specified timeframe.

Constraints:

- The project is under severe time constraints.
- Some members of the development team will be away for disclosed amounts of time during the development time frame.
- The project's front-end must be developed using React and associated libraries.
- Limited user testing might cause disparities between client and user needs.
- Budget constraints may limit the scope of the project, as well as the AI capabilities.
- The development environment must be standardized in docker containers.
- Depending on the AI API choice, requests might be limited.
- Unit and integration tests must be passed before merging into the master branch.

High-level project description and boundaries

Description:

The deliverable product will be an interactive online platform which aims to facilitate the learning and practicing of programming skills. The system is designed to be accessible by anyone but will require registration of an account to use. Key features include the ability to choose the programming language to practice, select a topic of study as well as a level of proficiency. Users will be able to practice on AI-generated questions, and receive personalized feedback after submitting their answers. The system will offer a variety of badges, achievements and milestones for the users to chase and show off in their personal profile. Additionally, the platform empowers instructors to access a structured learning environment akin to a virtual classroom. This includes distinct leaderboards for each class, and the ability to set mandatory topics for students, thus fostering a competitive and focused learning environment.

Boundaries:

The project's focus is on providing an interactive learning platform rather than a comprehensive commercial, educational solution. It will not incorporate features that are more relevant to Learning Management Systems such as Canvas, such as instructor-student messaging, graded assignments or other. Furthermore, the system will not provide an API to be integrated into other learning platforms such as prairie learn or canvas.

The platform is meant to be used on computers. As a result, mobile devices will not be supported initially. The project's programming language will initially be limited to JavaScript, Python, HTML/CSS, but could be expanded to Java, C and C++, with the possibility to include even more languages in the future.

Due to time constraints, the developing team will need to prioritize certain functionalities over others. To facilitate requirements management, the Moscow Prioritization System will be used. As such, tasks labeled with "must" will have highest precedence and define the core features of the project.

Lastly, the generated questions' relevance and correctness rely on the capabilities of the AI engine. This dependence on the AI's capabilities set hard boundaries for the development of the project.

High-level risks

When working with a team there are inherent high-level risks that need to be managed effectively. The potential risks to consider:

1. Communication and collaboration: Effective communication and collaboration are essential for project success. Risks may arise if team members struggle to communicate their ideas clearly, leading to misunderstandings, delays, or misaligned goals. Differences in work styles, time zones, or language barriers can also pose challenges to collaboration.

2. Scope creep: As the project progresses, there is a risk of scope creep, where new features or requirements are added without proper evaluation or documentation. This can lead to project delays, increased costs, and compromised quality if not managed and controlled effectively.

3. Skill and knowledge gaps: Each team member brings unique skills and expertise to the project. However, there may be risks associated with skill or knowledge gaps, where certain team members may struggle to meet project requirements. This can hinder progress and impact the overall quality of the software.

4. Dependencies and coordination: In a team-based project, there are often interdependencies between various tasks and deliverables. If dependencies are not well-managed or team members fail to coordinate effectively, it can result in bottlenecks, delays, and reduced productivity.

5. Technical challenges: Building software projects can involve complex technical challenges, such as integrating different systems, dealing with scalability issues, or ensuring security measures. These challenges may require specific expertise or research, and any lack of technical proficiency or planning can increase the risk of project failure.

6. Time and resource management: Effective time and resource management are crucial for meeting project deadlines and ensuring project success. Risks may arise if team members underestimate the effort required for tasks, fail to prioritize effectively, or encounter unforeseen obstacles that impact project timelines.

Summary milestone schedule

1. Week 1 (Starting May 14th):
    - Background document preparation

2. Week 3 (Starting May 28th):
    - Finalized requirements and WBS/FBS due

3. Week 4 (Starting June 5th):
    - Design presentation and design documents due

4. Week 8 (Starting July 2nd):
    - Mid-project status update and assessment, MVP (Minimum Viable Product) design review presentation

5. Week 14 (Starting August 13th):
    - Delivery of the final report and code

Each of these milestones represents a significant checkpoint in the project where important deliverables are expected.

