Notation Plus

Group 7

Chandler Crisp, Cristhian Hoyos, Edwin Nealis,

Issac Hall, Isabella Ochsner, Markus Beamer, Zack Harris

Project Overview

Notation Plus is the ultimate solution for efficient note-taking in your browser. It allows users to effortlessly take notes in the margins or lines of any online document or book, just like a real physical book. Our goal is to simplify note-taking for anyone who loves to read or needs it for school or work.

Our features include a note hub where all your notes for a particular page are saved, allowing users to delete individual notes or all of them simultaneously. Every time you open a book and start taking notes, those notes are saved to that individual document or book, ensuring everything stays organized.

Our Sticky Note feature allows users to drag notes on the screen to exactly where they want them on the document or book, making it easy to keep track of important information. Additionally, we generate genre maps based on the kind of books or documents users read, giving them a visual representation of what they're reading and recommending new books or documents to try.

Finally, our "Random Book of the Day" button when clicked displays a new random book every day, powered by Goodreads. All of these features are connected to your Google account, so users can easily access their notes on any device. With Notation Plus, note-taking has never been easier or more efficient.

Architectural Overview

Notation Plus is designed to be used in unison with your browser, specifically with the web pages and books, so we created a Google Chrome Extension. By creating an extension, we can pass information from the browser to the user. Our project works by creating three different

types of files, the manifest, content scripts, and finally HTML pages. So the manifest, manifest.json, is used as the base configuration of the extension. This includes information about the name, description, current version, chrome API keys, and files used throughout. The next type of file is the content script. These scripts are used to actually inject the host page with code. By adding these, we can then interact and modify each page in the browser, thus allowing us to display information to the user. The final type of file is the actual HTML pages, these are displayed to the user such as the popup.html, and when we display the list of notes to the user.

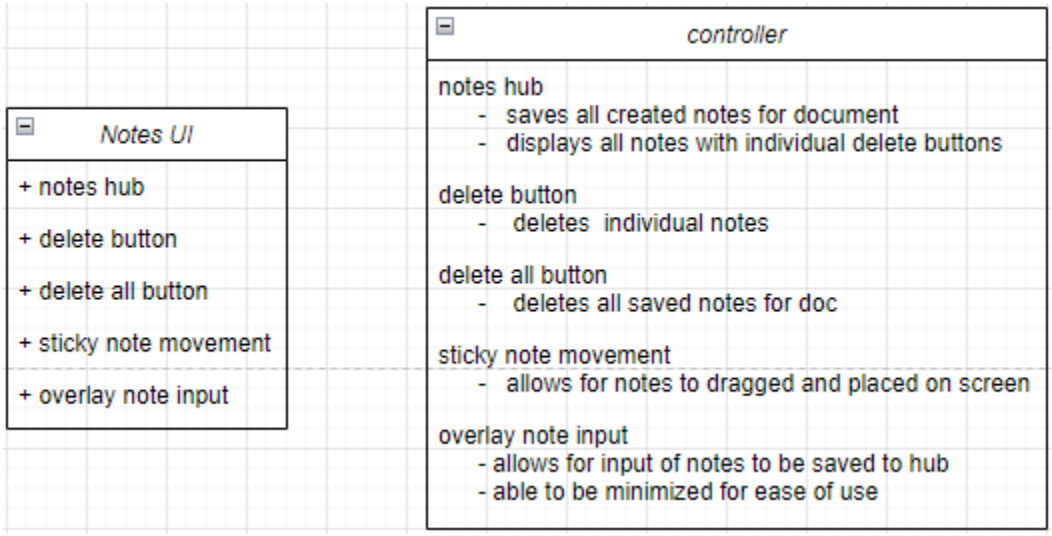
For our extension, we used HTML, CSS, and JS to create a sudo-MVC architecture. The model in our extension would be the Chrome API and the web pages the user selects. Both of these serve as the sources for the data we use throughout. While we don't have dedicated "controllers", the event listeners contained in the content scripts act as the controller deciding what functions need to be run. Finally, the other js and HTML pages, with CSS, act as our view and what the user sees on the page.

