User Data Visualizer: Snapchat

Richard A. Haase

Campbell University – Senior Project

Author Note

Some aspects of the documentation are a bit odd for two reasons. First, it was my first attempt at documenting a full project. Second, looking back the direction the instructor gave on how to do some things may have been dated and/or not the best way to document a software project.

Visit the current implementation of the system at https://rickyh.dev

Table of Contents

Abstract

The User Data Visualizer for Snapchat is a system which allows users to view the data that Snapchat collects on them and view the data in a concise, easy to understand, and organized manner. The system does this by representing the data graphically when applicable and as organized lists when not. The application interface is accessible via a web app (found at https://r-haase-cu.github.io) and is designed with ease of use in mind. The system is fully functional but with many improvements, including expanding platform compatibility, on the roadmap.

Keywords: User Data, Visualization, Maps, Location Data, Metadata

User Data Visualizer: Snapchat

Many people are aware that internet companies, social media platforms in particular, track them to some degree for various purposes. Very few people are aware of exactly what data is being collected on them nor how that data is being used. The User Data Visualizer for Snapchat seeks to aid in educating users of the platform as to how the company is tracking their every move while in the app. It does this by organizing the data collected by Snapchat and creating easy-to-understand visualizations of the data. Future plans intend to expand the system to support other internet platforms, but this initial release focuses exclusively on Snapchat.

This report documents the development of the User Data Visualizer for Snapchat. The report is structured in accordance with Systems Development Life Cycle (SDLC) covering all 5 phases in chronological order: planning, analysis, design, implementation, and concluding with the roadmap for future improvements.

System Feasibility

System Architecture

Feasibility Criteria	Weight	File Server Architecture	Client-Server Architecture
Economic Feasibility	5%	Very cheap, even free to just host on a web server and send over the web files when requested. **Score: 100**	Much higher operational costs as processing is done server-side. Bandwidth costs also will be significantly higher as files are transferred back-and-forth between the client and the server. **Score: 80**
Technical Feasibility	30%	Very simple to convert the prototype to a file-server architecture. **Score: 100**	Would require re-architecting the entire system. **Score: 20**
Schedule Feasibility	50%	Deliverable by April 12, 2021 **Score: 100**	Impossible to implement before deadline. Earliest deliverable July. **Score: 40**
Operational Feasibility Functionality, compatibility, security, performance.	15%	Security is very good as all data is stored and processed locally on user's device. Very good network performance as very few files are transferred over the network. **Revision: Only supports files of limited size (discovered late in testing)[1] **Score: **75**	Security is a concern as additional measures would have to be implemented to ensure secure transmission and storage of user data. Improved performance when processing large files. Additional functionality enabled for downloading an export of user content using the system. **Score: 90**
Ranking	**100%**	**96.25**	**43.5**

Assumptions and Limitations

Assumptions

1. All Snapchat user data files share the same data structure.
2. Any given user's data file will fit in the local storage of the browser.[1]
3. Users of the system will be able to follow the instructions to properly import their data into the system.

Limitations

1. The system is limited to only data files exported from Snapchat.
2. System cannot process files that exceed the browser's local storage capacity (i.e., data files of users that use Snapchat excessively).

Project Planning

Problem

Snapchat and many other internet companies make it difficult for users to understand what data is being collected on them and for what purpose the data is being collected.

Objective

Generate a graphical/visual view of the data that Snapchat exports when user's request a data export (future compatibility for other companies).

Business Requirements Statement

The system's functionality is centered around one thing: visualizing user data. It must provide a simple interface for importing their data file with clear instructions on how to obtain the file if the user does not already have it on their device. Once the file is imported, navigating between views, understanding the data displayed, and interacting with the data should be clear and self-explanatory.

Project Worth

The value that this project provides is entirely to its users. After using the system, users will gain a deeper understanding and better awareness of what data is being collected on them as they go about their digital lives on their phones and internet browsers. This understanding can be invaluable to user's who may have not been aware of the data being collected on them or just did not understand entirely what data was being collected about them. Providing users with this awareness enables them to make informed decisions about how they go about using technology platforms in the future.

Project Scope

The system will be implemented as a web application where users can import the data export that they receive from a given internet platform (initial compatibility limited to Snapchat) after requesting the data file. Once imported, the data will be processed and output as visualized graphs and charts with other datatypes being organized in expandable lists. Five key areas of data collection that the system will focus on visualizing are:

1. Location History: Put the user location history on a map with a time-lapse feature to show the collection of location data over the passage of time.
2. Communication History: Show the users who were most communicated with and the medium of communication that occurred (text, video, photo, voice, etc.).
3. Contacts: Show that the platforms keep a record of everyone that a user interacts with, even if a contact is "deleted" by the user.
4. Interests: Show how the platform is profiling users (usually with intent of showing more accurate interest-based ads).

