

# CS 362 Group 17 Project Proposal

## 1. Team info

- List each team member and their role in the project.
  - Team Leader: Samuel Jamieson
  - Code Quality Enforcement Officer: Oscar Ludwig
  - Document Review and Assurance Officer: Jacob Porter
  - Chief Coding Officer: Kai Turner
  - Communication Officer: Andrew Vu
  - Research and Resource Coordinator: Cameron Dilworth
- Link to each project relevant artifact such as your git repo and or project management system
  - Github: <https://github.com/srj0407/Group17-Project>
  - Trello board: <https://trello.com/b/JSf9TgUc>
- List communication channels/tools and establish the rules for communication.
  - Microsoft Teams: team and project related discussion, meeting times and updates.
  - Communication rules: Be respectful to team members, work constructively to resolve differences, and respond in a timely manner.

## 2. Product description

### Team name: Project Painter

- Samuel Jamieson, Oscar Ludwig, Jacob Porter, Andrew Vu, Cameron Dilworth, Kai Turner.
- **Abstract:** The first paragraph of your document must be an abstract (or executive summary, or TL;DR) that explains your project at a high level. If someone read nothing but that one paragraph, what do you want them to know?
  - In modern educational environments, Canvas has become a mainstay in both higher and lower educations. While Canvas offers a robust learning platform, its interface and other functionalities are severely lacking. These shortcomings fail to meet the needs of both students and instructors alike. It only further exacerbates the communication gap between students and instructors. This project aims to develop a Google Chrome extension that will better the user experience and attempt to fill many of Canvas's shortcomings. This project will improve upon canvas productivity, accessibility, and engagement with its growing student and

instructor population. Building an entirely new system for students to interact with would be impractical for students, as the transition from one interface to another requires time and commitment to learning a system when most students are already stretched on time. This extension would aim to solve many of the shortcomings of canvas while not changing the primary interface. It is suitable to fix many issues with a simple upgrade package marketed to simplify students' experiences to allow for greater efficiency. This extension aims to reduce the efficiency problems with Canvas as a learning platform without sacrificing the familiarity it has gained with students and instructors.

- **Goal:** What are you trying to do? For example, what task or problem will your system help users with?
  - Improve student productivity, accessibility, and engagement with their classes. Tasks such as time management and communication are features of Canvas that lack in many ways. An example being that students often need to search through Canvas pages and/or syllabuses that have much more info (noise) disguising what they're searching for; leading to frustration and potential ignorance.
- **Current practice:** How is it done today, and what are the limits of current practice?
  - There is a "Better Canvas" extension that adds a couple of features we plan on implementing, but overall looks and interacts the same as traditional canvas. There's also "Tasks for Canvas" however that only adds a "completion wheel" to the side canvas to better display due dates and assignment progress.
- **Novelty:** What is new in your approach, and why do you think it will be more successful than other approaches? Do not reinvent the wheel or reimplement something that already exists, unless your approach is different.
  - We're not trying to re-create Canvas, we're just trying to improve its shortcomings. Outside a couple ideas, most of our goals are features that already exist in Canvas in some way, but the execution is poor and could use an update from a group of users that rely on it. Examples such as the Canvas Inbox being overlooked by just emailing professors directly, and having to search through a Canvas page or entire syllabus to find when TA office hours are that work with your schedule, and many more. Additionally, we plan to add accessibility features to make more people able to use canvas effectively and support people who may have trouble with its current user interface.
- **Effects:** Who cares? If you are successful, what difference will it make?
  - Improve student productivity, accessibility, and engagement with their classes. Students will spend less time frustrated with differing and confusing Canvas course pages, and spend more time finding the information they need as soon as they open the home page.
- **Use Cases (Functional Requirements):**
  1. Jacob Porter
    - a. **Actors:** Student who wants to see TA times for their class
    - b. **Triggers:** Student needs help with completing their weekly assignment
    - c. **Preconditions:**
      - i. Student is "Logged-in" to their school-related Canvas account

