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
    - https://github.com/srj0407/Group17-Project
- List communication channels/tools and establish the rules for communication.
    - Microsoft Teams: team and project related discussion, meeting times and update.
    - 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.
- **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.
- **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**
            - <u>Identifying Core features</u>: This means prioritizing enhanced notifications, email verification, a customizable side dashboard, and accessibility features.
            - <u>Research into the Canvas API or alternatively Web Scrapers</u>: Understanding the endpoints in the Canvas API for accessing assignment, grades, announcements, and profiles of the students and instructors.
            - <u>Browser Extension Constraints</u>: We will need to assess Chrome extension permissions, data storage limits, and script restrictions
        - **Architecture Design**
            - <u>Modular Architecture</u>:
                - 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 or 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 iinjection,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
            - <u>UI Design</u>:
                - Use React to build popups and options pages
                - Design simple to use, and intuitive interfaces with Material-UI or Tailwind
        - **Deployment Plan**
            - <u>Packaging</u>:
                - Minify Javascript and CSS files for efficient loading
                - Generate an optimized manifest.json with accurate permissions
            - <u>Publishing / Distribution</u>:
                - Submit the files to the chrome extensions webstore
            - <u>Updates</u>:
                - 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”.
    - <u>Authentication</u>:

        - 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

Additionally, add the following:

- 4+ major features you will implement.
    - Dark/Themes Mode
    - Built in Discord based chat (Only if we don't want to use our own DB)
    - Email verification
    - Easier Access to Links (Syllabus)
        - Possibly standardized table of syllabus information
    - Gamify Calendar on canvas (Due assignments sideboard)
    - Names and Emails of your professors
    - TA office Hour table
    - Grading Policies Page / table in Canvas
    - Rate my Course Implementation
    - Average Grade Feature / Assignment Averages
    - Add "What If" Assignments for future unseeable assignments on Grades Page
    - Hide Old Canvas Classes from Dashboard
    - Fix 10am assignments appearing on the next day
    - Automatic grade calculator (complicated)
- 2+ stretch goals you hope to implement.
    - Screen Reader
    - AI Powered Ranked Importance of Announcements
    - Faster load time
    - Screen Reader
    - Friends in classes and view their grades

You should aim to commit your living document to your project's repository latest by **01/14/25 11:59 PM.**