5. Content Interaction: Show how the platform is keeping track of every action taken – every post viewed, liked, log-in, etc.

Project Plan

The development strategy for this project is as follows:

1. Create and analyze the logical design options.
2. Finalize system requirements.
3. Prototype the data processing and visualization models to be used.
4. Finalize system design.
5. Construct the system using web technologies: HTML, CSS, JavaScript, and required JS frameworks and libraries (decided upon in physical design).
6. Complete implementation of product on a functioning web server.

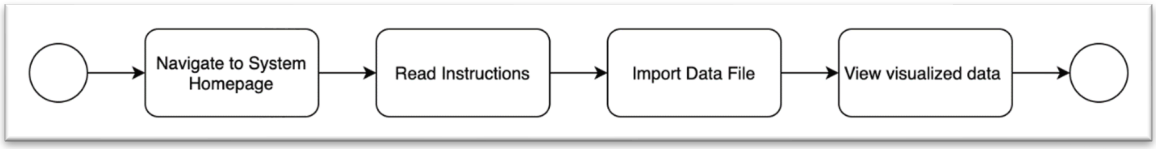
System Analysis

Problems and Opportunities

Internet companies collect substantial amounts of data on their users. By law (at least European law), these companies are required to allow their users access to view and download their data. However, a problem arises when we look at how that data is formatted. Most companies make no effort to format the data that they make available for download readable by users, instead just giving them download links to unformatted .json files. Most users don't know how to open such files, let alone format and read the information they contain.

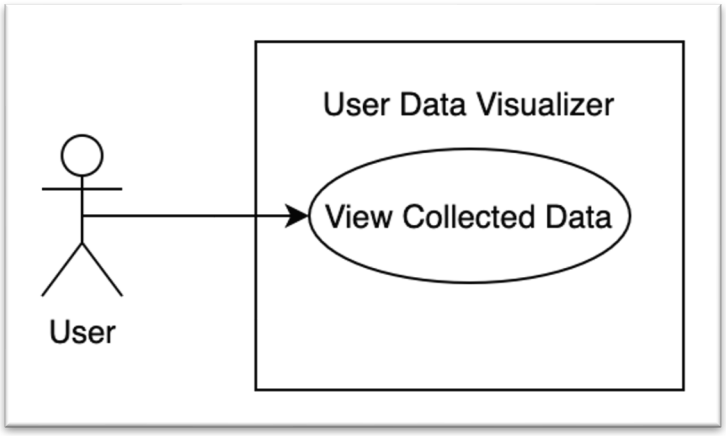
This problem is what this project seeks to address. The objective of this project is to make a system that allows users to import their downloaded data and returns a nicely formatted, interactive, graphical, and easy to understand representation of their data that is being collected.

Business Processes Model



1. System is accessible as a web app through the homepage.
2. The homepage contains instructions on how to use the system and information about what the data is that will be displayed.
    a. Homepage also contains an upload button/drop zone for importing the data file.
3. Once a file is imported, system checks if file is compatible.
    a. If no, user is instructed as to which data files are compatible.
    b. If yes, the system moves straight into the application
4. Once the user is in the application, they are shown the default view.
    a. For Snapchat, the default view will be the location view.
    b. Additional views are displayed in the left pane.
    c. Selecting a new view will change what data is being displayed for the user.

Use Case Analysis



Use Case Title: View Collected Data
Primary Actor: User
Level: Blue
Stakeholder: User
Precondition: User has downloaded data file from Snapchat
Minimal Guarantee: The system will alert the user as to why the files cannot be read
Success Guarantee: User will be redirected to a new page where they can view visualizations of their data
Trigger: User imports data file into system
Main Success Scenario: 1. User's data file imported into system. 2. System parses data file into proper data structure. 3. System outputs default view for location data. 4. User is able to browse other views containing visualizations of other data types.
Extensions: 1a. Data file is incompatible with the system. 1a1. System outputs message informing user of the acceptable data files. 2a. Location data is not contained in data file. 2a1. System hides location view and displays the next prioritized data type by default.

