Final Report

CS 4850 Fall 2023 Semester

SP - 17 Chrome Plug In

Prepared by Deja Hintzen, Josh Warren, Christian King

Professor Sharon Perry

11/29/23

https://christianking23.github.io/

https://github.com/ChristianKing23/SP17-PlugIn

Table of Contents

1. Introduction

For our Senior Project, our team developed a user-friendly Google Chrome extension called Notelee. We aimed to enhance the functionality of Google Chrome with this project as well as develop our skills in software development. The goal of Notelee is to elevate the user's browsing experience by giving them a way to easily organize and create notes within the browser, where the notes will be secure.

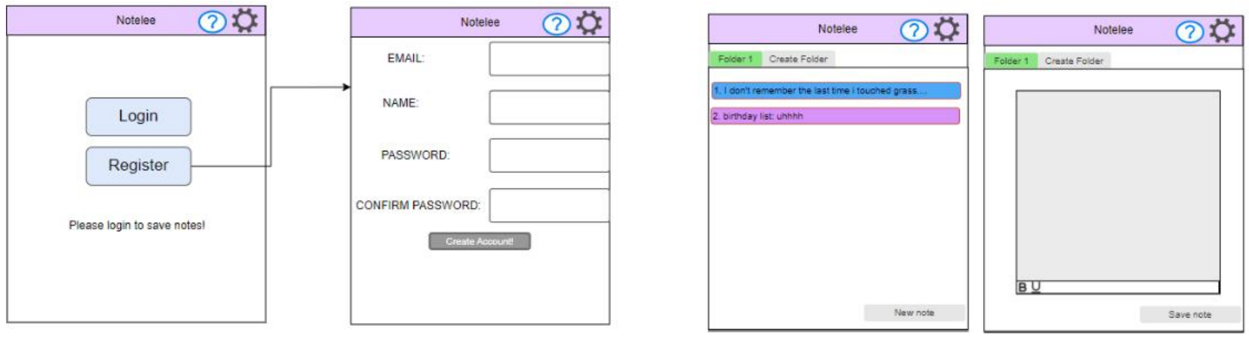
In this final report, we will dive into the details of Notelee, outline the key functionalities of the plugin, discuss the software design, a description of the version control that was used for the development of the plugin, a detailed development discussion that will include what we did and how we did it, the challenges that we had when developing Notelee, and the test plan and report. This report will highlight our team's collaborative efforts to create Notelee.

2. Software Design

2.1. Overview

Notelee is a Google Chrome extension that was designed to improve Chrome's functionality by enabling users to take notes through our user interface, either for/on a webpage or as generic notes. The functionalities include creating/editing/deleting notes and grouping notes into folders. The core design strategy is to use HTML with stylesheets as the frontend and Javascript as the backend, together with a database system powered by Firebase.

Using a manifest.json file, we can load the Notelee folder in Google Chrome and add it as an extension. The frontend then enables a popup with which the user can engage in a variety of ways, all utilizing HTML and CSS through the user's web browser. Our user interface allows the user to create, edit, and delete items, while our backend database, which uses Firebase, continuously saves the changes made. Additionally, the backend is running javascript while using event listeners and handlers. Our frontend user interface and our backend logic coding can communicate with each other through HTTP requests.



2.2. Manifest.json

A manifest file is needed for every Google Chrome extension. It includes the current version of Notelee and sets the default popup as popup.html and titles as 'Notelee.' When a user clicks on the extension icon, 'popup.html' is opened and will give them the Notelee user interface.

2.3. Popup.html

Popup.html is the default html file to be displayed when opening Notelee. It features a pink topbar with the plugin's name and tutorial icon. Beneath are two buttons meant to login to Notelee and Registering for an account, along with a welcome message. The functions for popup.html are held inside of accountManagement.js.We wanted to make sure that the structure of the html was as seamless as possible while also looking appealing to the user.

2.4. Background.js

Whenever the extension is loaded, background.js uses chrome storage to get any userdata from chrome storage to find an active user. The window.onload function in background.js auto-logs in the last user that logged in from popup.html by searching for the userdata's status. If the status says "active", popup.html is skipped to load the folder notes automatically for the user's first folder. Otherwise, if the userdata does not exist, or the status is shown as "inactive" or blank, the plugin will stay at popup.html for the user to manually register or log in once more.