- ii. Student is on the “Dashboard”
    - iii. Student has already filled in their TA times for the class in the table provided by the extension
  - d. **Postconditions (success scenario):** Student sees a table of TA hours for Monday-Friday of their desired class
  - e. **List of steps (success scenario):**
    - i. Student “scrolls down” their Canvas Dashboard
    - ii. Student views the “TA Times Table”
    - iii. Student is able to see the TA times related to the class they're searching for
  - f. **Extensions/variations of the success scenario:**
    - i. Student does not need to “scroll down”
    - ii. Student sees TA times of other classes too
  - g. **Exceptions (failure conditions and scenarios):**
    - i. Student has no classes on their Canvas page
      - 1. Table is “empty”
    - ii. Classes on their Canvas page don't specify TA times
      - 1. Table is “empty”
2. Kai Turner
- a. **Actors:** A student in a class.
  - b. **Triggers:** The student wants to see what grade they need on an assignment to achieve a grade in the class and how hard they need to try (how high their grade would be compared to previous grades).
  - c. **Preconditions:** The student has an assignment and grade they want to see as well as previous class grades if they want to see how hard they need to try.
  - d. **Postconditions (success scenario):** The user is displayed the grade they need and percentile they would need to be in relative to their work.
  - e. **List of steps (success scenario):** The user inputs the grade they desire and the assignment they want to check. Then the program uses math to compute the results. Then the program displays the results.
  - f. **Extensions/variations of the success scenario:** N/A
  - g. **Exceptions:** failure conditions and scenarios: The student inputs a grade that cannot be reached, in which case the program will alert them and stop. The student may also have too few assignments to sample, in which case the program would alert them and only display the required grade and not the percentile.
3. Andrew Vu
- a. **Actors:** Students and instructors who prefer a customized theme for improved accessibility or visual comfort.
  - b. **Triggers:** A student or instructor finds the default Canvas interface visually uncomfortable.
  - c. **Preconditions:**
    - i. The Chrome extension is installed and active.
    - ii. The user is logged into Canvas.

- d. **Postconditions (success scenario):** The user applies a customized theme (e.g., dark mode) that persists across sessions.
  - e. **List of steps (success scenario):**
    - i. The user logs into Canvas using the Chrome browser with the extension installed.
    - ii. The user opens the extension's settings menu.
    - iii. The user selects a predefined theme (e.g., dark mode) or customizes colors manually.
    - iv. The extension dynamically updates Canvas pages with the selected theme.
    - v. The theme settings are saved locally and persist across user sessions.
  - f. **Extensions/variations of the success scenario:**
    - i. The extension provides multiple predefined themes (e.g., high contrast mode, warm tones).
    - ii. Users can share and download custom themes created by others.
  - g. **Exceptions (failure conditions and scenarios):**
    - i. The user selects a theme that causes readability issues > A reset option is available.
    - ii. Changes to Canvas's DOM break theme application > The extension reverts to default or provides a warning.
4. Oscar Ludwig
- a. **Actors:** A student in a canvas course
  - b. **Triggers:** A student who wants to quickly see a standardised layout of syllabus information
  - c. **Preconditions:**
    - i. The Chrome extension is installed and active.
    - ii. The user is logged into Canvas.
  - d. **Postconditions (success scenario):**
    - i. User sees standardized information of Instructor and TA contact information and office hours and grading policies all in a table for easy reading
  - e. **List of steps (success scenario):**
    - i. The user goes to <https://canvas.oregonstate.edu/> to access their dashboard
    - ii. The user selects a class with support for the standardized syllabus
    - iii. The user selects the "Standard Syllabus" link on the middle-left sidebar
  - f. **Extensions/variations of the success scenario:**
    - i. The extension could have user entered information on the "Standard Syllabus" page or it could have data synced with an online database.
  - g. **Exceptions: failure conditions and scenarios:**
    - i. The link does nothing and no table is accessible
    - ii. There is no link
    - iii. There is no data filled for the class, so no information is visible
5. Samuel Jamieson