Logical Design

Project Vision (Functionality Targets)

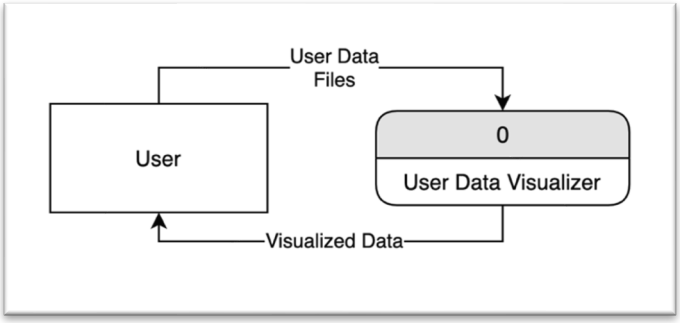
The system will be accessed via a web application where the user will import their data files. After the initial import is complete, the user is sent into the application. There are multiple application views, and each view corresponds to a different category of data that has been collected. These views are selectable via a nav menu accessible from every view within the app. Expected views include the following:

1. Location History: Display a map with pins dropped for each location recorded. Begin at zero and have a time-lapse run through to current day. Every time a location is recorded, drop a pin in the corresponding location on the map. As time goes on the pins add up. In order to keep the map from becoming too cluttered, remove pins that are more than x days old (x will be determined in testing). Have a pause and scrub function available for the time-lapse using a timeline below the map. Current plan is to use Google Maps API.
2. Communication History: Show charts of a user's communication history (number of messages exchanged) with each contact. Also include a pie chart that represents the media used in communication (text, photo, video, voice, etc.). Below the charts should be an expandable list of the user's talk history.
3. Friends View (Address Book): Show all known associates of a user, regardless of whether the user has added the contact or has deleted/blocked them. It's all saved.
4. User Profile: Show the interest and demographic based profile that the platform has built on the user.
5. Content interaction: Create an interactive visualization of how every single user interaction has been collected by the platform.

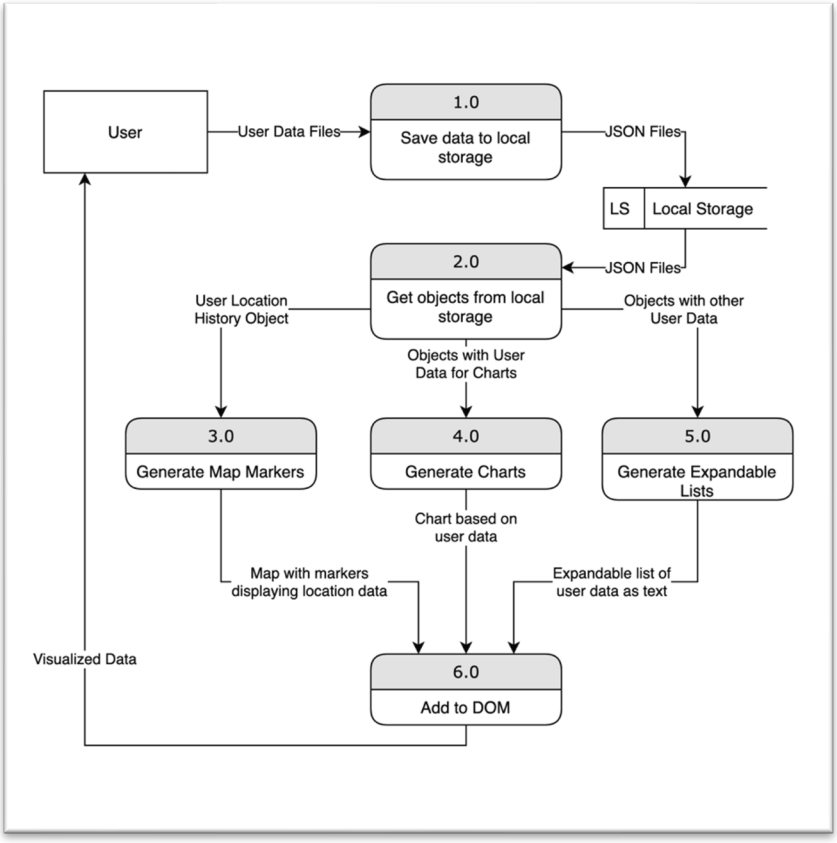
6. Memories: Show the total number of saved photo/video files and the breakdown of media types. A future version should integrate a feature to download all saved files with one click.