Subsystem Architecture

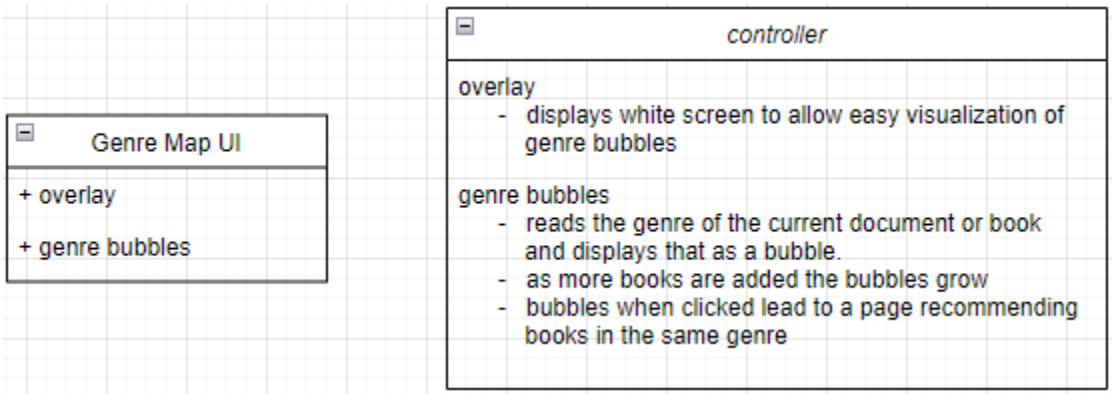
There are three major subsections of our software architecture: notes, genre maps, and random book of the day.

- Notes
    - Composed of an overlay that displays over the user's web page allowing us to display created notes. Also allows us to add individual deletion functionality and delete all notes. Currently, the overlay is broken up into 3 parts: note hub locked on the right of the screen, and "sticky note" which we currently have locked to the left side of the screen but plan on making it movable around the screen. On the

right side of the screen is where users currently can type in notes to be displayed on the hub and the “sticky note”.

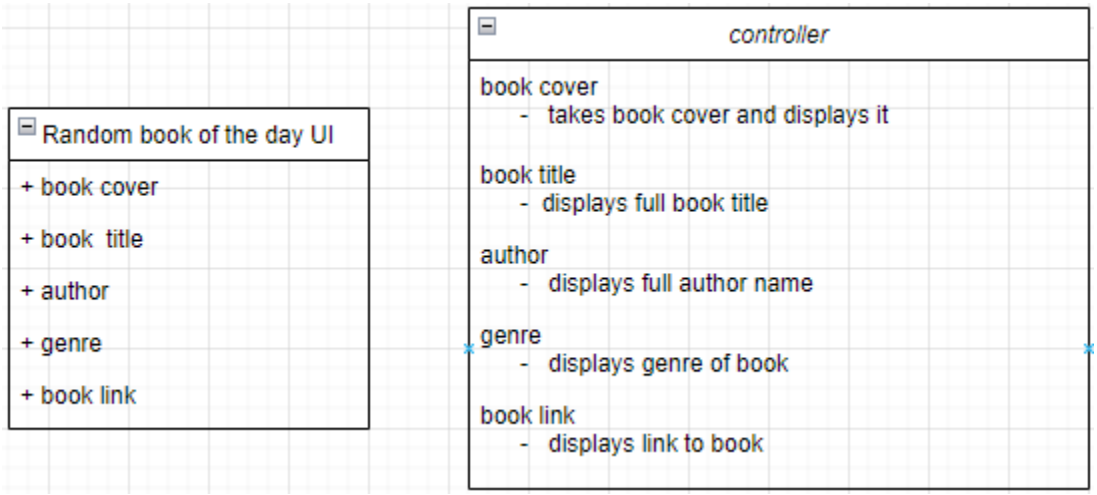


- Genre map
    - This is still currently in progress but the plan is to have a visual representation of the user's documents and books genres. The map would be changed based on the continual reading of the user, and allow them to click on the genre bubbles to give the recommendation of books in the same genre.