2.5. accountManagement.js

By implementing various firebase functions, accountManagement.js handles all account managing functions for the users along with the tutorial and manage account buttons inside of the top bar for Notelee. First, event listeners are added to both the login and register buttons in popup.html. When clicked, the user will be prompted to enter in text for the email and password fields.

If the user clicks register, register() will check that the confirmed password matches the password and that the password is 6 or more characters long. If the conditions are satisfied, it calls createUserWithEmailAndPassword from firebase to create an authenticated user with that email and password into the system. Then, it uses sendEmailVerification from firebase which takes the email the user entered and sends an verification email to the user. To add the user into the realtime database, it creates userData which is the user's id, their email, and four empty folders. Chrome storage is used to add the id, email, and status of the user into the specific chrome browser the user is on. Once the user is in the realtime database, the plugin is reloaded to the welcome menu of popup.html.

If the user clicks login, tryLogin() checks that the email and password fields are not empty. Once the condition is satisfied, it uses the firebase function signInWithEmailAndPassword to search through the authenticated users for the email and password combination. If something does not match, an alert will show saying that the user entered invalid credentials. Next, it gets a reference of the user in the database using the user's id. If it exists, the user will be checked for verification. An email is only verified if the link in the email verification from register() has been clicked. A verified user will have its chrome user data changed to "active" and will be taken to the folder's screen to view their available folders.

Inside of the login menu is a "Forgot Password" button. When clicked, it calls the function sendPasswordReset() which lets the user input the email they forgot the password for and a "Reset Password" button to use sendPasswordResetEmail from firebase to have an email sent. This email will allow them to enter a new password and Notelee will be reloaded to popup.html.

This javascript file also handles the manage account menu, which is represented as the person icon in the top right of Notelee, only after logging in. When the icon is clicked, it will show a new display under the top bar to show the account management menu. Here, users will have three options: to log out, to delete their account, or to go back to viewing their folders.

- The log out button will set their user data in chrome to inactive, alert that they have successfully signed off, and reload the plugin.
- The delete account button will show a new display asking them to confirm that they want to delete their account. Two options will be displayed: a "Yes" button and a "No" button. "No" will send them back to view their folders, but "Yes" will delete their data from the realtime database, delete them as an authenticated user, alert that all of their information has been deleted, and reload from popup.html.
- Pressing the "Back to folders" button sends them to view their folders in Notelee.

2.6. Folders.js

The purpose of the folders.js is to switch between folders and load folder content. Functions for this folder can only be used once the user has successfully logged in. Logging into Notelee successfully calls loadFolderEventListener(string, userId), which gets all the buttons in the folder navigation bar and adds several event listeners to all 4:

1. When hovered, make the border pink.
2. When not hovered, make the border black.
3. When clicked, call loadFolderNotes(folderID, userId) to get the notes from firebase.

Additionally, this function loads the manage account button event listener because the manage account button should only work after logging in, along with the tutorial button. Both of those functions are imported from accountManagement.js. showTutorial() creates a popup window with the tutorial for Notelee as a web page, while manageAccountEventListener(userId) loads the manage account display and menu mentioned in section 2.5.

Function loadFolderNotes(folderID, userId) checks the folder for notes through firebase by getting a reference for that folder in the database. If the folder has data, then make the note list for the folder. If there is no data for the folder, then create an empty note screen.

A folder that has data is sent to the function makeNoteList(folderNum, notes, userId). After finding the folder in firebase, the note is found in the user's database and the id for the note is saved into an array. Every note is then separated to get the title of the note, which is used as the text content of the note's button in the note list table. Next to the column of notes, the note list also has a column of delete buttons which use the note's id to delete the note when clicked. After the note list has successfully been added to Notelee, it adds a unique "Create Note" button which will allow the users to create notes. The functions for creating, deleting, and editing a note are all handled in Notes.js.

If loadFolderNotes finds that the folder has no data, then it calls the function loadFolderScreen(folderNum, userid) which will display a “No Notes” screen for the user along with a “create note” button.