System Model

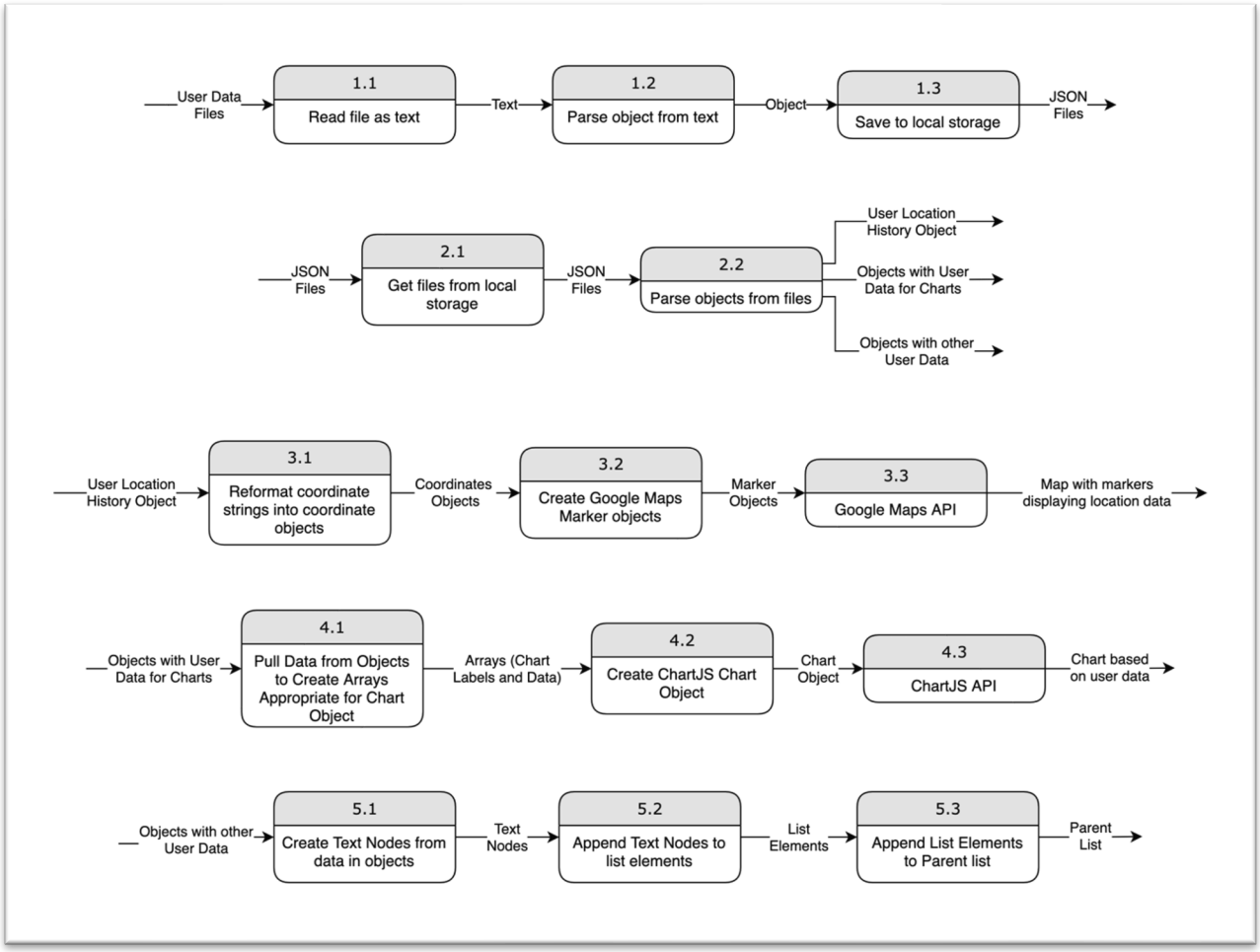
Context Diagram



Data Flow Diagram



Decomposition Diagrams



Physical Design

Candidate Solutions

Maps API

Google Maps	OpenStreetMaps
• Most Familiar to users • Very Reliable • Easy to use API • Touch friendly • Good Performance • Expensive for large amounts of requests	• Open source (free) • Unfamiliar to users • Tests show less-than-desirable performance • More difficult to use/implement

Charts Library

ChartsJS	amCharts	CanvasJS
• **Open Source (free)** • Clean and simple design • Easy implementation • Great performance • **CDN Delivery**	• Free (Watermark) • Fairly robust library of charts • Includes maps	• Proprietary software (requires payments) • Largest charts library • More complex integration

Data Storage

Browser Local Storage	Database
• All storage is done locally on device • Cheaper web hosting costs • More privacy conscious as no user data ever seen by servers • **Discovered that some data files are too large to fit in local storage.[1]	• Higher server costs • Added complexity of integrating and securing database

Data Processing

Client	Client/Server
• All processing is done in-browser on the user's device • Potential for worse performance • Cheaper server costs • Private as all processing is done on user's device and no data is transferred over the network	• Potential for faster performance of application • More expensive server costs • More complex to implement/secure

Web Host

Self-Host	GitHub Pages
• Added complexity of implementation • Cost of server uptime • Increased control of operating environment • Requires stable internet connection	• Extremely simple web hosting • Operates only as fileserver (no backend processing) • No Fees, cloud-based • Requires that code be public

Software Tools

HTML and CSS	Used to create the layout and design of the site
JavaScript	All system logic and user interaction will be coded in JavaScript
Browser	Operating environment for the system
Maps API (Google, OpenStreet, Apple MapKit)	API for integrating a maps service into the system for location data visualization
Charts Library (Chart.js, amCharts, canvasJS)	JavaScript libraries used for creating charts for user data visualization
MySQL	Open source, used to design the database for a client-server architected web app
Node.JS	Used to operate the backend of a web app in a client-server configuration

Selected Solutions

- **Google Maps API**: Selected for reliability and ease of use/implementation.
- **ChartJS**: Selected because it's open source, easy to use, and supports CDN delivery.
- **Browser Local Storage**: Selected because of the reduced cost and complexity.
- **Client based processing**: Selected for reduced cost and complexity of system architecture.
- **GitHub Pages**: Selected because of dead-simple setup and self-hosing is a no-go with current internet setup.