- a. **Actors:** A student trying to keep track of what they need to do and trying to find motivation to complete it.
  - b. **Triggers:** A student who wants to see a neat and engaging to do list with incentives to complete tasks.
  - c. **Preconditions:**
    - i. The Chrome extension is installed and working correctly.
    - ii. The user is logged into canvas and is either on the home page or the calendar
  - d. **Postconditions (success scenario):**
    - i. The user sees a list of tasks to do as well as an associated points system and rewards for completing assignments early. A colorful and engaging screen greets them with a number of custom options for decoration.
  - e. **List of steps (success scenario):**
    - i. The user logs into canvas and accesses the calendar feature to see the gamify section.
  - f. **Extensions/variations of the success scenario:** N/A
  - g. **Exceptions: failure conditions and scenarios:**
    - i. The canvas extension is working incorrectly or is not properly installed or syncing with Canvas LMS.
6. Cameron Dilworth
- a. **Actors:** A student who is disorganized
  - b. **Triggers:** A student wants to eliminate stressors about organization, and complete assignments with ease and minimal planning.
  - c. **Preconditions:**
    - i. The Chrome extension is installed and working properly
    - ii. The user is logged into Canvas and linked their extension
  - d. **Postconditions (success scenario):**
    - i. The user sees an options menu and can navigate through, select preferences and save them.
  - e. **List of steps (success scenario):**
    - i. User opens Canvas and is logged in
    - ii. User selects the Extension menu
    - iii. User navigates between each class
  - f. **Extensions/variations of the success scenario:**
    - i. Once saved, the preferences are updated and stay the next time Canvas is opened.
  - g. **Exceptions: failure conditions and scenarios:**
    - i. The extension shows up blank
    - ii. The extension is not accessible
    - iii. The extension is not installed correctly
    - iv. The extension does not save any changes to the theme or preferences
- **Non-functional Requirements:**
    - The user's data should be protected and private from other users.

- Users should be able to utilize all features of the extension in 6 clicks or fewer from a given canvas page
- The extension should be easy to modify and improve without large rewrites to core functionality.
- **Technical approach:** Briefly describe your proposed technical approach. This may include what system architecture, technologies, and tools you may use.
  - To build a robust Chrome extension packed with functionality for Canvas LMS, the technical approach will need to include a planning phase, modular development, and an integration phase with the Canvas API.
- **Requirements Analysis**
  - Identifying Core features: This means prioritizing enhanced notifications, email verification, a customizable side dashboard, and accessibility features.
  - Research into the Canvas API or alternatively Web Scrapers: Understanding the endpoints in the Canvas API for accessing assignment, grades, announcements, and profiles of the students and instructors.
  - Browser Extension Constraints: We will need to assess Chrome extension permissions, data storage limits, and script restrictions
- **Architecture Design**
  - Modular Architecture:
    - Content Scripts: Inject scripts to interact with Canvas pages modifying DOM elements for customizing dashboards
    - Background Scripts: Handle long-running and data intensive processes like notifications and API polling
    - Features for other pages: Provide user interface for features like grade average, or adding assignments that don't exist yet.
  - Data Flow:
    - Canvas API pipes to the Background Script pipes to the content script, then pipes to the DOM.
    - Use Chrome's messaging system for communication pipelines
  - Storage: Use Chrome's local storage or IndexedDB for user preferences and cached data.
- **Coding Languages**
  - Frontend:
    - JavaScript, HTML, and CSS for UI and DOM manipulation
    - React for creating dynamic and reusable components in popup and options pages
  - Backend:
    - Canvas API for fetching/updating user data

- Background scripts for API interaction and data syncing. This process may also improve runtime speeds.
  - Firebase for email and account verification
- Database:
  - Chrome Storage API for lightweight local data storage
  - IndexedDB for storing large amounts of data (Course Materials, User setting and preferences)
  - Firebase for email and account verification
- **Development Workflow**
  - Setup and Permissions:
    - Configure manifest.json for appropriate permissions within chrome
    - Define content script injection rules to target Canvas LMS URLs
  - Canvas API Integration:
    - Implement OAuth2 for secure authentication for Canvas LMS
    - Build utility function for API requests
    - Store API responses to reduce request frequency and improve load times
  - Feature Developments:
    - Faster Load Times: Fetching whitelisted linking within the current page, and holding them until the user clicks off the page.
    - Dark/Theme mode: Inject CSS dynamically into Canvas pages using content Scripts
    - Built in chat: Database to store and authenticate chat messages of students displayed on Canvas page
    - Email Verification: Separate website and email service for users to create an account and verify a school email
    - Gamify Calendar: Track students submission dates and give points based on how early assignments are submitted that can be used for visual updates to their “Calendar” in Canvas
    - Names and emails of Professors and TAs: Scrap HTML of pages to find keywords and create a table on the dashboard it to be visible all in one place
    - Assignment Averages: HTML/JS injection for seeing these numbers easier
    - Easy Access to Syllabus and Grading Policies/TA Office Hours: A table of quick access information like emails, office hours, grading policies, with the syllabus fluff removed and in a constant organization between classes either stored in a local chrome storage with the single