- Random book of the day
    - Displays a random book of the day in the overlay so users can simply see the book and decide if they want to read it or not, without leaving their current page.

This is done by using the Google API and displaying the results such as the book cover, the title, the author, and the genre on the overlay with a hyperlink to the book.



Deployment Architecture

This software will run on a single processor.

Persistent Data Storage

Throughout Notation Plus, we use various information that needs to be saved in a central location. This information includes the user's notes and some basic information on each book and webpage. All the information is saved using the Google Chrome Storage API. The API provides users with an extension-specific way to save user data and state. For our extension, we specifically use the storage.sync area that allows us to save this information to the browser itself. This allows users to still access their notes on any browser that the user is logged into and has the extension installed. Additionally, if the user does not have a Chrome account or is offline, the information will be saved locally until the user either gets back online or creates an account. For the notes, we save the title, comment, and the scroll position on the page into a separate json file.

We can then use this file to display the note at the same location it was recorded. For each web page and book, we save the genre and other information used in our genre mapping functionality.

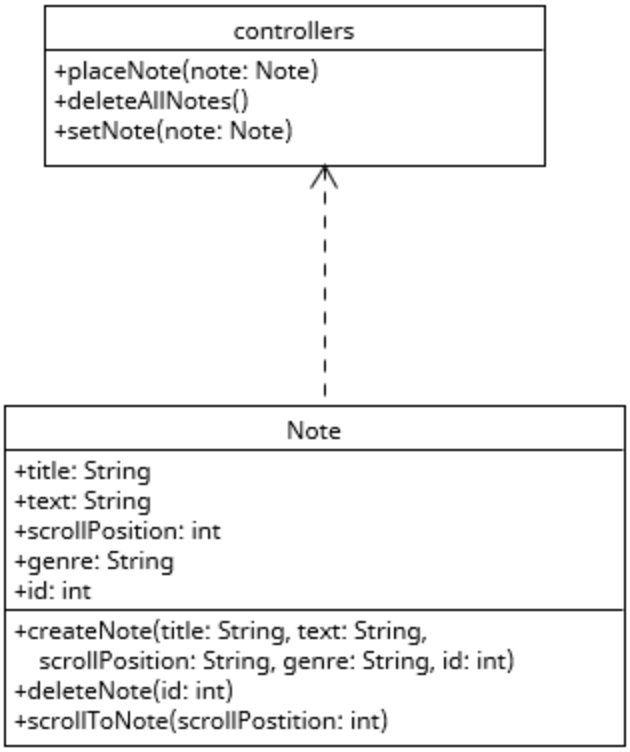
Global Control Flow

Our software is event-driven. As users read their documents and books, they decide when to take notes. As they progress through the documents they can decide to move their notes as well as click the delete button when they want to get rid of a note or completely start over. We have one time-dependent part, a random book of the day displays a random book every 24hrs. We don't do the randomization but we call from a website and we just display it and if the user clicks on it, they will be taken to the website for more info on the book.