2.7. Notes.js

The notes javascript file is responsible for the actual note taking ability in Notelee. It is responsible for creating, editing, viewing, and deleting notes. Functions newNote(folderNum, userID) and saveReloadNote(folderNum, userId, matchingNoteId) both remove the current display of Notelee and providea note taking display to input a note title and text of the note. newNote makes the title and text area empty, while loadNote inputs the title and text into the input fields so the user can view the note they’ve previously created and edit any content within the note.

Saving a new note for a folder calls saveNote(folderNum, userId), which saves the note under the user’s folder in firebase to allow folders.js to add it into the notelist. However, saving edits to a note calls saveReloadNote(folderNum, userId, matchingNoteId), which replaces the original note content in the database with the new edits.

Deleting any note requires pressing the “x” button to the right of the respective note. By calling deleteNote(deleteNoteButtonID, userID, folderNum), clicking the delete button presents a message asking the user if they are sure they want to delete the note. By pressing no, the user is sent back to view their folders. By pressing yes, the note is deleted from their folder in firebase and the user is sent back to view the folder.

2.8. Firebase Files

The firebase files are responsible for connecting the extension to the Firebase service. They are responsible for initializing the Firebase app, interaction with the Firebase Realtime Database and Firebase references. The files include: firebase-app.js, firebase-auth.js and firebase-database.js. Firebase-main.js is given from firebase to set up our firebase project into our folder. It features the firebase configuration for the Notelee realtime database, which allows for communication between Notelee and the database for reading and writing.

3. Version Control

Version control played a big role in the Software Development Lifecycle of our Chrome plugin, Notelee. It provided us with a framework to collaborate about code management and track our progress throughout the semester. We used GitHub as our version control tool to organize and develop our project.

With the use of Github, we were able to seamlessly collaborate and update the code as we progressed through the development of the extension. We were able to track every modification made to the code through GitHub and can see the history of changes made and identify the major implementations that were made. Version control is also very pivotal for maintaining the different versions of Notelee. We can see each major point of development throughout the semester. Also, if there were issues, we were able to identify them easily.

4. Development Discussion

In this section, we will discuss the details of Notelee by discussing the process, the decisions made, any challenges we faced, and how we implemented our ideas.

4.1. Introduction and Overview

For our senior project this semester, our team decided to create a Google Chrome extension. We discussed what our strengths and weaknesses were such as if we were strong coders or if we were better with documentation. The two main purposes of this project was to put all of our skills that we have learned over our college careers into a project and get experience with working as a team to complete a project.

The main objective was to follow the Software Development Life Cycle when completing this project. With each step of the life cycle, we completed many documents. The Project Plan document set up what we wanted to do for our project, how we planned on communicating, the goals and deliverables we needed to accomplish, and our choice of version control. The course deliverables included the Project Plan, Software Requirements Specification, Software Design Document, Software Test Plan, Prototype Presentation, and the CS Department Website. The next document that we created was the Software Requirements Specification. With this document, we were able to provide the foundation for the architecture of the software as well as the design. We defined the software requirements needed to create the Chrome extension as well as the overall description of Notelee, the product functions that we wanted to include, the major features, and the security requirements. We also had the Software Design Specification document which discussed the low and high-level system architecture of Notelee. In that document, we detailed how the system will interact.

As we continued throughout the semester, we took all of this information about what we needed in order to complete this progress and implemented it into Notelee. When creating Notelee, there were many things that we were able to accomplish from the list of functions that we wanted to include. There were also many things that we were not able to include for the time that we had to complete the project. We took everything we were able to do and created a prototype presentation to present that included the progress that we had so far.

For the conclusion of this senior project, we were tasked to create a CS Department webpage that would hold all of our work that we did this semester. On the website, we needed to include documents that were already mentioned, the final report, the prototype presentation, and the source code of all of the files we made to create the extension.

This project took a great deal of collaborative effort and time management. We planned how we should organize to complete this project and how to meet all of the requirements within the time we were given. In this discussion, we will discuss how we decided on what we needed and how we were able to accomplish what we needed.