- users content or the database with public user generated content
  - Automatic grade calculator (complicated): HTML/JS injection, adding a system that reads and analyzes grades to give users a summary of how they should move forward in the class
  - Hide Old Canvas Pages: hide canvas courses that are not properly removed from dashboard
- UI Design:
  - Use React to build popups and options pages
  - Design simple to use, and intuitive interfaces with Material-UI or Tailwind
- **Deployment Plan**
  - Packaging:
    - Minify Javascript and CSS files for efficient loading
    - Generate an optimized manifest.json with accurate permissions
  - Publishing / Distribution:
    - Submit the files to the chrome extensions webstore
  - Updates:
    - Set up version control in manifest.json
    - Release bug fixes and new features if applicable
- **Risks:** What is the single most serious challenge or risk you foresee with developing your project on time? How will you minimize or mitigate the risk? Don't state generic risks that would be equally applicable to any project, like "we might run out of time".
  - Authentication (Stretch Goal):
    - Use OAuth2 to securely authenticate users with Canvas and a separate website
    - Ensure token storage is secure using Firebase
  - API Rate Limits:
    - Cache frequent API responses locally and refresh whenever possible to reduce load times within canvas
  - DOM Changes by Canvas:
    - Use robust Selectors and dynamic scripts to adapt Canvas updaters
  - Performance Optimization:
    - Minimize background script activity
    - Optimize DOM Manipulation to reduce load times and further lag
- **Subgroups and Assigned features**
  - Subgroup 1:
    - Assigned Features:
      - Canvas Side Menu per class with important information
        - Easier Access to Links (Syllabus)
          - Possibly standardized table of syllabus information
        - Quick to view Info Table



- Names and Emails of your professors
    - TA office Hour table
    - Grading Policies Page / table in Canvas
  - Fix 10am assignments appearing on the next day
  - Gamify Calendar on canvas (Due assignments sideboard)
- Gamify Canvas Calendar (integrated into the side bar)
  - Points system
  - Cosmetic Rewards for Completing assignments
  - Additional points for completing assignments early
- Assigned Stretch Goals if major features completed early:
  - AI Powered Ranked Importance of Announcements
  - Rate my Course Implementation
- Members:
  - Samuel Jamieson
  - Oscar Ludwig
  - Cameron Dilworth
- Subgroup 2:
  - Assigned Features:
    - Faster load time
      - Pre - Cache web data while canvas loads to reduce load times
    - Accessibility Menu for cosmetic customization
      - Dark/Themes Mode
      - Hide Old Canvas Classes from Dashboard
    - Improve Grades Screen
      - Average Grade Feature / Assignment Averages
      - Add “What If” Assignments for future unseeable assignments on Grades Page
      - Automatic grade calculator (complicated)
  - Assigned Stretch Goals if major features completed early:
    - Friends in classes and view their grades
    - In-Built Canvas chat room per canvas class
      - Email verification
  - Members:
    - Kai Turner
    - Jacob Porter
    - Andrew Vu
- **Timeline:** Tentative high-level weekly goals for your project for the coming 9 weeks.
  - Week 1: Gather the group and discuss project ideas. Develop early plans on project requirements and group communication/understanding.
  - Week 2: Refine project, and create project sprint plans. Create and agree on a “Team Contract” to ensure everyone understands what is expected. Prepare the “Project Presentation” for Week 3 as well as complete Project Proposal.

- Week 3: Present “Project Presentation”. Split project group into two subgroups assigned to work on different features. These groups will be assigned their own sprint and timeline to follow. The research team will start to delve into what APIs we will be utilizing and how we will interface with the canvas API.
- Week 4: Start the first sprint goal for each subgroup. Gamify calendar for group 1, and faster load times for group 2.
- Week 5: Subgroup 2 finishes sprint 1 goals, and moves onto review of sprint 1. Beginning sprint 2 for subgroup 2, this can consist of planning and research for the next phase or starting the project sprint 2. Subgroup 1 continues with Sprint 1, should be on track for completion next week.
- Week 6: Subgroup 2 completes sprint 2 by end of week, subgroup1 completes sprint 1 by end of week, review of code will take place.
- Week 7: Both groups begin the final sprint for the development phase after code review has been completed.
- Week 8: Both groups should have completed sprint development phases by the beginning of the week, leaving time for beta and exhaustive testing the remainder of the week. Subgroups will be disbanded once both sprints are completed.
- Week 9: Beta testing results will be analyzed and compiled into a list of features that can be added, and the project will be reviewed and redesigned if needed.
- Week 10: Finalize any remaining requirements of the project and class assignments.