Summary Budget

Money: The project will utilize one of two major AI API's on the market, these APIs charge per token where each query may contain many tokens and multiple queries will occur during the normal usage of the software. Thus the project will require a dedicated cash flow towards usage of the API. (The current price of this usage has not been determined). There will be an additional cost associated with Digital Ocean hosting, this will be covered by the project sponsor.

Time: The development team consists of 5 individuals who are expected to provide on average 20 hours per week to the completion of the features required to release a MVP (Minimal Viable Product) to the client.

Development Process

The development process of the project will be guided by the SCRUM-BAN framework with the following key features:

1) Daily standups meetings which may vary in length based on the nature of the work.
    a) Discussion of progress, requirements, and pending talking points .
    b) Delegation of new tasks amongst team members.
    c) Backlog management to reprioritize pending tasks.
    d) Retrospectives to identify and reflect on development pain points.

2) Weekly client meetings
    a) Gather feedback and approval on current ideas and/or progress of the project.
    b) Brain-storming solutions on project pain points.
    c) Realignment of team vision with client vision.

3) Action boards
    a) Indicate pending issues, their notes and responsible team members.
    b) Track progress of pending issues through categorized sections.

4) Data-driven development.
    a) Tracking time for tasks, issues addressed, and other performance metrics.
    b) Identify bottlenecks or areas of improvements indicated by the measured data.

Development standards and tools

Programming Languages and Technologies:

- JavaScript, HTML/CSS3, Node.js, React, etc.

Coding Standards:

1. Naming Conventions:
    - Use meaningful and descriptive names for variables, functions, and classes.
    - Follow camelCase naming convention for variables and functions.
    - Use PascalCase naming convention for class names.
    - Avoid using abbreviations and single-letter variable names.

2. Commenting:
    - Include comments to explain the purpose and functionality of complex or non-obvious code sections.
    - Add comments to provide context and explanation for important algorithms or business rules.
    - Ensure that comments are clear, concise, and up-to-date with the code.

3. Formatting:
    - Use consistent and readable indentation with spaces (e.g., 2 or 4 spaces).
    - Keep lines of code within a reasonable length (e.g., 80 characters) to improve readability.
    - Maintain a consistent code style throughout the project.

4. Error Handling:
    - Use appropriate error handling mechanisms, such as try-catch blocks, to handle exceptions.
    - Provide meaningful error messages and handle exceptions gracefully to prevent crashes or unexpected behavior.

5. Code Structure:
    - Follow modular and maintainable design principles.
    - Encapsulate related functionality within classes and functions.
    - Keep functions and methods concise and focused on a single responsibility.
    - Avoid deep nesting of control structures to improve code readability.

6. Testing and Documentation:
    - Write comprehensive unit tests to ensure the correctness of the code.

- Include meaningful test case names and cover edge cases.
- Document the purpose, inputs, outputs, and usage of functions and classes.
- Maintain up-to-date documentation alongside code changes.

Tools:

- IDE: Visual Studio
- Version control systems: Git
- Project management tools: Google Drive, Slack, Discord,.
- Unit testing: Jest
- CI/CD: Git Action
- Deployment, Hosting: Docker, Digital Ocean

Software Development Methodologies:

ScrumBan, Kanban

Project Requirements

Functional requirements

Must Have:

1. The system must be accessible through a publicly available online domain.
2. The system must be able to accept user feedback on questions and reward feedback with bonus points and badges, which prompts system improvements.
3. The system must be able to prompt AI-generated questions based on a specified topic, level of proficiency, and programming language chosen by the user.
4. The system must be able to track and display user progress through course milestones, user points, achievement badges, and usage analytics.
5. The system must allow instructors the ability to view student and class analytics including students' progress in the assigned questions, how many they have completed, how many they have left, percent success rate, visual dashboard of progress, number of questions they have completed, their streak/points and the number of badges earned.
6. The system must support user login, logout, registration, and password retrieval via email linked to the account via email linked to the account.
7. The system must be able to indicate whether user-provided answers are correct or incorrect and provide meaningful AI-generated feedback with high confidence e.g. 95%.
8. The system must be able to validate whether an AI-generated question relates to the topic selected with a high degree of confidence (95%)
9. The system must support JavaScript, Python, and HTML/CSS.
10. The system must support the main user archetype: learners.
11. The system must include tooltips to guide users.
12. The system must be visually appealing to the users with a modern design.

Should have:

1. The system should be able to give users AI-generated suggestions or hints when prompted by the user or a series of incorrect answers.
2. The system should have a public leaderboard categorized into topics and programming languages. This can be turned off, and the scores will be calculated by correct answers (weighted by question difficulty), milestones, streaks and badges.
3. The system should contain user dashboards for students showing course progress, badges earned, and other user achievements.
4. The system should further support Java, C, C#, and C++ .
5. The system should easily accommodate additional programming languages for students to practice.
6. The system should support the main user archetype: Instructors.
7. The system should have a class platform for instructors to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.
8. The system should allow learners to join classes through a join code similar to Kahoot platform.
9. The system should have an FAQ page that includes questions/answers that may be popular.
10. The system should include terms of service and privacy policy information.

Could have:

1. The system could have an embedded IDE supporting autocorrection, linting, and semantic highlighting (which may give local variables distinct colors to improve code comprehensibility and structure).
2. The system could have an additional user archetype: administrators
3. The system could have an admin portal to view site usage analytics and (update/delete/add) users and classes.

Non-functional requirements

Must Have:

1. The system must pass an accessibility audit (WCAG 2.1).
2. The system must be responsive for all non-mobile devices.
3. The system must have a seamless and straightforward user interface that minimizes user clicks or actions to achieve the goal.
4. The system must be properly documented for future developers through written documents and in-line code documentation.
5. The system must have a guide for teachers and students on how to use the platform.
6. The developers must create test cases before writing or committing new additional features or components.
7. The developers must be comfortable using HTML, CSS, JS, React, NodeJS, and SQL.

Should have:

1. The system should not store personally identifiable information except email.
2. The AI-generated questions should have less than a 5% user report rate.
3. The system should allow instructors the ability to view the confidence score of each of the questions generated for the student.

Could have:

1. The system could provide users with a dark theme option.
2. The system could be responsive for mobile devices.

Will not have:

1. The system will not support an adjustable modularized layout design to allow users to personalize the layout of different UI components.

Technical requirements

Must Have:

1. The system must be developed using ReactJS for the front-end.
2. The system must be developed using NodeJS for the back-end.
3. The system must update asynchronously.
4. The system must integrate with an AI API to generate practice questions.
5. The system must be dockerized for deployment.
6. The system must securely store user passwords through salting or hashing.
7. The system must have server-side security for user inputs.
8. The system's database must be a relational database.
9. The system must have validation for all user-input fields.
10. The system must make use of stable, popular libraries instead of in-house solutions.
11. The system must include server-side logging that captures all database queries and events that may lead to a failure.

Should have:

1. The system should indicate to users any errors found within the system.
2. The system should have client-side security for user inputs.

User Requirements

Must have:

1. All users must be able to register and login to the system.
2. All users must have the ability to change their password and reset it through the email linked to their account.
3. Learner users must be able to choose their programming language and topic and level of proficiency.
4. Learner users must have an alias or username.
5. Learners must be able to view their streak while accessing the platform.
6. Leaders must be able to see a timer when their streak is about to expire via the platform and/or email notification.
7. Instructor users must be able to create a class and share a join code.
8. Instructor users must be able to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.
9. Instructors must be able to create and keep track of multiple courses at the same time.