4.2. Features

Starting from the very first meetings as a team, we needed to decide on what we wanted to accomplish with our Chrome extension. We had a few ideas on what type of Chrome extension we wanted to create and ultimately decided on an extension that works on the Google Chrome web browser that would allow users to take notes within the web browser. We then discussed what are some of the features we wanted to make sure we had. A big detail that was important to us was making our extension stand out in comparison to other note taking applications. We also had to keep in mind what was possible to do in the scope of our class and the time limit we were given to do this project.

The major feature that needed to be implemented was the actual ability to create notes. For this feature, we wanted the user to be able to create notes quickly and efficiently. The desired functionality was when the user wants to create a new note, they would click create note. A note page will pop up which will allow them to create a note of any length, using different font options such as font size, font color, and heading. Each note that is created will be able to be drawn on, highlight, add pictures, links, etc. Notes can also be archived. We were able to implement most of the desired functions, but some such as the font size and color were unable to be implemented.

The next feature that we discussed that we wanted was folder creation. With folder creation, the user would be able to create different folders within Notelee and sort their notes in the different folder created. The desired function was on the home page, there would be an option for the user to create a new folder. Existing notes can also be sorted into these folders. Titles can be edited and the folder will have different customization options such as colors. Folders can also be archived. The implementation of folders that we completed was different from what we desired as in our implementation, there would be only four set folders.

In that same discussion, we discussed how users would be able to login and access their notes. We originally wanted the user to login into Notelee with their Google account. The desired function was when the user opens Notelee, they will receive a popup asking to connect their Google account with the extension. Instead, we used Firebase as the way to register any email as long as it was not already registered to another user. The user is able to login with the email and password. Also everytime the extension is closed and opened again, it opens for the last user that was logged in, so there is no need for the user to login every single time that Notelee is closed.

Some of the other desired features and functions that we wanted to include, but ultimately did not was webpage markup. With this feature, the user would be able to markup the current webpage they were on by drawing on the page and saving their edits within Notelee. When the user goes to create a note for the current webpage they are on, there will be an option to markup on the page. They can draw on the page and highlight content on the page. Everytime that the user opens that web page in the future, all of their edits will be there. This function, however, would have been very complex and difficult to implement.

For the security requirements, we wanted to make Notelee as secure as possible. In our discussion, we wanted to ensure that users will only be able to be logged into Notelee through the Google Chrome browser. Other webpages will not be able to see any notes made by the user. Notelee does not keep track of information entered on other pages. For example, if the user enters their debit card information onto Amazon, Notelee will not be able to read this information and store it.

4.3. Design and Architecture

When planning the design and architecture of the extension, important decisions had to be made that would affect the system. When creating a Google Chrome extension, we had to adhere to the programming requirements for Chrome plugins. These requirements included only using specific programming languages as well as how well we should code in Javascript, such as no inline event handlers, resulting in us to code differently than normal. Also, Chrome requires us to only utilize HTML and CSS for the plugins interface.

There were other things that we needed to consider before we began the production. For one, since this was a Google Chrome extension, no other web browser such as Opera or Microsoft Edge would not be considered when developing. Next, other than javascript for the logic coding and event handling for HTML5 user interface, no other programming languages would be utilized. Also, in order for data to be transported through Chrome, a manifest.json is required.