Additionally, add the following:

- 4+ major features you will implement.
  - Accessibility Menu for cosmetic customization
    - Dark/Themes Mode
    - Hide Old Canvas Classes from Dashboard
  - Canvas Side Menu per class with important information
    - Easier Access to Links (Syllabus)
      - Possibly standardized table of syllabus information
    - Quick to view Info Table
      - Names and Emails of your professors
      - TA office Hour table
      - Grading Policies Page / table in Canvas
    - Fix 10am assignments appearing on the next day
    - Gamify Calendar on canvas (Due assignments sideboard)
  - Gamify Canvas Calendar (integrated into the side bar)
    - Points system
    - Cosmetic Rewards for Completing assignments
    - Additional points for completing assignments early
  - Improve Grades Screen
    - Average Grade Feature / Assignment Averages

- Add “What If” Assignments for future unseeable assignments on Grades Page
  - Automatic grade calculator (complicated)
- Faster load time
  - Pre - Cache web data while canvas loads to reduce load times
- 2+ stretch goals you hope to implement.
  - AI Powered Ranked Importance of Announcements
  - Rate my Course Implementation
  - Friends in classes and view their grades
  - In-Built Canvas chat room per canvas class
    - Email verification

## 4. Team process description

*Describe your quarter-long development process.*

- Specify and **justify** the software toolset you will use.
  - We will be using JavaScript and HTML/CSS as it makes it easy for the project to be imported in most browsers, as well as being used throughout the web making it possible for us to implement our features.
  - We will also use VSCode to develop our extension, with its extensive toolset and extensions.
- Define and **justify** each team member’s role: why does your team need this role filled, and why is a specific team member suited for this role?
  - Team Leader: Samuel Jamieson
    - A Team Leader is necessary to ensure that we all have someone to report to, as well as someone to engage in communication with the professor. Samuel has created web extensions before, so he has the experience to guide the project as a whole. He also has lots of team and communication experience to ensure the group can successfully work together towards our goals.
  - Code Quality Enforcement Officer: Oscar Ludwig
    - A Code Quality Enforcer is necessary so that all code is readable by other group members, as well as passable for prototyping so that the feature is viable. Oscar has experience writing and reviewing code from an internship he participated in last summer, so he’s well qualified to so for our group this term.
  - Document Review and Assurance Officer: Jacob Porter
    - A Document Review and Assurance Officer is necessary so that the group can be confident in the completion and submission of assignments. Jacob help manage the assignments for his last group in Software Engineering I, so he has experience ensuring that everyone completes

their sections in group assignments, reviewing the work, and submitting on time.

- Chief Coding Officer: Kai Turner
  - A Chief Coding Officer is necessary because the group needs someone who is knowledgeable to ask questions to, as well as get guidance from. Kai is competent in many aspects of software engineering, and was a leader in development in his previous Software Engineering I class.
- Communication Officer: Andrew Vu
  - A Communication Officer is necessary so that the group can always be on the same page in and out of the classroom. Andrew is well-spoken and has lots of group work experience that he'll use to keep everyone informed and on track with the goals of the project.
- Research and Resource Coordinator: Cameron Dilworth
  - A Research and Resource Coordinator is necessary so that the group can have someone to decide how and what the software should look and execute in order for it to be successful. Cameron has experience with web-development and research, meaning he can combine both previous and newfound knowledge to present to the group when questions arise.
- Provide a schedule for each member (or sub-group) with a *measurable*, concrete milestone for each week. "Feature 90% done" is not measurable, but "use case 1 works, without any error handling" is.
  - *Samuel Jamieson*
    - Week 3: Begin work on my weekly assigned feature according to the Project Schedule after the project presentation has been completed.
    - Week 4: Continuing working on feature implementation as expected. Make sure weekly meetings with TA are happening consistently and accordingly.
    - Week 5: Proceed with Sprint 1 development for my assigned team. Implement backend API calls for sidebar as well as front end integration.
    - Week 6: Complete sprint 1 and authorize it for testing and review by other subgroups.
    - Week 7: Begin work on Gamify calendar points system. Ensure all Instructor checkpoints are met and the team is keeping with the assigned deadlines.
    - Week 8: Complete sprint 2 and authorized testing and for other subgroups. Begin beta testing and control group testing.
    - Week 9: Gather data from beta testing and assist other subgroups with other implemented features if needed.
    - Week 10: Wrap up final beta reviews and implement appropriate changes. Make sure the product is ready for deployment come week 11.
  - *Kai Turner*
    - Week 3: Complete weekly assignments and begin developing my assigned feature.

