Noteworthy

Team 8 - Design Document

Team Members: Adrik Herbert, Aidan Welch, Josh Kleyman, Quin Houck

Index

1. Purpose

One of the most powerful and culturally significant uses of the Internet is connecting with people through social media. As technology progresses, we're finding deeper ways to connect with others online. As of now, there are many different isolated options for sharing videos, text messages, images, audio clips, and other media. Interactions with other people are limited to these specifically-designed social media websites. We believe that we can harness the power of the web browser to open up the opportunity for interaction to every web page.

There are numerous extensions available online, such as Beanote or Note Anywhere, that allow users to take notes or draw images on top of websites. Most of these extensions keep those notes on the webpage when the site is reloaded or accessed at a different time. None of these extensions allow the notes to be shared with other users, nor do they have a centralized location to view all of the created notes easily. The extensions available are all primarily for a single user to better annotate a website and none of them include the social media aspect that we provide.

Through Noteworthy, we're effectively turning the Internet itself into a social medium, thereby expanding and strengthening the realm of possibilities for online social interaction. With Noteworthy, users will have a centralized site to keep track of their own personal annotations on websites as well as a method to share any type of online media with their friends or other groups. By creating Noteworthy, we hope to turn the Internet into a social bulletin board, enriching the online experience for every user.

2. Functional Requirements

2.1 User Accounts

1. As a user, I would like to be able to register for a Noteworthy account.
2. As a user, I would like to be able to login and manage my Noteworthy account.
3. As a user, I would like to be able to login to my Noteworthy account through Google.
4. As a user, I would like to be able to logout of my Noteworthy account.
5. As a user, I would like to be able to delete my Noteworthy account and its data.
6. As a user, I would like all of my data on my friends' accounts to be deleted when I delete my account.
7. As a user, I would like my password to be reset if I forget it.
8. As a user, I would like to login once and have access to both the central site and the extension.

2.2 Note Manipulation

9. As a user, I would like to be able to create notes on a website in my browser
10. As a user, I would like to be able to delete my notes
11. As a user, I would like to be able to place created notes at a specific location on a website
12. As a user, I would like to be able to delete my note collections
13. As a user, I would like to make my notes public or private
14. As a user, I would like to be able to form note collections
15. As a user, I would like to make my note collections public or private
16. As a user, I would like to be able to sort my notes (by date, name, website, etc.)
17. As a user, I would like to be able to toggle note visibility on and off

2.3 Note Access and Viewing

18. As a user, I would like to be able to access all of my notes
19. As a user, I would like to be sent to the specific website my note is on when it is clicked
20. As a user, I would like to be sent to the specific location of a note link on a website

2.4 Friends

21. As a user, I would like to be able to find and add friends
22. As a user, I would like to be able to remove friends
23. As a user, I would like to be able to see my friends' public notes
24. As a user, I would like to be able to see my friends' notes on websites

25. As a user, I would like to be able to react to my friends' notes (like, dislike, etc.)
26. As a user, I would like to comment on my friends' notes

2.5 Groups

27. As a user, I would like to form friend groups
28. As a user, I would like to rename my friend groups
29. As a user, I would like to add and delete collections from my friend groups
30. As a user, I would like to join other public groups
31. As a user, I would like to request to join private groups
32. As a user, I would like to be able to accept requests to join my private groups
33. As a user, I would like to be able to remove people from my groups
34. As a user, I would like to be able to delete my groups

2.6 Notifications

35. As a user, I would like to be able to receive notifications by email.
36. As a user, I would like to enable/disable email notifications.
37. As a developer, I would like to be able to send data from the central site to a user's email.

2.7 Miscellaneous

38. As a user, I would like to be able to have a nearly equivalent experience with the web app on my mobile device as my PC.

3. Non-Functional Requirements

3.1 Architecture and Performance

1. As a developer, I would like the extension to connect to the same database as the central website.
2. As a developer, I would like to develop the back end server using Flask.
3. As a developer, I would like to develop the front end in React.
4. As a developer, I would like my application to be search engine optimized.
5. As a developer, I would like my application to support many concurrent users.
6. As a developer, I would like my application to have robust error-handling.
7. As a developer, I would like to design my application so that scaling the application requires minimal renovation of the code base.

3.2 Appearance

1. As a developer, I would like to create an aesthetically-cohesive suite of applications with a simple and memorable design.

3.3 Security

1. As a developer, I would like to hash the passwords for better security.
2. As a developer, I would like to prevent SQL injection.
3. As a developer, I would like user data to be secured during transmission.

3.4 Server

1. As a developer, I would like for the server to be able to efficiently communicate with a database to store user data.
2. As a developer, I would like for the server to be able to pass information back and forth between the database and the browser extension in real time.
3. As a developer, I would like to implement my server functionality using the principles of RESTful APIs.

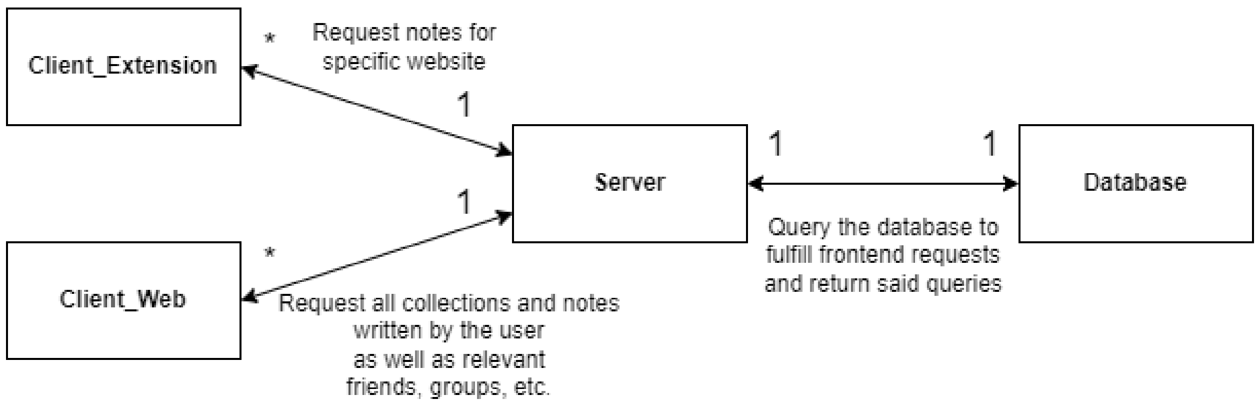
3.5 Hosting/Deployment

1. As a developer, I would like to host my server on an AWS EC2 instance.

4. Design Outline

4.1 High Level Overview

Noteworthy is a web application and browser extension to create and share notes. Our application will use the client-server model with one backend server using the Flask framework in Python. This server will be responsible for communicating with the database as well as the business logic for our client.