User Interface Design

Inputs

Landing page

- File Input: File input is handled by an HTML input element which allows the user to either drag and drop files into their data files into the drop zone or click the link to pull up the file selection dialogue. This is only found on the landing page.

All Views:

- Expandable lists: Users can select the parent element to expand the list.
- “Clear Data” Button: Users can click this button to clear the data from browser local storage and return to the landing page.

Location View

- Time-lapse controls: Shown in the location history wireframe, the time-lapse controls include the following:
    - Play/pause button: users can click on this button to stop and start the time-lapse.
    - Slider: the slider indicates the position of the time-lapse. The user can click or drag this slider to scrub to a new position in the time-lapse.
    - Clear Markers Button: Button allows users to clear all markers displayed on the map.
- “Show All Markers” Buttons: Clicking these buttons shows all markers for corresponding dataset on the map.
- “Show on Map” Buttons: Clicking these buttons displays the corresponding list element as a marker on the map.

Other Views:

- Chart-selection buttons: Used to show corresponding chart. Clicking a button will hide all other charts and only display the chart that it is associated with.
- Chart tool tips: Hovering over elements of the chart will display details about the data being represented.

Outputs

Landing Page

Information output on the landing page include an introduction, written instructions, and an instructional video guiding the user on obtaining and importing their data files.

All Views

All views list user data in organized lists that are collapsed by default. Users can click on the list title to expand any list.