Detailed System Design

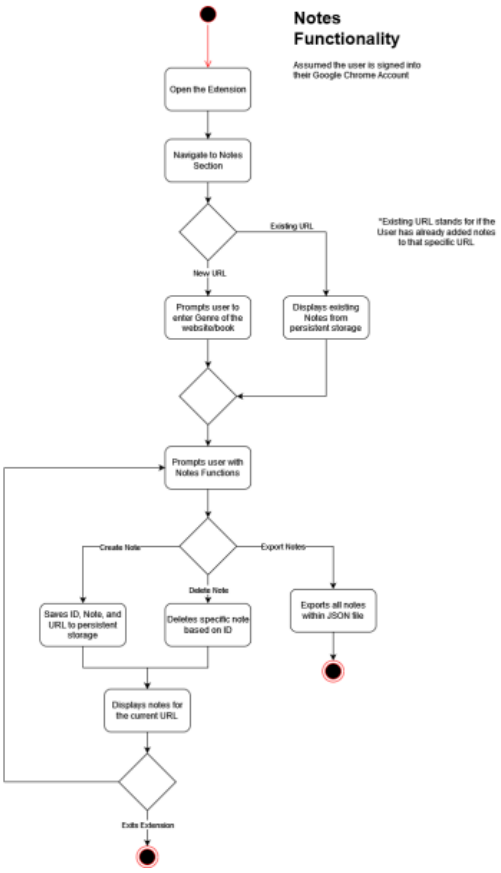
Our architecture is limited by the setup google requires for chrome extensions, but it is generally close to a MVC. For our model we used data from the page that the user has navigated to as well as the chrome storage API and the google API. The Google API is used to get information about a randomly chosen book for our book of the day and the storage API is used to store all the notes a user might create. For our view we used HTML, CSS, and js to create our extension pop as well as the overlay used to input and display notes. For the controllers event listeners are used inside js files to respond to user input such as clicking to add a note or click to scroll to a note.

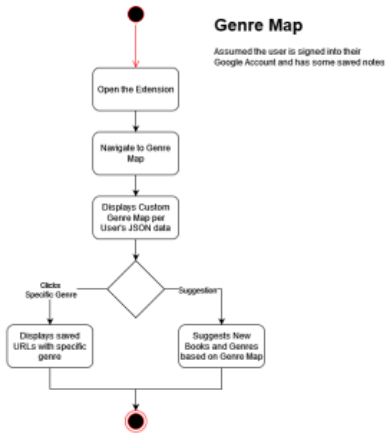
Static View

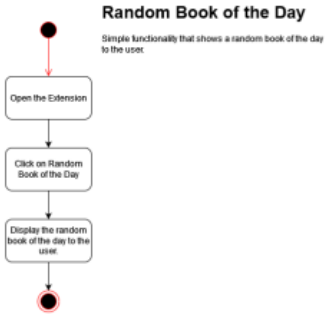


Our application does not use class as Google requires chrome extension to have most functionality in a content.js script. However, we do have a note object that we use inside multiple scripts which is created based on user input of title, and text content as well as taking the location on the page the user is at when adding the note and creating a unique id. This object is then stored in the chrome API by event listeners, which we have called controllers in the above diagram. These controllers are not in a separate class as they are simply in the same js file, but they do act as controllers and call functions that use a note as an input. There are functions to put a note on screen, delete all notes, and save a note. A single note also has event listeners tied to it that can delete it and scroll to its place on the screen.

Dynamic View







For the dynamic functionality of the website, it is split into three different main routes: the notes, genre mapping, and random book of the day.

When the user is on a webpage or book, the extension will then look and see if there are saved notes tied to the webpage already. If so, the extension will display these notes. After the notes are displayed, the user can either save new notes, delete previous notes, and then finally export their entire portfolio of notes. After the user has finished using the extension, they can exit by clicking the quit button on the top right.

For the Genre Map functionality, the user can navigate to the genre map for a custom map to be displayed on the page. This map will take in the user's custom JSON data including the genre of previously read books, and also basic information about each page. Users can then

take a deeper look into the map being provided with suggestions and also a detailed description of each specific genre when clicked.

The final major functionality of Notation Plus is the Random Book Of The Day. When the user clicks on the link, it will run a script that will randomly select a book from the Google Books API. The selected book will be displayed including the cover image, title, author, genre, and a link to the book page on Google Books.