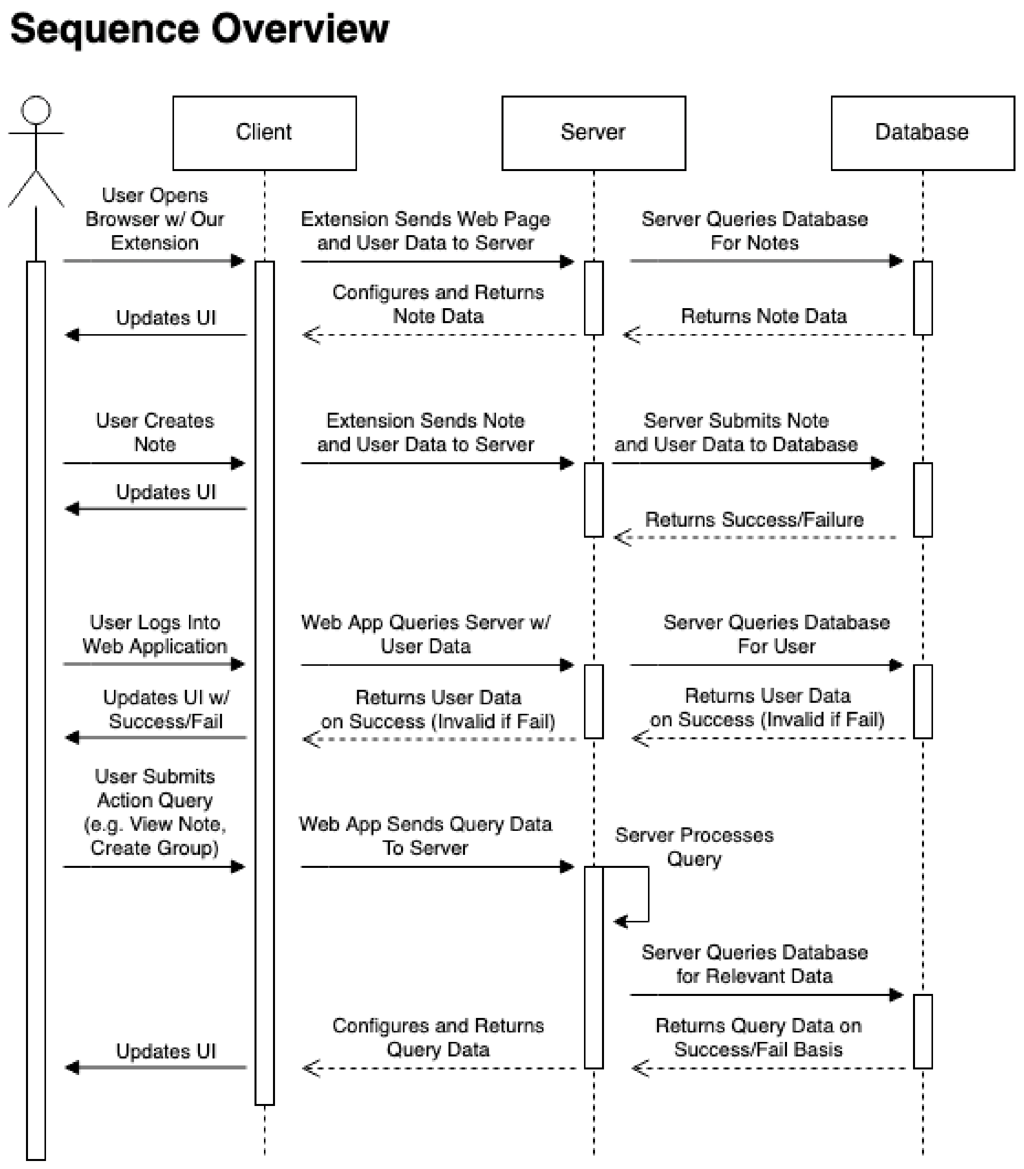

Client_Extension
*
Request notes for specific website
1
Server
1
1
Database
1
Client_Web
*
Query the database to fulfill frontend requests and return said queries
Request all collections and notes written by the user as well as relevant friends, groups, etc.

1. Client
    a. Client provides a user interface with our system
    b. We will have separate interfaces for the web app and the extension
    c. Client sends and receives API requests to the server to then render updates received
    d. Client Web:
        i. Centralized location for all created notes
        ii. Collection system for organization
        iii. Add friends and create groups
    e. Client Extension:
        i. Browser extension
        ii. Specific notes on a per-website basis
        iii. Interface is for creating notes on the website and (if desired) viewing friends' notes
2. Server
    a. Server handles requests from the client and makes queries to and from the database
    b. Server responds appropriately to the respective client
    c. Server validates requests

3. Database
    a. Relational database to store information
    b. Sends information to the server when queried

4.2 High Level Sequence Diagram

The diagram below shows the general sequence of events a user will take when using Noteworthy. Since Noteworthy is divided into two main sequences (which are technically two different clients), we combined both sequences into one client for the purposes of demonstrating high-level interaction with the system. The structure shown below illustrates a basic client-server-database implementation, including some functionality specific to our implementation.


