1.1 Plan Introduction

The software development plan provides the details of the planned development of the Squibble web application which provides a website with a whiteboard application that can be added to, deleted, and modified.

Squibble is a web application developed with React and Node.js that will allow a user to sign in and add multiple types of media to the whiteboard, including text, images, drawings, and post-it notes with all of the aforementioned media. This website will last for three days before resetting itself, deleting all current media and archiving it in the media archive, via screenshotting. Squibble is being developed by passionate developers due to a goal of improving personal coding skills and having a fun senior project that can be completed in a reasonable timeframe. The project's milestones include going from the whiteboard display at the end of September all the way to final testing and debugging in early December.

1.1.1 Project Deliverables

- Project Proposal Document + Presentation: Opening ideas and presentation for Squibble.
- **Due September 11th**
- Whiteboard Display: General whiteboard functionality allowing for scrolling and media interaction.
- **Due September 22nd**
- Requirements Specification: Document and detail every part of Squibble and its creation.
- **Due September 25th**

- Software Development Plan: Initial plan detailing software development and assignment plan for Fall 2024.

- **Due October 9th**

- Media Objects: Adding functionality for images, gifs, text, and media applications to be added to the whiteboard.

- **Due October 19th**

- Post-it Note Object: Adding media object that stacks on top of other whiteboard objects and combines all other media.

- **Due October 26th**

- Software Design Document Architecture: Details software components and interactions of Squbble

- **Due October 30th**

- Header + Firebase: Adding firebase functionality and header fixes to current setup and media.

- **Due November 4th**

- Timer: Add timer for resetting website and link it with firebase.

- **Due November 9th**

- Alpha/Beta/Critical Design Review Presentation: Presentation showing off near end project.

- **Due Week 12-14 (November 11th-29th)**

- User Manual Final Updates: Final updates on how to run and use Squibble

- **Due November 13th**

- Software Design Document Detailed: Highlights classes and interfaces throughout project code.
- **Due November 13th**
- Archive: Add archive that screenshots website and stores screenshots for future use.
- **Due November 19th**
- Performance and Environment Testing: Testing and modifying changes to website + possible early launch.
- **Due November 25th**
- Final Testing and Debugging: Wrapping up Squibble development.
- **Due December 3rd**
- Final Presentation: Final Presentation of Squibble.
- **Due December 4th**
- Final Product Delivery + Code Freeze: Freezing Squibble development.
- **Due December 9th**
- Written Status Reports: Reports detailing current status of project.
- **Due every week after Week 8 (October 14th -December 9th)**

1.2 Project Resources

The Developers of Squibble include:

- Aidan Esposito
- Matthew Savitt
- Alex Gordon
- Parker Jazayeri

1.2.1 Hardware Resources

Hardware used for Squibble development and presentation include:

- Four laptops ranging from Windows 10-11 machines to Apple Mac Laptops
- Firebase applications hosted on Google Servers
- Apple iPhone (for testing of mobile website)
- HDMI cords for hooking up laptops to projector
- Projector used to broadcast images during presentation

1.2.2 Software Resources

Software used for Squibble development and presentation include:

- Windows operating system (10-11)
- Apple Mac operating system
- VStudio Text Editor
- React Web User Interface Designer
- Balloon-css extension for React
- Node.js v22.9.0 Javascript environment
- Javascript coding language
- Puppeteer API for screenshotting
- Tenor API for gif search
- Google Firebase for authentication, database, and hosting
- Google Docs for writing project descriptions
- Google Excel for schedule making
- ChatGPT for code testing and clean up
- General web browsers for hosting website

1.3 Project Organization

The Squibble team is composed of the following members with the following roles:

- Aidan Esposito:

Developer in charge of launching and hosting React and Firebase backend. Will help implement API use throughout the project and will help create media and general whiteboard functionality. In charge of the first website header design and will help future changes. Helps organize text chat and general meetings.

- Matthew Savitt

Frontend developer that is chiefly in charge of managing development of the user interface, graphical components, user experience, and optimizing drawing and client-side image-rendering and animation mechanics. Will be mainly responsible for guaranteeing a smooth and intuitive feel for the end user that is both convenient and satisfying to wield. Communicates with partners via text channel over iMessage, and github.

- Alex Gordon

Will help implement media objects such as gifs, images, and text onto canvas. Very flexible and looking to help out wherever possible! In communication through weekly meetings in the Keck Lab, and weekly communication through messages.