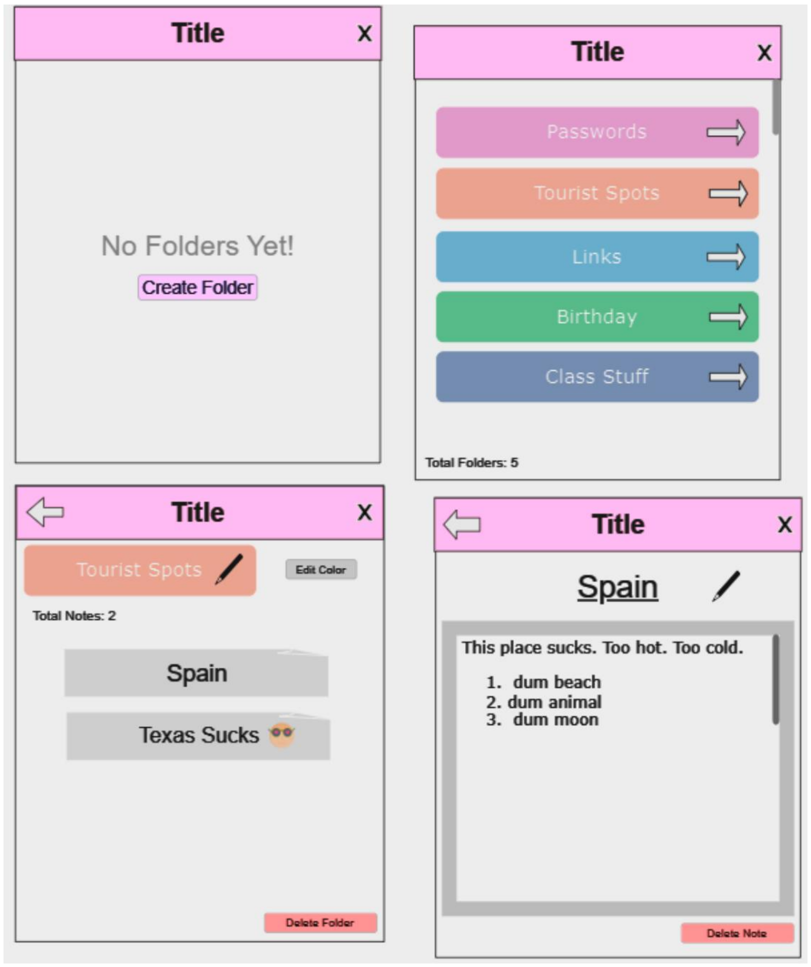
As far as performance was concerned, we wanted the extension to work fluidly and without latency as user data and usage grows. Also, If the user were to have slow or no internet access, we wanted Notelee to still work on creating notes and folders. If not, we would require error handling.

For the actual implementation of the extension, using HTML, we wanted to create a popup window for Notelee which would enable the user to interact with and include various buttons and text areas. With CSS, we needed to visually style Notelee by changing the color, margins, alignments, and more. In order for our extension to actually be useful, it would need to save all of the user information such as their login and notes created. We needed to create a database where we would code in ways to add and retrieve information from it.

4.4. User Interface

When discussing what we wanted for the user interface, our main goal was to make Notelee as user-friendly as possible. We wanted to make sure the extension was easy to use for the user, not confusing to understand, and intuitive. When discussing how we wanted Notelee to look, we decided on vibrant, pastel colors. For the home page, we wanted to give the user the ability to create a folder. Each folder would be a different color. Within each folder, we originally wanted the user to be able to name the folder and edit the color of the folder. The folder would contain each note and the user would also have the ability to delete the folder if they wanted. Below is the mockup of what we wanted the user interface to look like:

4.4.1. Original Design



This place sucks. Too hot. Too cold.