Sequence Overview
Client
Server
Database
User Opens Browser w/ Our Extension
Extension Sends Web Page and User Data to Server
Server Queries Database For Notes
Updates UI
Configures and Returns Note Data
Returns Note Data
User Creates Note
Extension Sends Note and User Data to Server
Server Submits Note and User Data to Database
Updates UI
Returns Success/Failure
User Logs Into Web Application
Web App Queries Server w/ User Data
Server Queries Database For User
Updates UI w/ Success/Fail
Returns User Data on Success (Invalid if Fail)
Returns User Data on Success (Invalid if Fail)
User Submits Action Query (e.g. View Note, Create Group)
Web App Sends Query Data To Server
Server Processes Query
Server Queries Database for Relevant Data
Updates UI
Configures and Returns Query Data
Returns Query Data on Success/Fail Basis

5. Design Issues

5.1 Functional Issues

1. What information do users need for creating an account?
    - Option 1: Username and password only
    - Option 2: Username, password, email address
    - Option 3: Username, password, email address, phone number

- Decision: Option 2
- Justification: When setting up an account, an email is absolutely required to make sure no two accounts are using the same email. The email will also be used as well as a password to login to an account. A username will be required and used as a display name for the user. A phone number would only be useful for a two-factor authentication which we don't think we need.

2. Should there be a character limit for notes? If so, how long?
    - Option 1: Unlimited
    - Option 2: 255
    - Option 3: 1000

- Decision: Option 3
- Justification: While we would love to say that notes can be unlimited in size to allow the user more freedom, it doesn't make sense storage wise for our database or for any database. When looking at other sites, many restrict the character limit for text to be 255 due to ASCII limitations on past databases; we shouldn't have that problem and 255 characters seems quite small to express a full idea or to get the most out of a singular note. Because of this, we wanted a character limit that would allow a user to be able to express everything they wanted to but also not be too long. We settled on 1000 characters as it is slightly larger than the length of a large paragraph.

3. What is the main purpose of our website?
    - Option 1: Educational/Research
    - Option 2: Social Media
    - Option 3: Both

- Decision: Option 3
- Justification: While we first set out to create Noteworthy as a social media app, we believe that it can grow far beyond that as a tool for educational/research purposes as well. Being able to accommodate the wants of all our users will allow a more diverse and expanded user base. Even though some users may want to enjoy Noteworthy as a social media application with their friends, we wouldn't want to limit anyone's potential if they desire to use our app for educational goals. Therefore, we chose to design our application with both options in mind.

4. Should groups have administrators and regular users?
    - Option 1: Users in groups can have admin privileges or regular privileges
    - Option 2: All users in a group have the same privileges

- Decision: Option 2
- Justification: We first thought we would implement administrators or moderators to different groups to only allow some users to invite new members, remove members, delete group notes, etc, but we decided that groups should have an environment where everyone is equal as groups should be formed between users one is familiar with, such as friends or study groups. We don't intend for the group feature to be used with a large group of users that one doesn't know and trust.

5.2 Non-Functional Issues

1. What web service should we use?
    - Option 1: AWS
    - Option 2: Heroku
    - Option 3: Google Cloud
    - Option 4: Azure
- Decision: AWS
- Justification: Heroku is no longer a free service and knowledge of AWS is valuable for future endeavors. AWS is well documented and has specific tools for deploying web apps. The reason we chose AWS over Google Cloud or Azure is because we have some familiarity with AWS's EC2 instances. We will likely not be using any other features of any of those platforms. We don't need high-performance machines, we need something that's quick and dirty. Considering our previous experience as well, AWS EC2 seems to be a good choice for us. AWS EC2 also supports VM hibernation, whereas Azure and Google Cloud do not.

2. What backend language/framework should we use?
    - Option 1: Python/Flask
    - Option 2: JavaScript/Node
    - Option 3: Python/Django
    - Option 4: Java/Spring Boot
- Decision: Python/Flask
- Justification: Two of our group members had experience with Python and minor experience with Flask. We wanted to learn more about using Python and Flask as well. There is good official documentation and good examples of best practices. We chose Flask over Django because Flask is more lightweight and easier to learn. Flask is closer to writing normal Python, whereas Django