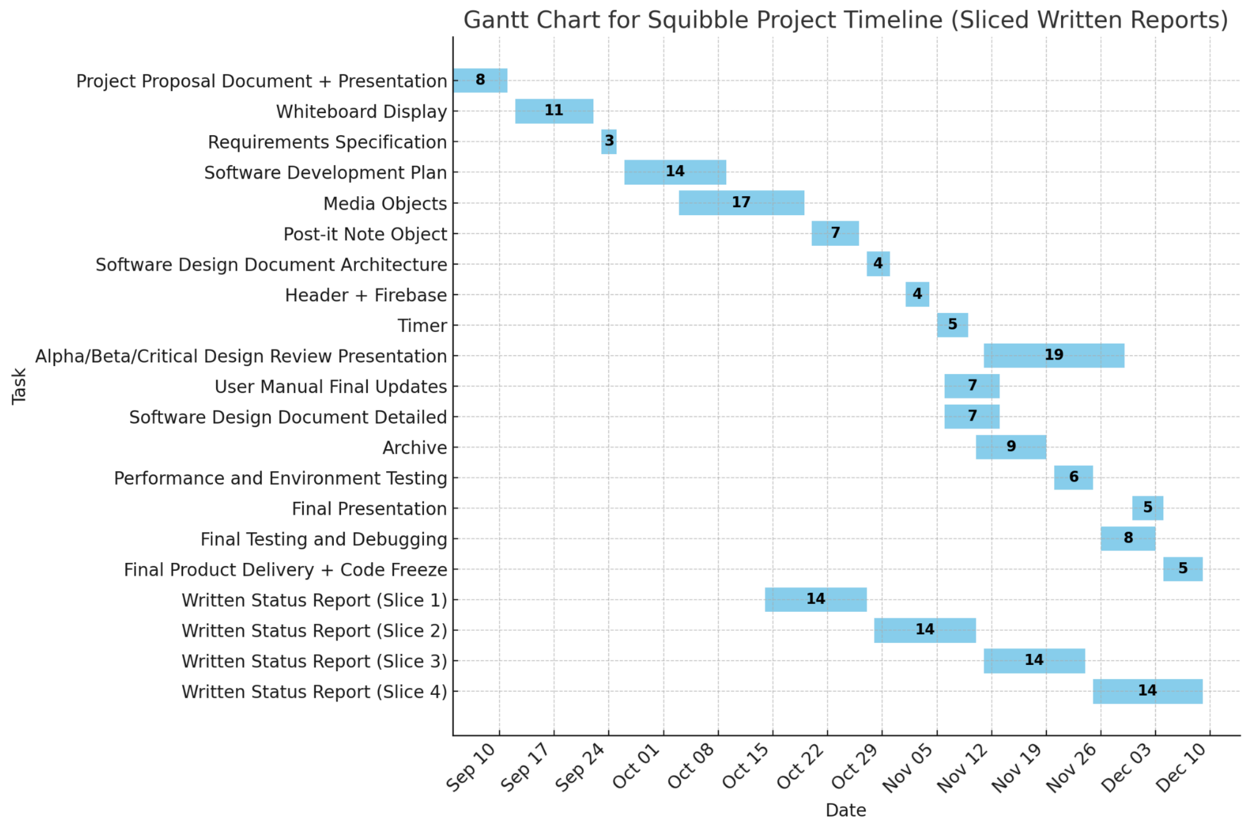
- Parker Jazayeri

Implementing GIF functionality and API usage. Generally helping test functionality and user experience, and fixing problems therein alongside other members. (This is making sure things work well *as* we create them as opposed to having someone else test a prototype.)

1.4 Project Schedule

This section provides the schedule information for the Squibble project. The full details will be broken down into the following sections, which are described below.

1.4.1 GANTT Chart



1.4.2 Task / Resource Table

Team Name	**Resources**	**Tasks**
Aidan	Windows 11, VS Code	- Header development - Text media object implementation - Firebase integration - Assist with Archive page
Matt	Windows 10, VS Code	- Pen, eraser, marquee, and panning tool development - Insert media object tools for user interaction
Parker	Windows 11, VS Code, Testing on non-Chromium (Firefox) which has shown some differences	- Integrate API for GIF search and addition (Tenor/Giphy) - Testing compatibility on non-Chromium browsers
Alex	Mac Laptop, VS Code	- Image media object upload functionality - General support for user experience - Collaborative work on the Archive