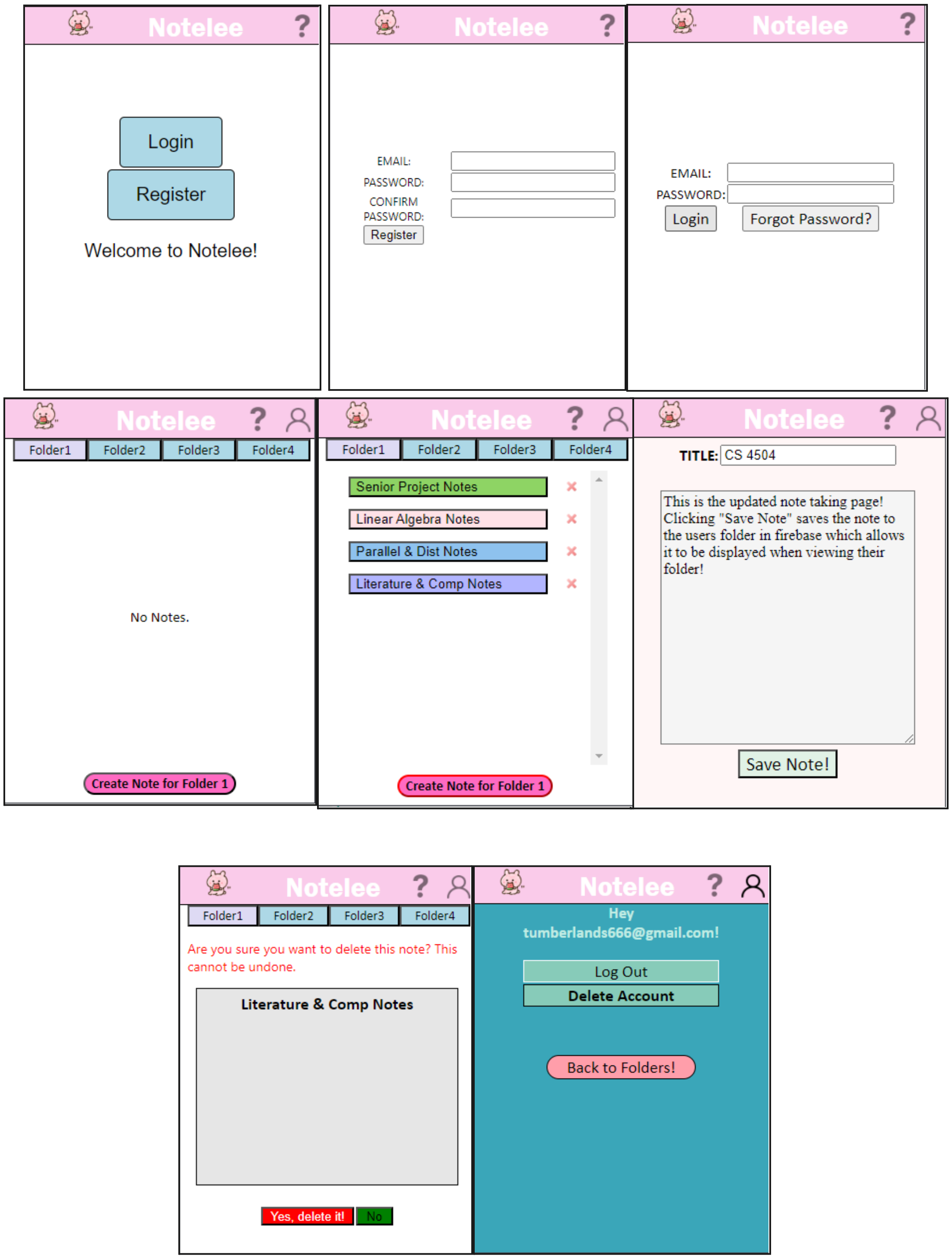
1. dum beach
2. dum animal
3. dum moon

For the final user interface, we stuck with the same vibrant, pastel color scheme. On the opening popup, you are greeted with the options to login or register. When you click login, you put in your email and password. If your information is in the database, you will be brought to the page with your folders, if not, you will have to try again. If you click on the register option, you will have to register an account. When you register an account, you will have to verify your email address with an email sent to the user.

On the main page, the user will see the main screen where folders and notes are within each folder. When the user hovers over one of the folders with their mouse, the border of that box will turn orange and when the user hovers one of the notes, the border of that box will turn yellow. The user will know which folder they are in because the folder box will turn a bright purple. Each note in the folder is a different pastel color with the option to edit, delete, and create a note for the folder. If the user clicks on a note or creates a new note, they can edit the title, edit the content of the note, and save it. In the current version of the extension, we were unable to add the ability for the user to bold, italicize, and underline the contents of each note.

Finally, when the user clicks on the user icon on the top right of the extension, there will be the option to log out and delete the account. When the user clicks to log out of their account, a popup from the browser will appear and alert the user that they have been signed out. If the user chooses to delete their account, they will get the warning that if they delete their account all of their notes will be deleted forever. If they select no, they will be directed back to the main page with the folders and the notes. If they select yes to delete their account, the user will get a popup from the browser saying their account has been deleted and they will be directed back to the login and register page.

4.4.2. Final User Interface



This is the updated note taking page! Clicking "Save Note" saves the note to the users folder in firebase which allows it to be displayed when viewing their folder!

Are you sure you want to delete this note? This cannot be undone.

5. Challenges

The main challenges that we faced during the development of our project was the registration of an account, login, and saving notes. With the very versions of Notelee, you would be able to login or create an account and create an account. However, since there was no database, everytime the user exited the extension, everything the user had done in that session would be erased. To ensure we were able to implement a way to store all of this information, we would use Firebase.