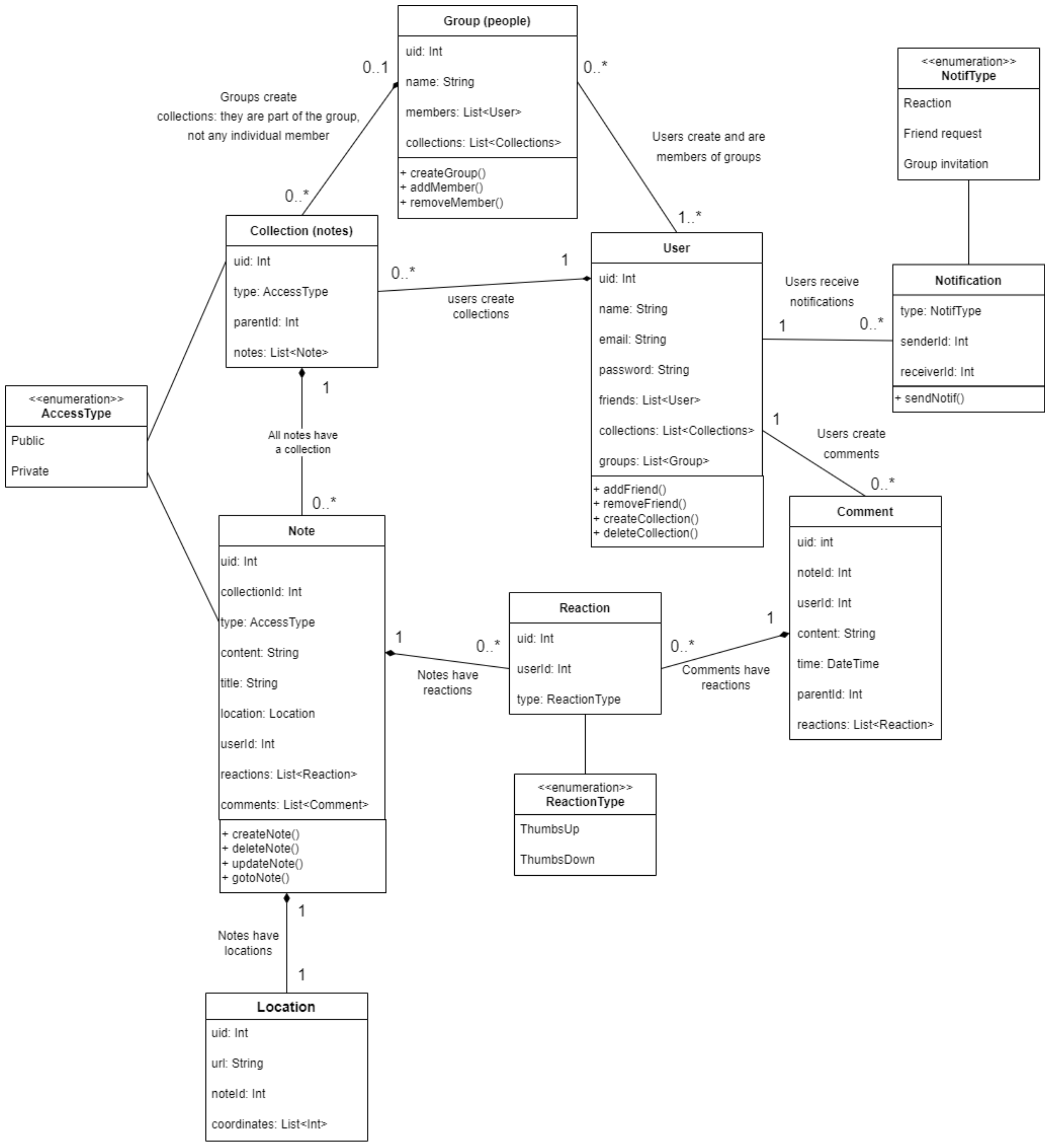
has a lot of custom things required. We are going to be writing a backend API, so we don't need too many fancy things nor do we need much fullstack support.

3. What frontend language/framework should we use?
    - Option 1: React
    - Option 2: Angular
    - Option 3: Vue
- Decision: React
- Justification: Two of our group members have prior experience working in React. React is a widely documented framework with many public libraries and tutorials online to use. While React is an excellent tool for web development, it can also be used to make our chrome extension, which will help keep consistency regarding frameworks between the two frontend components of our project.

4. What database should we use?
    - Option 1: MySQL/PostgreSQL
    - Option 2: MongoDB
    - Option 3: Neo4j
- Decision: PostgreSQL
- Justification: Our group members have the most experience with relational databases. A relational database also makes sense for this style of web app, as ownership relations between notes, users, groups, etc. must all be established. Our web app does not have any high-speed aggregation requirements or other such data manipulation. Neo4j may be a viable option, but only one group member has even heard of it so we decided that wasn't the best way to go.

6. Design Details

6.1 Class Design



6.2 Descriptions of Classes and Interaction between Classes

Each class has a list of fields that represent the values that are stored for each of its objects.

- User
  - The user object is created when someone creates a Noteworthy account.
  - Each user will have a unique ID.
  - A user will have an Email and Password for logging in.
  - A user will have a Name for display purposes.
  - Each user will have a list of users that designates their friends on the platform. Users will be able to add and remove their own friends as they wish.
  - Each user will have a list of groups that they are a part of. The groups will be displayed on the side navigation bar and on the user's homepage.
  - Each user will have a list of the different collections of notes that are theirs personally. Each user will have a general collection of notes initially.
- Group
  - A group object is created when a user decides to make a new group.
  - Each group will have a unique ID.
  - Each group will have a Group Name for display purposes.
  - Each group will have a list of the users that are in the group. New users are added to the group through an invitation by a user already in the group.
  - Each group will have a list of the different collections of notes that anyone in the group may view and edit.
- Note
  - A note object is created when a user creates a note through use of the extension on a website.
  - Each note will have a unique ID.

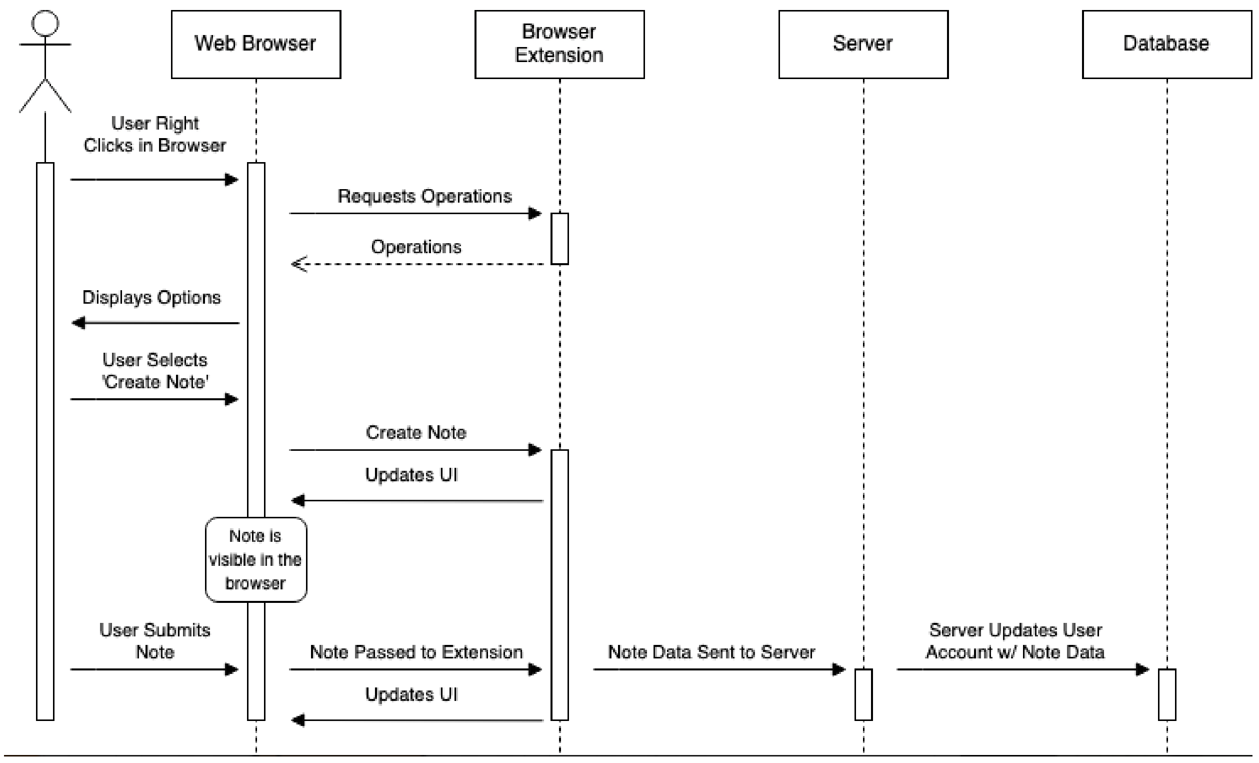
- Each note will store the ID of the user that created it.
- Each note will store the ID of the collection it belongs to.
- Each note will have a boolean field that dictates whether the note is private or public. The publicity of a note can be toggled on and off at any time.
- Each note will contain a String field that represents the text of the note written by the user.
- Each note will have a title where users can write the general idea of the note.
- Each note will have a location field representing a location object of where on the internet the note is located.
- Each note will have a list of all of the comments that were made on that note.
- Each note will have a list of all of the reactions made by other users on that note.

