Team Name: Team 626

Team Members and Emails: Hamza Awad hawad003@ucr.edu, Huzaifah Simjee hsimj001@ucr.edu, Qi Jie Guan qguan001@ucr.edu, Shreya Kumar skuma021@ucr.edu, Katherine Legaspi klega002@ucr.edu

Project: Google Keep

Communication Channel: WhatsApp

Tech Presentation: WordPress

Github: https://github.com/hsimj001/Team-626-Keep

Google Keep (Digital Notes)

Description: A web-based platform for note-taking developed by Google. On this web app, the user can write, edit, customize, and share their notes. The user can organize their notes by categorizing and archiving their notes. The notes are linked to the user's gmail account, so it can be accessed and modified everywhere. It can also be shared to other users using their gmail accounts for cooperative note taking.

1. **Google Authentication Account:** As a user, I would want to create login credentials to be able to utilize the websites functionality.
2. **User Account Login/Logout:** As a user, I would want to log in using my credentials and log out to keep others from manipulating my account details and notes.
3. **Homepage:** As a user, I want to visit the homepage where I can login, register, logout, create, delete, tag, archive, pin and share notes (UI). Notes should be displayed on the homepage where it can be organized by alphabetical and recency filters.
4. **Add/Delete Note**: As a user who wants to keep memory of information that I can access, I want to make a digital note. The note must include a subject and description. If a previously created note is no longer needed, I want to be able to discard unwanted digital notes.
5. **Share Note**: As a user who wants to keep memory of information that others and I can access, I want to share my digital note with others' emails.

6. **Pin note**: As a user who has a lot of notes, I want to keep my most important notes at the top of the page.
7. **Edit**: As a user who wants to revisit and modify a note, I want to make a change to a note that already exists.
8. **Archive**: As a user who is done with a note temporarily, I want to temporarily set that digital note aside from the page. I can restore digital notes that I archived back into the main page.
9. **Upload Image:** As a user, if necessary, I want to be able to add displaying images in my digital notes for effective note taking.
10. **Search By Tag:** As a user, I want to be able to identify and locate my notes by adding search tags.

Feature Design Document (User/Functional Reqs.)

Google Authentication Account

Purpose:

Users want to start using GoogleKeep through a simple authentication process.

Database dependencies:

To authenticate a user, the database needs to register the user associated to his/her email .

Technical Note:

Users need an existing email to create their login. They cannot use a registered email already in GoogleKeep database to create a new login.

Feature Result:

User can use an email to create login credential to be able to utilize GoogleKeep's functionality.

User Account Login/Logout

Purpose:

Users need a login and log out function to keep others from manipulating their account details and notes.

Database dependencies:

The database finds a match to the user's email in the users list. Then, it retrieves and secures notes associated with the user's ID and their login email.

Technical Note:

GoogleKeep needs to keep the user signed in on the browser unless the logout function is called.

Feature Result:

Users can log into GoogleKeep using their credentials and log out to hide their notes and account data from each other.

Homepage

Purpose:

Users needs a homepage where they can have readily access to GoogleKeep’s functions.

Database Dependencies:

User inputs are stored as data in different components in the database associated with the registered user.

Technical Note:

Notes should be displayed on the homepage where it can be organized by alphabetical and recency filters.

Feature result:

Users has a homepage where they can login, register, logout, create, delete, tag, archive, pin and share notes (UI).

Add/Delete Note

Purpose:

Users need to keep memory of information in a digital note that they can access.

Database dependencies:

The database stores the created note with the associated user ID and login email. The database removes the note selected by the user to be deleted and dissociates it with the user's ID and login email.

Technical Note:

The note must include a subject and description. If a previously created note is no longer needed, I want to be able to discard unwanted digital notes.

Feature result:

The user can successfully create a new note which shows up on the homepage. The user can successfully delete and existing note and it no longer shows up on their homepage.

Share note

Purpose:

Users want to be able to share digital notes with each other.

Database dependencies:

GoogleKeep parses the user and recipient's email input and directs to the database. The shared note's ID is now referenced to both the user and the recipient's ID.

Technical Note:

If a shared note is deleted, the note is removed for both the sender and recipient.

Feature result:.

The user is able to share and view notes with recipients through email.

Pin note

Purpose:

Users want to keep their most important notes at the top of the page.

Database dependencies:

Technical Note:

Feature result:

Edit

Purpose:

As a user who wants to revisit and modify a note, I want to make a change to a note that already exists.

Database dependencies:

The database needs to reflect the new changes that the user has made to a note. When a user chooses to edit a note, the database needs to find it in the user's associated notes and update its contents.