Firebase access is required for the plugin to connect to and interact with the database in order to save and retrieve data from Notelee. Using Firebase provided a solution for data persistence. The implementation of Firebase ensures that the user data would be stored and could be accessed across multiple sessions.

6. Test Plan and Report

The plan to test Notelee was to divide testing into 3 categories:

1. Logging in, registering, signing off, deleting account (All test case IDs that begin with 1)
2. Viewing folders, creating notes, editing notes, deleting notes (All test case IDs that begin with 3)
3. The tutorial (All test case IDs that begin with 2)

Within these 3 categories, we first thought about what *should* happen with each interaction in Notelee, including when purposefully entering in wrong information, and took note of whether or not the expectation happened.

Test Case ID	Test Case Description	Test Steps	Preconditions	Expected result	Actual Result	P / F
1AutoLogin	User is auto-logged in if they never logged off.	Click the Notelee plugin icon	A user signed in and never signed out.	Notelee skips the welcome menu and shows the user's folders and notes.	As expected	P
1LoginSuccessful	Authenticated users can log in.	Login with an correct email and password combination	user is already in firebase, user is verified	Shows the user's folders and notes	As expected	P
1LoginNotVerified	User logs in but is not verified.	Log in without verifying email.	Successfully registered but did not click the link in the verification	Shows an alert saying the user must verify their email before	As expected	P

			email sent to them.	continuing. Does not show folders.		
1LoginWrongEmail	User logs in with unrecognized email	Login with an incorrect email and correct password	Email is in firebase	Shows an alert saying the login credentials are invalid. Does not show folders.	As expected	P
1LoginWrongPassword	User logs in with wrong password and right email	Login with an email and password	Email is in firebase	Shows an alert saying the login credentials are invalid. Does not show folders.	As expected	P
1ForgotPassword	Send a password reset email	Click 'forgot password' from login menu, enter email, then click 'reset password'	Email is correctly formatted	Alert shows saying a password reset email has been sent. Send an email to the entered email. Plugin reloads.	As expected	P
1ChangedPassword	Password updates when changed from 1ForgotPassword	Click link in reset password email and enter in a new password	User with matching email is in firebase	User can log into Notelee with same email and new password	As expected	P
1RegisterSuccessful	Registering with an email and password is successful	Enter in a correctly formatted email and a password more than 6 characters	email does not match firebase	User is listed as an authenticated user and is in the database. An verification email is sent. Plugin reloads.	As expected	P
1RegisterIncorrectFormat	Registering with an email and password is unsuccessful	Register with either an incorrectly formatted email and/or password.	None	Receives an alert for each aspect that is not formatted correctly when "register" is clicked.	As expected	P
1LogOff	Logging off of Notelee	Click the person icon, click log off.	Already logged in.	Shows alert saying, "Logged off" and reloads Notelee.	As expected	P
1DeleteAccountDisplay	Displaying the delete account confirmation page.	Click the person icon, click delete account.	Already logged in.	Removes display. Adds an "are you sure" message in red along with a yes and no button with a light gray background.	As expected	P
1DeleteAccountYes	Deleting an account from the firebase from Notelee.	Click the person icon, click delete account, click yes.	Already logged in.	User is removed from the database and removed as an authorized user. Reloads Notelee.	As expected	P
1DeleteAccountNo	Choosing not to delete an account from firebase.	Click the person icon, click delete account, click no.	Already logged in.	Removes display, shows folder 1 display and makes the background white.	The background stays a light gray.	F