- Collection
    - A general collection object is created when a new user is created or when a new group is created.
    - A user can create new collection objects for groups they are in or for their personal accounts.
    - Each collection will have a unique collection ID.
    - Each collection will have a Name for display purposes.
    - Each collection will have a list of sub collections
    - Each collection will store the ID of the parent collection it exists in. This value will be 0 if the collection has no parent.
    - Each collection will have a list of the notes that are stored in that collection.
- Location
    - A location object is created when a note is created using the extension on a website.
    - Each location object will have a unique ID.
    - Each location object will store the URL of the website the note is on.

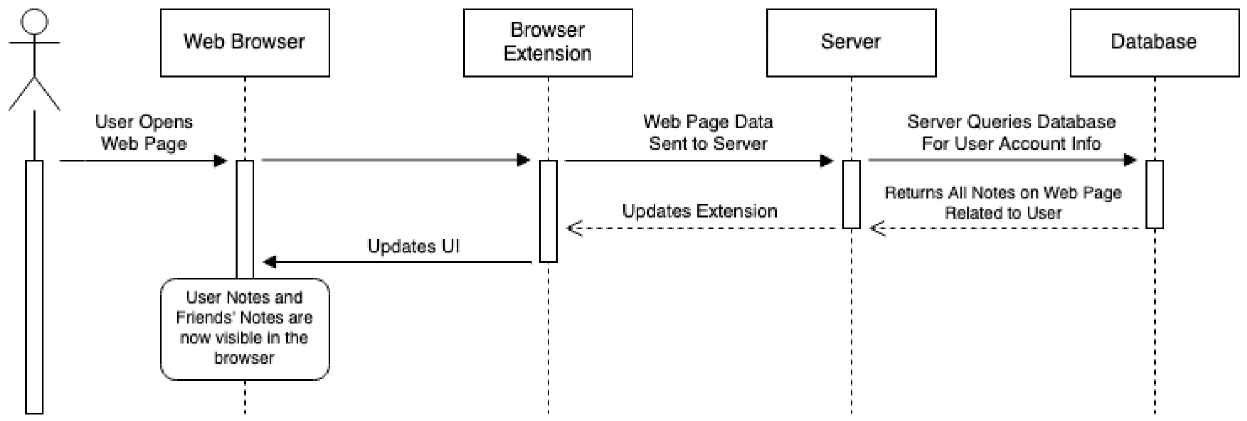
  - Each location object will store the coordinates on the webpage where the note resides
- Comment
  - A comment object is created when a user decides to comment on another user's note.
  - Each comment will have its own unique ID.
  - Each comment will store the ID of the user who made the comment.
  - Each comment will have a text field that represents the text of the comment.
  - Each comment will store the date and time it was created.
  - Each comment will have a list of the reactions made by other users to that comment.
- Reaction
  - A reaction object is created when a user wants to react to another user's note or comment.
  - Each reaction will have its own unique ID.
  - Each reaction will store the ID of the user who made it.
  - Each reaction will store the type of reaction it is.
- Notification
  - A notification object is created when a user comments on a post, reacts to a post or comment, or a user sends an invitation to another user to join a group.
  - Each notification object will have its own unique ID.
  - Each notification object will store the IDs of the sender user and the receiver user.
  - Each notification will store the type of notification it is.