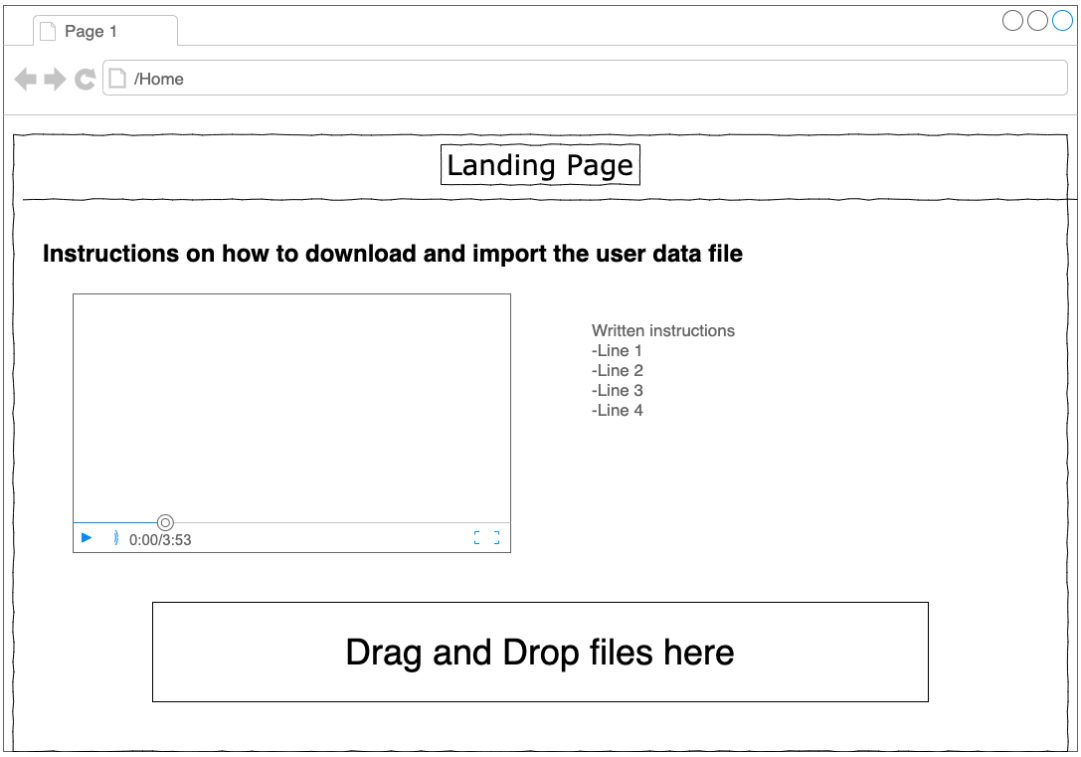
Location View

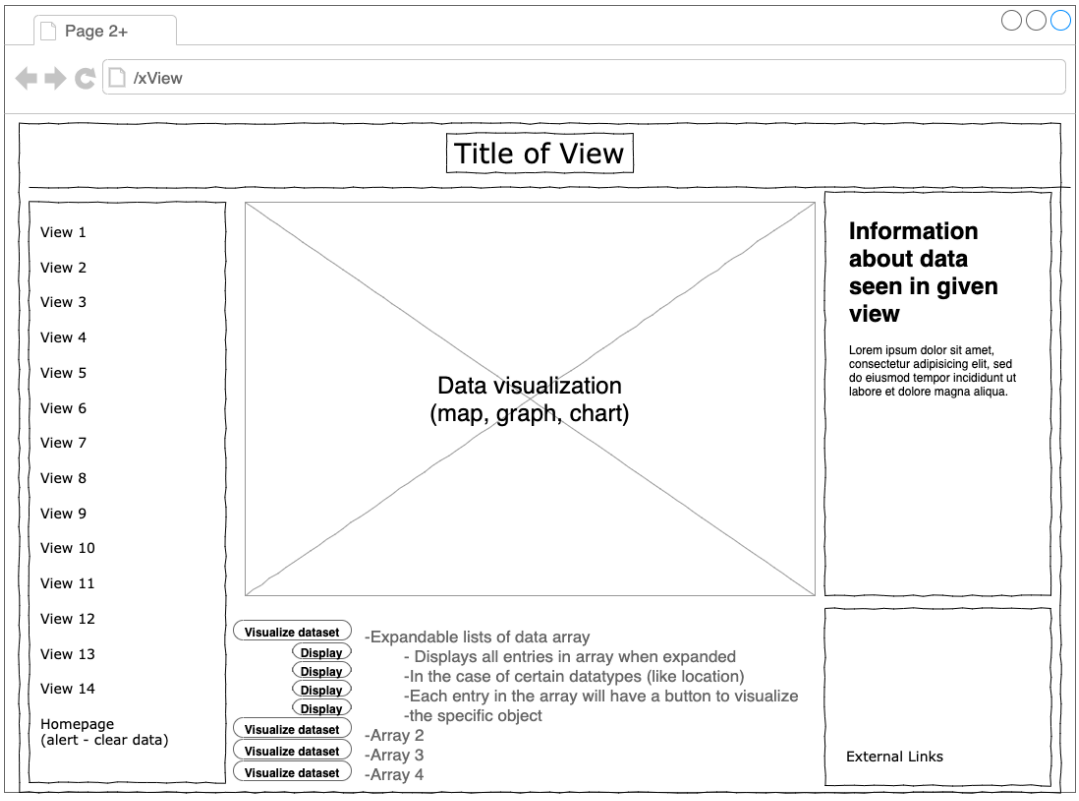
The location view contains a map with a time-lapse of the user's location history for the past 6 months. Data is output on the map as markers placed at each coordinate at the corresponding date. The expandable lists have buttons associated with each list element which creates a marker that is at the location corresponding with the element.