					Cannot travel between folders anymore.	
2TutorialPopup	Clicking the question mark icon opens a popup with a tutorial.	Click the question mark	Already logged in.	Popup Opens	As expected	P
2PopupPage	Clicking the plugin's icon opens popup.html with a login and register button.	Click the Notelee plugin icon.	Notelee source code is loaded as an unpacked extension in chrome.	Popup.html opens	As expected	P
3DefaultFolder	Autologin shows the notes for folder1.	Open notelee	User never signed out.	Notelee opens showing button folder1 with pink background, shows notes for folder, and a "create note" button for folder 1	As expected	P
3FolderButtonColors	Folder color changes when hovering and clicking	Hover over and click on a folder button.	Logged in.	Hovering a folder button turns the border pink. Not hovering turns the border black. Clicking makes the button pink and all other buttons blue.	As expected	P
3ChangingFolders	Clicking a folder button shows notes.	Click any folder button.	Logged in.	Clicking a folder removes the previous display and shows the note list for that folder or a text that says, "No notes." Shows a "create note" button for the specific folder.	As expected	P
3ShowNoteList	Clicking a folder that has notes shows the list of note titles.	Click on any note button from a folder that has notes.	Logged in and the current folder has notes.	Removes previous display and shows the table of created notes for the specific user with delete buttons.	As expected	P
3ShowNoNotes	Clicking a folder that has no notes shows text "No Notes."	Click on any folder that has no notes.	Logged in and the current folder has no notes.	Removes previous display and shows "No notes" text.	As expected	P

3DeleteNotesConfirmation	Deleting a note from a folder confirmation page.	Click on any folder that has notes, then click the "x" button.	Already logged in.	Removes previous displays and shows a confirmation screen with buttons "Yes" and "No".	As expected	P
3DeleteNoteNo	Choosing not to delete a note.	Click on any folder that has notes, click "x" button, click "no" button.	Already logged in.	Removes the delete confirmation page and shows the contents for folder 1.	As expected	P
3DeleteNoteYes	Choosing to delete a note.	Click on any folder that has notes, click "x" button, click "yes" button.	Already logged in.	Removes the delete confirmation page and shows the contents for folder 1 without the deleted note. Note data is also removed from the user's data in the database.	As expected	P
3CreateNewNote	Creating a new note for a folder.	Click on any folder, click the "Create Note" button.	Already logged in.	Removes display and adds an input box, text area, and save button.	As expected	P
3SaveNewNote	Saving a new note.	Click on any folder, click the "Create note" button, add a title and text, and click the save button.	Already logged in, boxes are not empty.	Adds note to the user's data in the database. Removes the display and shows the notes for the folder with the new note at the bottom.	As expected	P
3ViewNote	Viewing the contents of a note.	Click on any folder, click on any note button.	Already logged in, the folder has notes.	Removes display and adds an input box and text area that is filled in with the respective note, and a save button.	As expected	P
3SaveEdittedNote	Saving edits of a viewed note.	Click on any folder, click on any note button, change the contents of the title or text, click save.	Already logged in, the folder has notes.	Pressing save updates the note in the database and the note is placed back into the notelist in the same place as the original.	As expected	P
3HoveringButtons	Hovering buttons changes the color of its background or its border.	Hover over the folder buttons, note buttons, "x" buttons, create note button, save button, log out	Already logged in.	Hovering over a button changes the background color of the button or the border color. Not hovering puts it back to its original style.	All buttons change their display except the login/regi	F

		button, delete account button.			ster buttons in popup.ht ml.	

7. Conclusion

In conclusion, we created a functional Google Chrome Plug In which allows users to use the extension to login with the Chrome browser, create notes, and store those notes into folders. We can continue to improve the plug in to have more features that we were not able to implement at this time.

In this report, we introduced what we decided to create for our final project and the objectives and goals that we wanted to complete by the end of the semester. We then gave an overview of the system design which included the purpose of each of the files that we made to create Notelee. Next, we detailed our choice in version control then went into the narrative discussion of the entire process since the beginning of the semester, which discussed what we did and how we did it. Finally, we highlighted some of the challenges we faced during development and how we overcame those challenges and went into our test plan and report.

Since our project only lasts a semester, there were many features and functions that we were not able to implement. With everything that we did, those features can be implemented in the future if we decide we want to expand on the extension. Our development allows for expansion and so functions like webpage markup and note taking customization can be added to enhance this plugin.

8. Source Code

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/README.md

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/accountManagement.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/background.js https://github.com/ChristianKing23/SP17-PlugIn/blob/main/firebase-app.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/firebase-auth.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/firebase-database.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/firebase-main.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/folders.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/manifest.json

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/notelee-firebaseref.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/notes.js

https://github.com/ChristianKing23/SP17-PlugIn/blob/main/popup.js