6.3 Sequence Diagrams

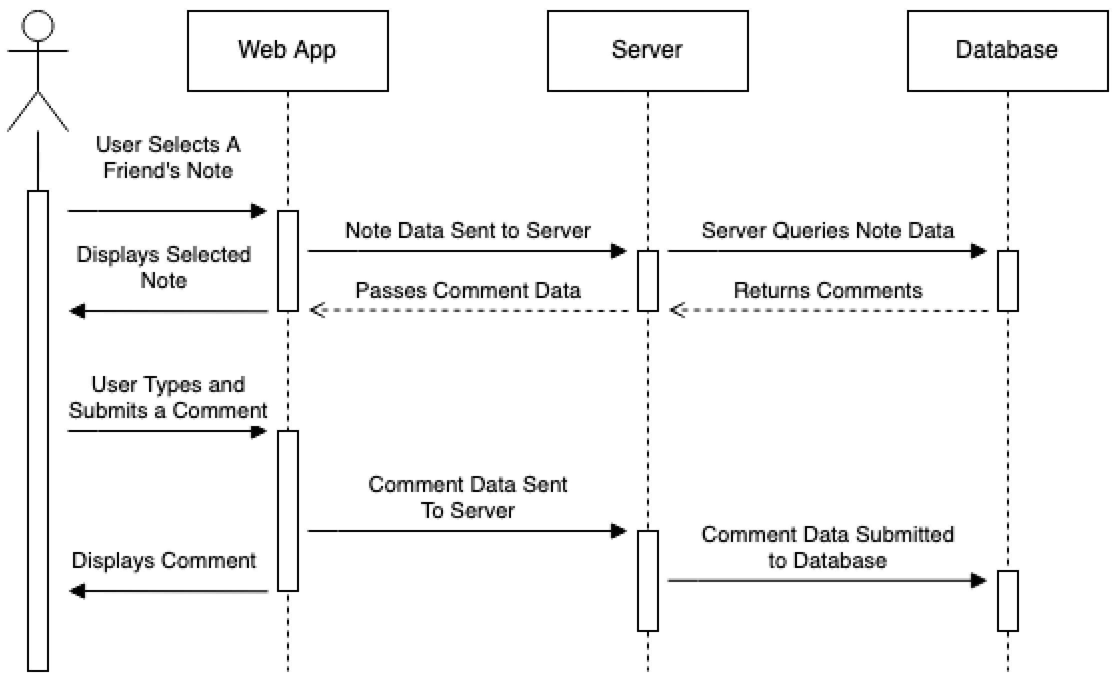
User Creates New Note



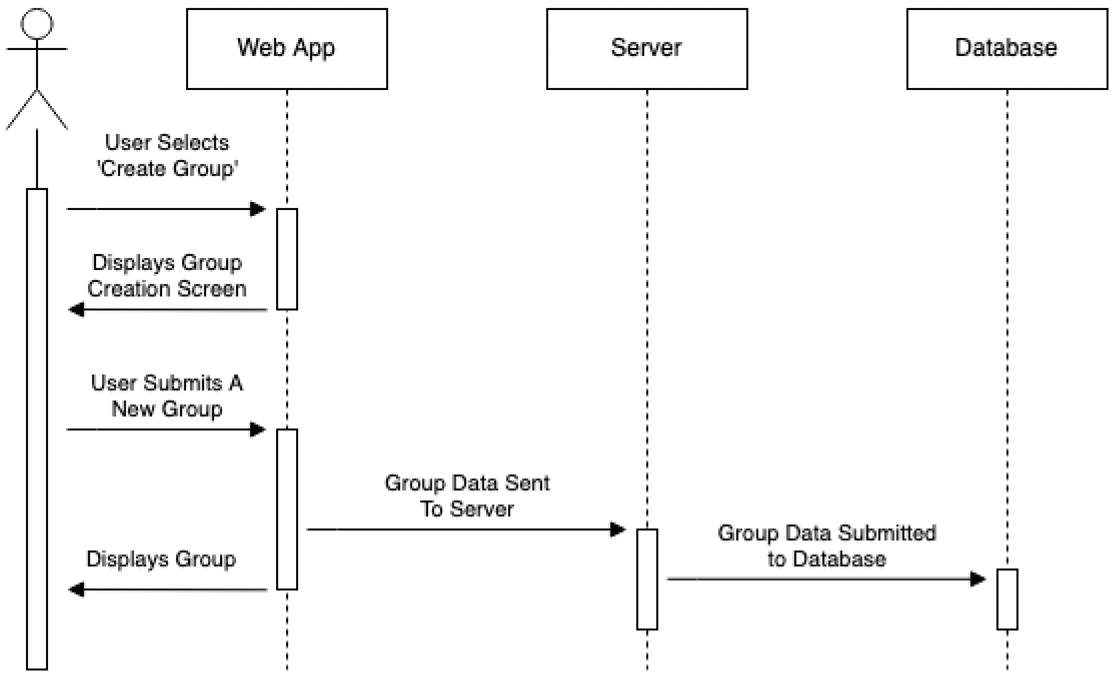
User Visits Web Page


Web Browser
Browser Extension
Server
Database
User Opens Web Page
Web Page Data Sent to Server
Server Queries Database For User Account Info
Returns All Notes on Web Page Related to User
Updates Extension
Updates UI
User Notes and Friends' Notes are now visible in the browser