Should have:

1. All users should be able to delete their accounts.
2. All users should be able to edit their profile information.
3. All users should be able to view other public user's achievements through their dashboard.
4. Learner users should be able to emphasize chosen achievements through chosen badges and courses.
5. Learner users could have the option to opt into or out of public leaderboards.
6. Instructor users should be able to archive courses after the term has ended.

Could have:

1. Learner users could have personalized and customizable profile cards to emphasize favorite achievements.
2. Learner users could receive notifications about their progress and streak reminders.
3. Learner users could be alerted when class lessons are updated.
4. Learner users could be alerted about their course progress and streaks via notifications (either via email or via the web application).

Project Deliverables

Deliverable #1 - Requirements, Scope, and Charter
- Due June 4, 2023

Deliverable #2 - Design Document
- Due June 9, 2023

Milestones

Week 1 (May 14th):

- Background document

Week 2 (May 21st):

- Client IP Agreements
- Report of Initial Finding
- Project Charter and Scope
- Preliminary Analysis Summary
- Preliminary Requirements and WBS/FBS

Week 3 (May 28th):

- Requirements and WBS/FBS Due **
- Preliminary Design Documents

Week 4 (June 5th):

- Design Presentation/Design Documents Due **
- Weekly Reporting

Week 5 - 7 (June 12th ~):

- Weekly Reporting
- WIP Demo and Testing Update

Week 8 (July 2nd):

- Mid-project Status Update and Assessment **
- MVP Design Review Presentation **

Week 9 - 13 (July 9th ~):

- Weekly Reporting
- WIP Demo and Testing Update

Week 14 (Aug 13th)

- Final Report and Code Delivery ***

Cost Estimates

AI API Expenses: The project will be integrating with one of two primary AI APIs currently available. These APIs operate on a cost-per-token basis, with each API query involving multiple tokens. Given the regular usage of the software, this will necessitate a consistent budget allocation for the API usage. As of now, the exact cost of this expense is yet to be determined.

Hosting Expenses: Our project will be hosted on DigitalOcean, an associated cost that will be covered by the project sponsor. Cost is estimated to be $10/month.

Labor Expenses: Our team, composed of five developers, will be dedicating, on average, 20 hours per week towards the project. The effort will be directed towards developing the necessary features to release a Minimum Viable Product (MVP). There is no exact monetary value for this labor expense.

Work Breakdown Structure

	Estimated Hours and Assignment

1.0 Learning	Joe Gasparri	Archita Gattani	Gyumin Moon	Jason Ramos	Alrick Vincent
1.1 Team Meetings (Cumulative)	25	25	25	25	25
1.2 The developers must be comfortable developing in React	10	10	10	2	20
1.3 The developers must be comfortable developing in NodeJS	10	5	5	1	10
1.4 The developers must be comfortable using SQL	5	5	5	1	5
1.5 The developers must be comfortable with the CI/CD workflow	5	6	6	4	10
1.6 The developers must learn the fundamentals of Docker	5	10	10	4	5
1.7 The developers must research relevant popular React/NodeJS libraries	5	5	15	5	5
Total number of Hours Assigned per Team Member	**65**	**66**	**76**	**42**	**80**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**5.91**	**6.00**	**6.91**	**3.82**	**7.27**

	Estimated Hours and Assignment				
2.0 Front-end Design	**Joe Gasparri**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
2.1 Research and conduct case studies on current market platforms	3	5	5	3	2
2.2 Design relevant userflow and user experience diagrams	10	10	5	5	15
2.3 Produce color themes, typography, brand elements	10	15	10	5	10
2.4 Produce initial design mockups	10	20	10	15	15
2.5 Gather and implement client feedback for final interface	11	11	11	4	11
Total number of Hours Assigned per Team Member	**44**	**61**	**41**	**32**	**53**

Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.00**	**5.55**	**3.73**	**2.91**	**4.82**

	Estimated Hours and Assignment				
3.0 Back-end Design	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
3.1 Define system architecture	5	5	5	5	5
3.2 Design routing	5	5	5	5	5
3.3 Design relational database schema	10	10	10	10	10
3.4 Design API & Format API requests	10	5	5	5	10
3.5 Generate test data	5	2	2	2	2
3.6 Design Open AI integration	5	5	5	0	5
3.7 Design question confidence engine	10	5	5	2	10
Total number of Hours Assigned per Team Member	**50**	**37**	**37**	**29**	**47**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.55**	**3.36**	**3.36**	**2.64**	**4.27**

	Estimated Hours and Assignment				
4.0 Front-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
4.1 Prepare Routing System	2	5	5	4	4
4.2 Implement initial sematic site structure through JSX	2	25	30	15	2
4.3 Implement site styling through Tailwind	2	15	15	15	4
4.4 Implement site interactions through JSX	2	60	65	50	4
4.5 Refactoring and modularization	15	25	35	15	15
4.6 Integration with back-end API	10	10	15	8	10
Total number of Hours Assigned per Team Member	**33**	**140**	**165**	**107**	**39**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**3.00**	**12.73**	**15.00**	**9.73**	**3.55**

	Estimated Hours and Assignment				
5.0 Back-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
5.1 Prepare Routing System	15	5	2	10	15
5.2 Dockerize SQL database, with appropriate schema	15	3	2	0	15
5.3 Build question confidence engine	20	2	1	0	20
5.4 Format data returning from Open AI API for the front-end	15	0	1	0	15
5.5 Develop server-side validation and error handling	10	0	0	0	10
5.6 Build data recall queries for SQL database	10	2	3	0	15
5.7 REST API creation	25	5	3	5	25
Total number of Hours Assigned per Team Member	**110**	**17**	**12**	**15**	**115**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**10.00**	**1.55**	**1.09**	**1.36**	**10.45**

	Estimated Hours and Assignment				
6.0 Test & Testing	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
6.1 Regression testing	2	0	2	0	2
6.2 Performance testing	2	5	5	2	2
6.3 Unit testing	20	20	20	20	20
6.4 Integration testing	10	10	10	4	10
6.5 System testing	5	3	5	3	2
6.6 Usability testing	10	5	5	5	10
6.7 User Acceptance testing	10	5	5	3	10
Total number of Hours Assigned per Team Member	**59**	**48**	**52**	**37**	**56**

Weekly Average of Hours Assigned per Team Member (11 weeks)	**5.36**	**4.36**	**4.73**	**3.36**	**5.09**

	Estimated Hours and Assignment				
7.0 Reports and Documentation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
7.1 Scope & charter document	2	2	2	2	2
7.2 Design document	10	10	10	10	10
7.3 Weekly team/personal logs	5	5	5	5	5
7.4 Mock ups/Diagrams	5	5	5	5	5
7.5 Final Report	20	20	20	20	20
7.8 Practice final presentation	5	5	5	5	5
Total number of Hours Assigned per Team Member	**47**	**47**	**47**	**47**	**47**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.27**	**4.27**	**4.27**	**4.27**	**4.27**

	Estimated Hours and Assignment				
8.0 Deployment	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
8.1 Build and maintain CI/CD workflow with Github actions	10	10	5	6	10
8.2 Set up a Digital Ocean host for system	10	5	2	2	10
8.3 Post-deployment monitoring	5	5	2	3	5
Total number of Hours Assigned per Team Member	**25**	**20**	**9**	**11**	**25**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**2.27**	**1.82**	**0.82**	**1.00**	**2.27**

Approvals

Parsa Rajabi

| --- | --- | --- |
| Project Sponsor | Signature | Date |

Scott Fazackerley

| --- | --- | --- |
| Project Sponsor | Signature | Date |