- Week 4: Finish a system to tell the user what grade they need on an assignment to get a grade in the class.
- Week 5: Integrate a percentile based on the user's previous grades to the previous system.
- Week 6: Program a system to load canvas web pages before they are visited.
- Week 7: Using the previous system cache canvas links in the canvas site making it load faster.
- Week 8: Assist with other features/development or take on another feature if assumed to be able to be completed by the end of the term.
- Week 9: Continue to assist with other features/development or complete another feature.
- Week 10: Wrap-up development and assignments.
- *Jacob Porter*
  - Week 3: Complete weekly assignments and begin developing my assigned feature.
  - Week 4: Continue developing my assigned feature so that it passes error handling.
  - Week 5: Complete my assigned feature and begin getting user feedback for it.
  - Week 6: Implement feedback from previous week, refine assigned feature based on feedback.
  - Week 7: Assist with other features/development or take on another feature if assumed to be able to be completed by the end of the term.
  - Week 8: Continue to assist with other features/development or work on another feature so that it passes error handling.
  - Week 9: Continue to assist with other features/development or complete another feature, get feedback if time permits.
  - Week 10: Wrap-up development, implement feedback if time permits.
- *Cameron Dilworth*
  - Week 3: Complete weekly assignments and begin developing my assigned feature.
  - Week 4: Finish implementation to where the user can select the "options tab" and close it
  - Week 5: Make sure different options can be selected and saved \*these options do not need to work yet\*
  - Week 6: Finish each different option and have them change the layout \*make sure each option works\*
  - Week 7: Get feedback from team and random users and implement new features/touch up
  - Week 8: Find and fix any bugs that come up during verification and validation
  - Week 9: Make sure other team members are finished, and assist where needed.

- Week 10: Finish all assignments early and double check that all different features work smoothly together.
  - *Oscar Ludwig*
    - Week 3: Complete weekly assignments, meet with my sub-group and start work on assigned features.
    - Week 4: Continue with assigned code development, attend TA meetings, answer any questions about code quality.
    - Week 5: Complete first feature assignment hopefully, possibly the standardized Canvas syllabus pages.
    - Week 6: Final touches to complete sprint 1 and write tests for my code for other sub-groups
    - Week 7: Begin working on next feature, possibly an extension of the standardized syllabus to sync with a database. This week would be finding a database to use and a design that other members approve of.
    - Week 8: Finish database sync and complete sprint 2. Start beta testing software.
    - Week 9: Assist members with software if needed, work on bug fixes for my code, ensure code quality.
    - Week 10: Finish assignments and Make sure code is ready for final submission.
  - *Andrew Vu*
    - Week 3: Complete weekly assignments, meet with my sub-group to discuss feature development, and begin planning Dark/Theme Mode.
    - Week 4: Start code development for Dark/Theme Mode, attend TA meetings, and answer sub-group questions on code quality.
    - Week 5: Complete the first version of Dark Mode, integrate theme toggles, and start testing across Canvas pages.
    - Week 6: Finalize and polish Dark/Theme Mode, write unit tests, and integrate feedback for Sprint 1 completion.
    - Week 7: Begin extending Dark/Theme Mode to include custom themes and accessibility options; propose UI designs.
    - Week 8: Implement custom themes, finish Sprint 2, and start beta testing the Dark/Theme Mode feature.
    - Week 9: Assist with software integration, address bugs from beta testing, and ensure feature meets code quality standards.
    - Week 10: Finalize Dark/Theme Mode, ensure full functionality, and support the team in preparing for final submission.
- Specify and explain at least three major risks that could prevent you from completing your project.
  - Canvas being changed significantly or shut down.
  - Chrome or Firefox updating or modifying the extension code or how it is used.

- Poor performance combined with unrealistic expectations from our team could lead to this project being incomplete.
- Describe at what point in your process external feedback (i.e., feedback from outside your project team, including the project manager) will be most useful and how you will get that feedback.
  - During the development process, we will be testing features with other OSU students to get feedback on the successes and shortcomings of our extension provide. This will be most useful when deciding what parts of a feature should or should not be present, along with gathering new ideas to make the product better.