Technical Note:

The note must exist and once I make the changes I want, the database should update the note and store those changes.

Feature result:

The existing note's old content is replaced with the edited content that the user wanted and is reflected on the users homepage.

Archive

Purpose:

User wants to temporarily set a digital note aside from the main page.

Database dependencies:

An event is triggered by the user in which references to targeted note's ID in the database. Using the note's ID, the system updates the archive state of the note and sends it to the archive sub database.

Technical Note:

Archived notes need to be able to be restored back into the homepage.

Feature result:

The user can view archived notes in another page and restore it back to the homepage by clicking the archive button.

Upload Image

Purpose:

If necessary, users want to be able to add displaying images in my digital notes for effective note taking.

Database dependencies:

Technical Note:

Feature result:

The user can upload images from a finder and display them on the selected note.

Search By Tag

Purpose:

Users want to identify and locate their notes by adding search tags.

Database dependencies:

When a user adds tags, the database needs to find the particular note and store the associated tags with the specific note.

Technical Note:

The tags need to be stored as an element of the note so that the specific note can be filtered using the tags.

Feature result:

The user can view their notes grouped by the specific tags.

October 14th, 2019

The idea of the project is to create a web-app that, in its simplest form, users can save notes. To do this, we will need a way for the user to create an account, log into their account, add notes, and delete notes. Once we have this done, we will expand on it to add more functionality for the user. In the near future, we want to add support for pinning notes, editing notes, and archiving notes. We have to work on the basics first.

We will create a homepage where there will be a log-in and create an account using the Google sign-in provision on Firebase. Of course if the user is already logged in, there will be a logout button available for them to press. When a user creates a new account using a new username and a new password, the Firebase database gets updated with a new username and password. This will allow for easy login for the user. In addition to that, to make sure we keep the web-app secure, when the user wants to log in, we will use the OAuth library in Firebase to validate whether the user’s input matches an existing account or not. This will also work to make sure that the user is not a bot. This is crucial because we could end up using valuable database space for something that is not a user.

When the user wants to logout, we will be redirecting them to the home page. This is crucial to making sure the web-app is super easy for the user to use. We want to prioritize user interface in the entire app, because it is a notes app. Notes are used to make things easier on people and by making our UI easy, we can contribute to that. We believe this is crucial to getting users for the web-app.

October 28th, 2019

For our first sprint demonstration, we split user stories that were complex into smaller and more manageable tasks in order to achieve a starting point for our Google Keep-like project. One of our splits included our profile logic which used Firebase's Google Authentication in order to keep track of account data. We started by creating a template react app. Then we imported firebase into our app and pasted the helper functions under our app component. Currently, when our user successfully logs in, our app props retain references to *user*, *signOut*, and *signInWithGoogle*. *user* holds information on our user's public details (i.e. name). *signOut* and *signInWithGoogle* are functions which we call when our user respectively clicks on our Sign Out and Sign in with Google buttons.

When our users do interact with these functions, our react state changes, so react efficiently calls render on all of our components. And you can see this with our side navigation tabs updating to allow user interactions as well as their styles changing; this is also true for our current and basic profile card which appears in the middle of the screen to confirm the log in or to request a log in.

This current version of our app requires further compartmentalizing our code in App.js so that we can render our upcoming screens more efficiently and clearly. To start on our next task which is in dealing with note creating and saving, we would need to start reconstructing some of our code to build such future features.

November 11, 2019

So far we have had to make some adjustments to our user stories. Certain user stories overlapped with one another, such as filter by color and filter by subject, so we combined those. For this second sprint demonstration, we added the registration feature so that the user can create a new account if they don't have an existing gmail account.

We began this sprint by developing our homepage for when the user logs in. Over here the user has access to a drop down menu where the user can select to view all their notes, edit notes, archive the notes or trash the notes. The add notes functionality is taken care of in the composer.js file. Here the note can be given a title and a description. These notes are stored in a container which allows individual users to have access to their own individual notes. The delete notes functionality is implemented in our Note.js file in the *handleDelete()* function.

In this sprint, we also incorporated the ability for the user to filter their notes by when they were added and alphabetically. This is handled in the Note.js file with the *filterRecent()* and *filterAlphabetical()* functions. In these functions the date associated with the notes or the subject of the notes are compared and then force updated.

So far we have our UI established pretty well. The users notes are displayed neatly for the user to view. For our next sprint we want to make the archive note, edit note, share notes and pin note features functional.

December 2, 2019

For our third and final sprint we accomplished a significant amount of work.