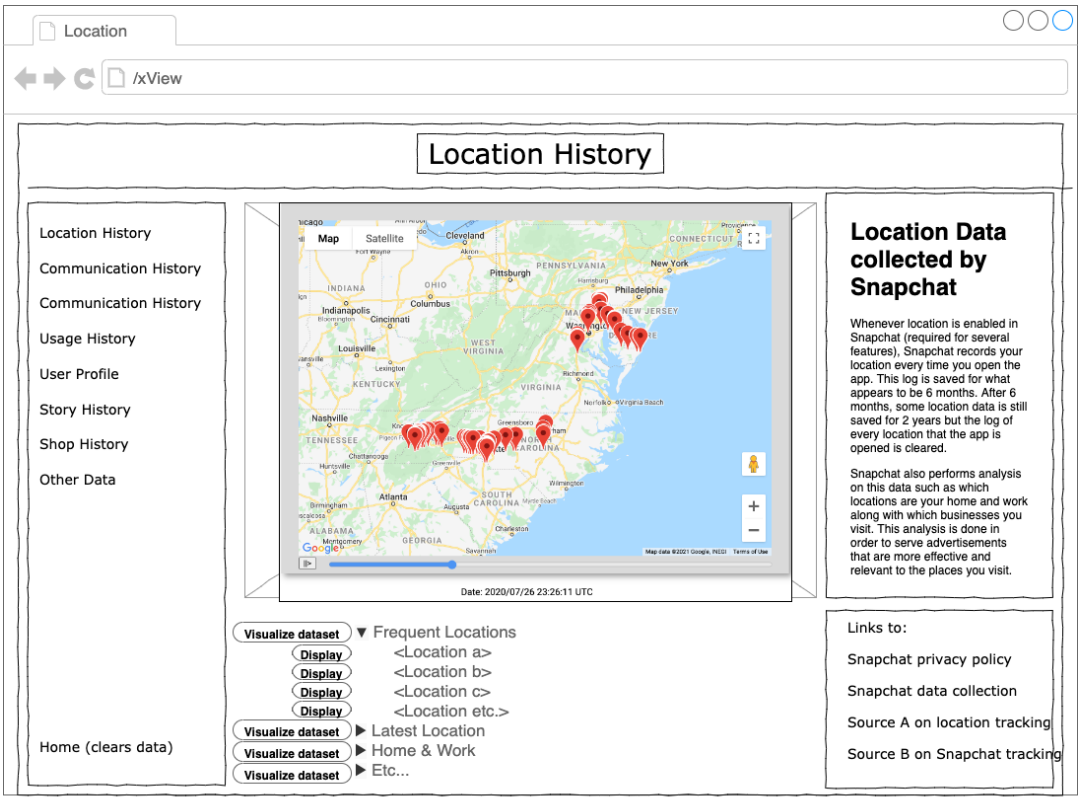
Other Views

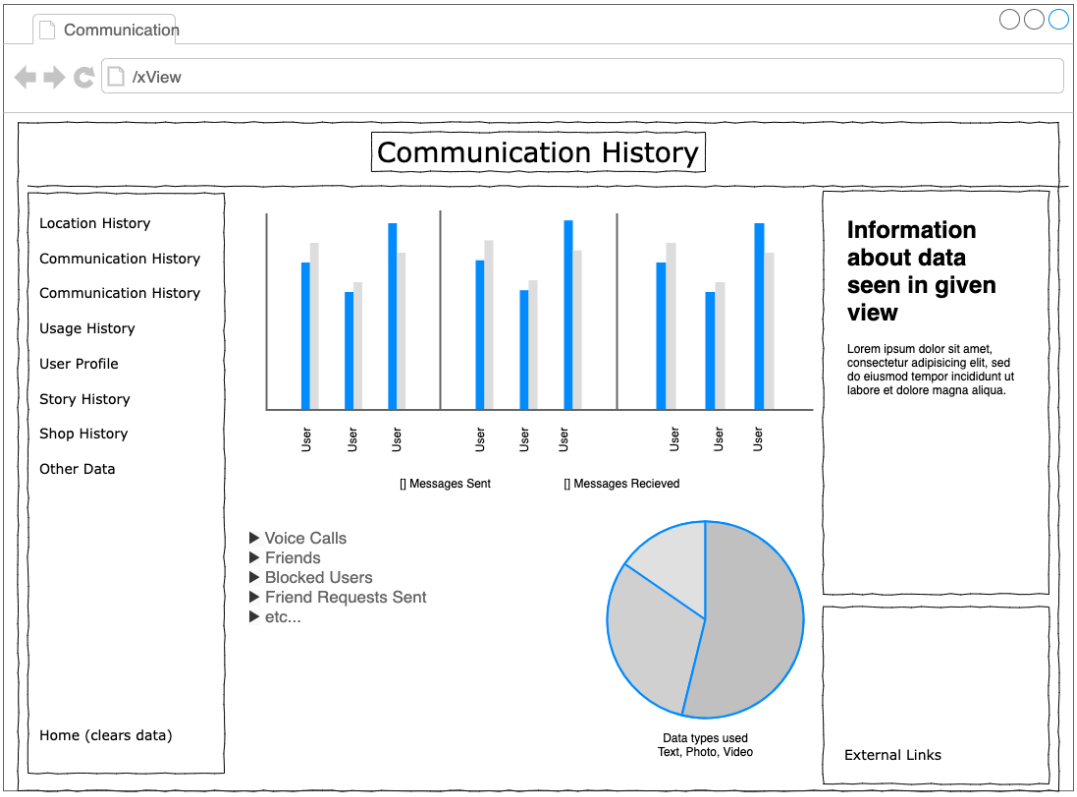
Other views typically contain, in addition to data organized in lists, one or more charts. These charts display an organized visualization of the data from files that the user imported into the system. Examples of these charts include bar charts for user's communication history, engagement statistics, etc., and pie charts to break down the data by statistics like what media types were used across the aggregate of all user communications.