User Comments on Friend's Note

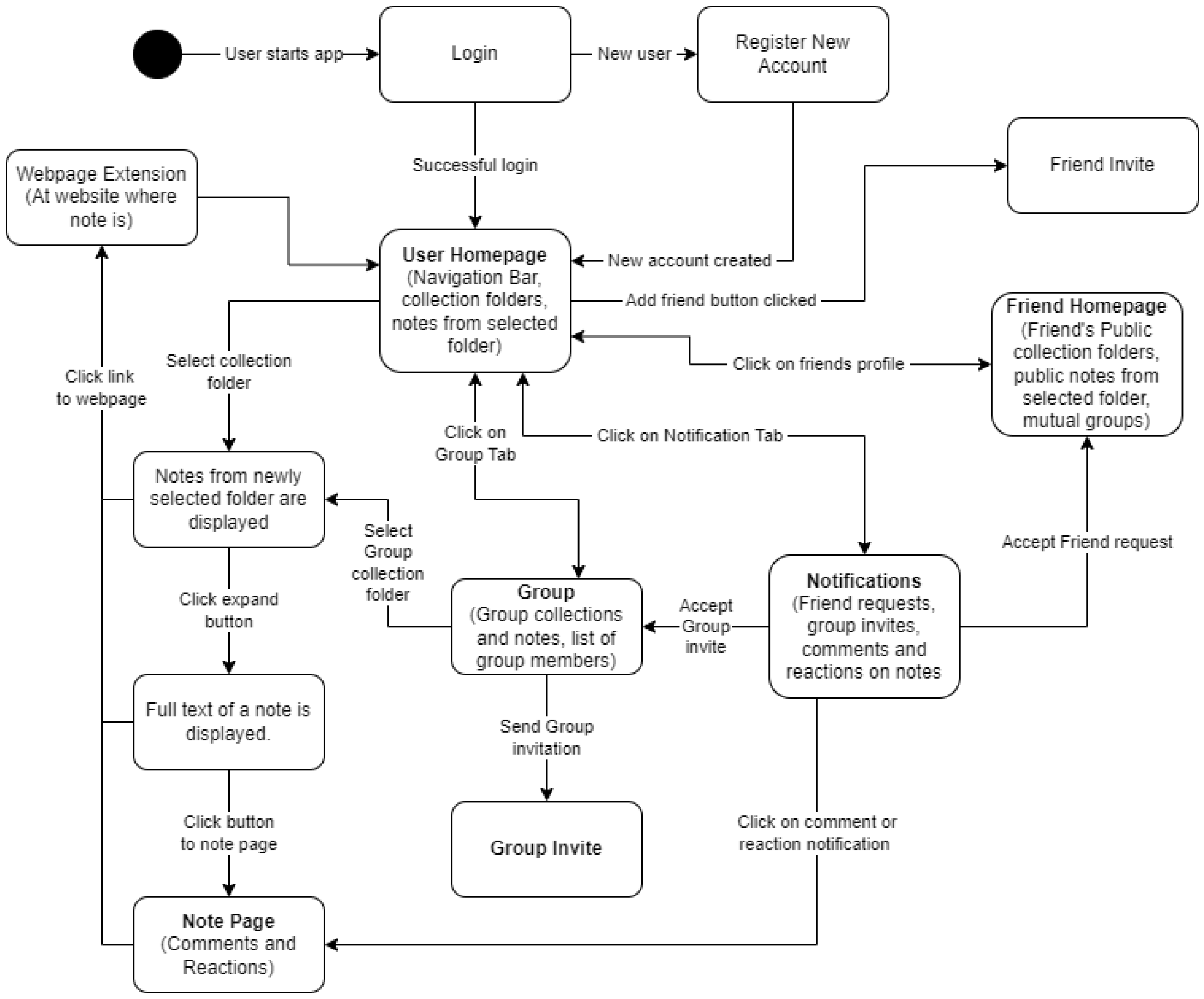


User Creates a Group


Web App
Server
Database
User Selects 'Create Group'
Displays Group Creation Screen
User Submits A New Group
Group Data Sent To Server
Displays Group
Group Data Submitted to Database

6.4 Navigation Flow Map

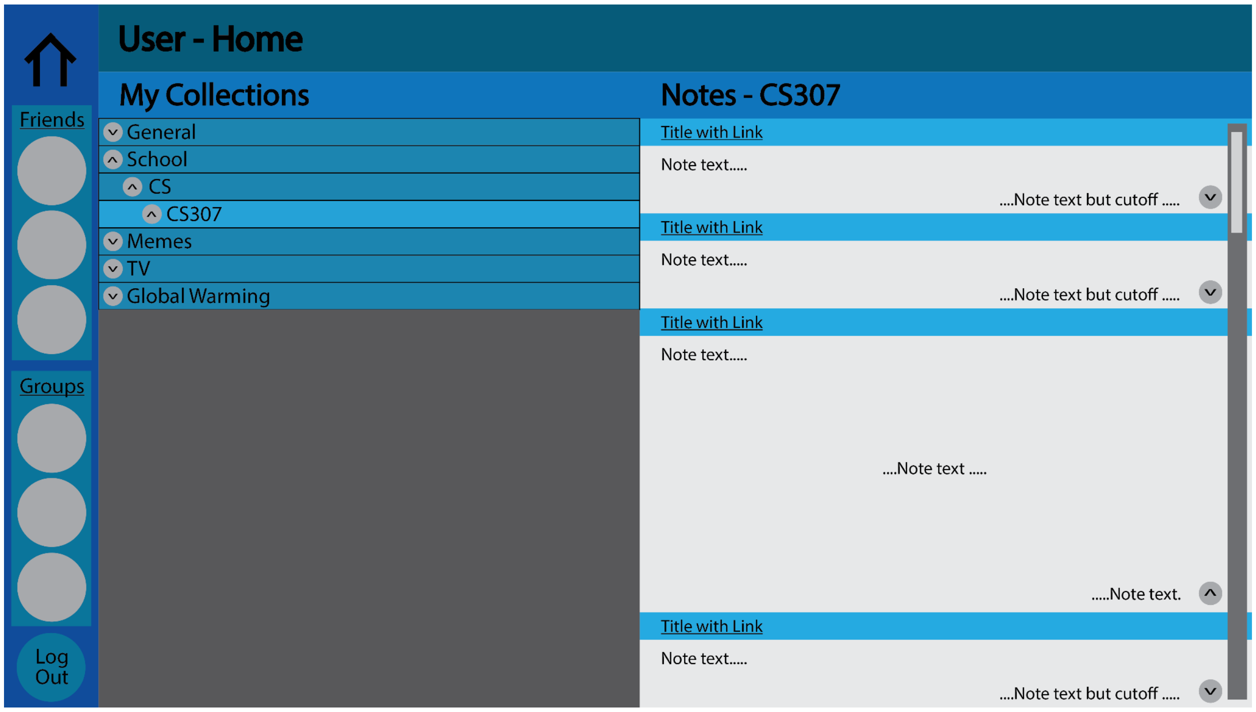
We have designed our webpage to be easy to use for the user. After either logging in or creating an account they will be directed to their own personal homepage. On that page, a list of the user's note collections will be available along with a drop down menu for each collection that will display the collection's subfolders. When a collection is accessed (clicked on), the notes for that collection will be displayed on the right hand side of the screen. The notes will start compact but will have drop down menus that display the whole note. If the user wants to see the comments and reactions on the note, they can click on the note itself to be sent to a separate note page for that note. The group page will look very similar but also display the group members and have a button for sending invites to other users. Invites, friend requests, and comments and reactions to a user's note will be displayed on the notifications tab. When accepted or clicked, the user will be sent to the respective page.


User starts app
Login
New user
Register New Account
Successful login
Webpage Extension (At website where note is)
Friend Invite
User Homepage (Navigation Bar, collection folders, notes from selected folder)
New account created
Add friend button clicked
Friend Homepage (Friend's Public collection folders, public notes from selected folder, mutual groups)
Click on friends profile
Select collection folder
Click link to webpage
Click on Group Tab
Click on Notification Tab
Notes from newly selected folder are displayed
Select Group collection folder
Accept Friend request
Click expand button
Group (Group collections and notes, list of group members)
Accept Group invite
Notifications (Friend requests, group invites, comments and reactions on notes)
Full text of a note is displayed.
Send Group invitation
Click button to note page
Group Invite
Click on comment or reaction notification
Note Page (Comments and Reactions)

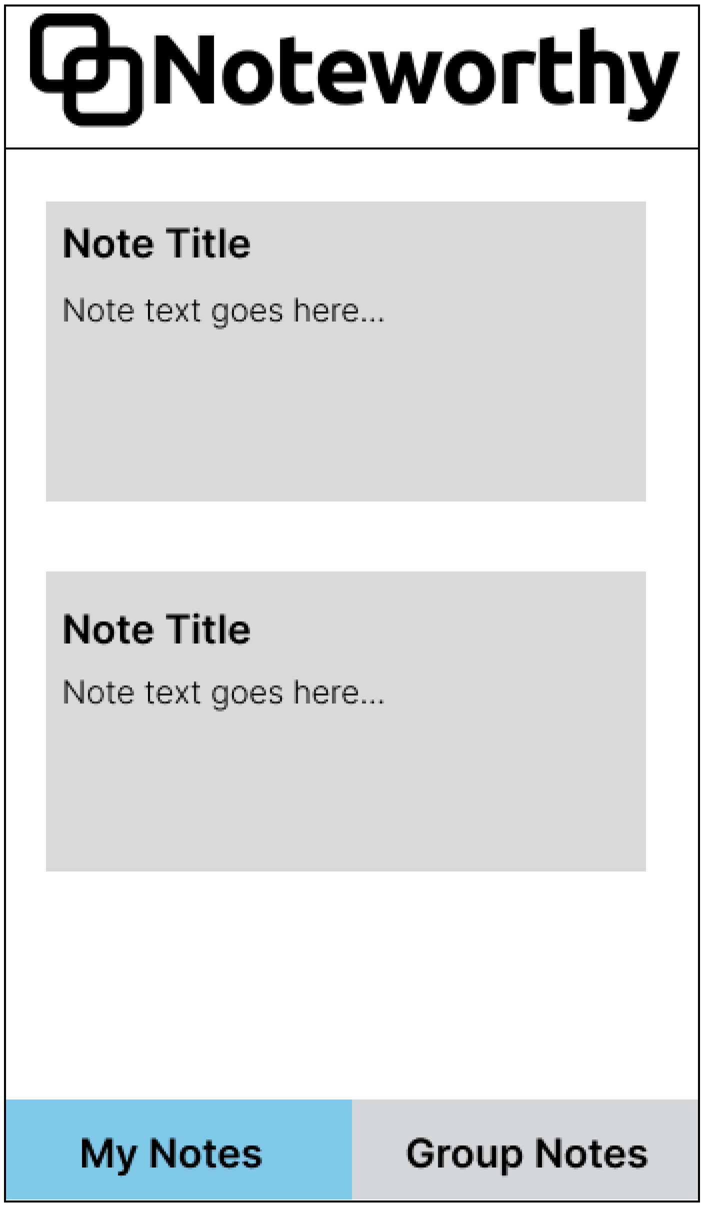
After the user has either logged in or created a new account, a navigation bar will be displayed on the left side of the screen. This navigation bar will be available to use no matter where on the central website the user is. The navigation bar will allow quick access to the user’s homepage, notifications, groups, and friends.

6.5 UI Mockups

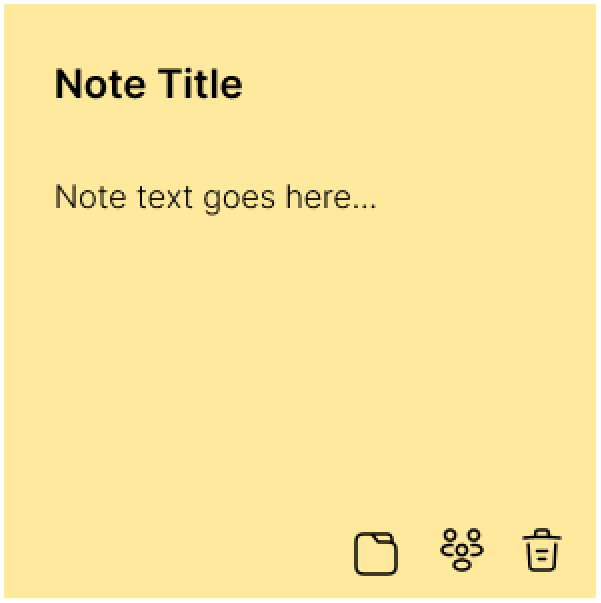
There are two main components of our applications with which users will be interacting: the browser extension component, and the web application hub. We designed these components to work together seamlessly, and to look similarly. We want the experience of switching between the browser extension and the web application to be intuitive to the user.



This is a user's home page in the web application portion. Notes, which are created via the browser extension, can be organized into Collections, shown in the left portion of the page. Selecting these Collections displays the corresponding notes on the right side of the screen.



This is a mock-up of the extension window for our chrome extension. The user will be able to view all of their own notes as well as any group notes that are attached to the webpage they are currently on. They will also be able to navigate to each note by selecting it. Additionally, users can navigate to the main web app hub by clicking on the Noteworthy logo.



This is a mock-up of a sample note that users would add to a webpage. The notes on a webpage would expand into this when selected and collapse into a smaller Noteworthy icon when the user clicks somewhere else so that these notes don't take up too much space on any specific webpage. The icons in the bottom right of each note will let the user file each specific note into a collection, add it to a group, or delete the note.