Interface Wireframes






Location
/xView
Location History
Location History
Communication History
Communication History
Usage History
User Profile
Story History
Shop History
Other Data
Home (clears data)
Map
Satellite
Date: 2020/07/26 23:26:11 UTC
Location Data collected by Snapchat
Whenever location is enabled in Snapchat (required for several features), Snapchat records your location every time you open the app. This log is saved for what appears to be 6 months. After 6 months, some location data is still saved for 2 years but the log of every location that the app is opened is cleared.
Snapchat also performs analysis on this data such as which locations are your home and work along with which businesses you visit. This analysis is done in order to serve advertisements that are more effective and relevant to the places you visit.
Visualize dataset
▼ Frequent Locations
Display
<Location a>
Display
<Location b>
Display
<Location c>
Display
<Location etc.>
Visualize dataset
▶ Latest Location
Visualize dataset
▶ Home & Work
Visualize dataset
▶ Etc...
Links to:
Snapchat privacy policy
Snapchat data collection
Source A on location tracking
Source B on Snapchat tracking


Communication
/xView
Communication History
Location History
Communication History
Communication History
Usage History
User Profile
Story History
Shop History
Other Data
Home (clears data)
User
[] Messages Sent
[] Messages Recieved
▶ Voice Calls
▶ Friends
▶ Blocked Users
▶ Friend Requests Sent
▶ etc...
Data types used
Text, Photo, Video
Information about data seen in given view
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
External Links

Implementation

Current system implementation is a simple file-server architected web app hosted by GitHub's free pages service. After building out the application in HTML, CSS, and JavaScript, it was essentially as simple as making a commit to GitHub and enabling GitHub Pages.

System Testing and Evaluation

System Testing and Error Handling

Initial system testing was conducted with my own personal user data. After completing the system prototypes using my dataset to test, I found two volunteers to share their user data with me for further system testing. All of this was done locally, before system deployment.

Error handling and system catches were put in place to catch errors such as:

- Catches improper file types when importing.
- Catches incompatible data files by comparing names to a list of compatible file names.
    - This is not fool proof because users could re-name their data files to match the names of compatible files and then the system would now work properly
- Catches if the files are too large.
- Catches missing file when loading view (location only, other views allow it because of partial functionality even with missing files).
- Catches if user only imports one file (and prompts user to select and import all files).

After internal testing, the system was deployed as a Beta. The first tester reported that his files were not all properly importing. It was found that his location_history.json file, totaling 17MB, was too large for browser local storage. This led to an added check at file import to catch and alert the user if their files are too large.[2]

Once the system passes all of the file checks, it is assumed that the user has uploaded the proper files. Proper files are assumed to be properly formatted with complete data sets. There are a few catches throughout the system to handle unexpected data but those catches only exist for errors found in testing.

Evaluation

System was primarily evaluated based on ease of use and consistency. Other than the beta tester who has issues with their file size, every tester interviewed said that the instructions were clear, the interactions were simple, and the information was straightforward and easy to interpret.

System Support Procedures

Files too big

If the system displays the “files too big” error message, users can often circumvent this by importing only a portion of the data files to visualize at a time. This will result limited functionality and some views will display error messages or simply not display the data that is expected from the missing files. The total size of uploaded files must be below 5.7 MB (according to testing, will revise if updated figure is found). Below is a chart to show which files to upload for each view to function:

Location	Communication	Profile
• location_history.json	• chat_history.json • snap_history.json • friends.json • talk_history.json	• user_profile.json • account_history.json • account.json • ranking.json
History	**Stories**	**Memories**
• account_history.json • account.json • search_history.json • user_profile.json	• story_history.json • friends.json • subscriptions.json	• memories_history.json

NOTE: At least 2 files must be imported at a time.

User Survey

Were the instructions easy to understand and follow?

Was the system easy to use?

Did you find the visualized data to be easy to understand?

Did you find the data shown to be useful or interesting?

What data was missing that you felt should be shown?

What additional information would you like to know about the data shown?

What additional services would you like to see supported by the system?

- Apple
- Amazon
- Google
- Facebook
- Instagram
- Twitter
- Inuit
- Other (please specify)

Did you experience any problems or unexpected behavior when using the system?

How would you recommend improving the system?

Any additional feedback?

Future Enhancements

1. Code Refinement:
    a. Refactor location.js to be more modular.
    b. Adjust algorithm used for analyzing Snap/Chat history to ignore entries with an empty MediaType.
    c. Optimize performance of algorithms in location view for displaying large datasets.
        i. Key example is the button to show all location history. System becomes unresponsive for a minute.
            1. Add a loading indicator.
            2. Create objects in the background to reduce work when button is clicked.
            3. Skip coordinates that are the same as previous coordinates.
    d. Improve size limit (may require client-server re-architect).
    e. Refactor slider-related functions
        i. Call a single function from event listener and time-lapse loop instead of repeating code.
2. Support documentation and Error handling:
    a. Create publicly accessible support documentation with links to relevant articles in the error alerts.
    b. Make error handling more robust for unexpected values in data.
    c. Add checks for required files to every view (as implemented in location view).
    d. Add error logging to backend error log file for diagnostics and troubleshooting.

3. User interface adjustments:
    a. Make the nav bar indicate which page the user is currently on.
    b. Add total Snaps/Chats to communication view.
    c. Remove DOM elements for missing data (instead of having an empty list/chart).
    d. Create new landing page that includes a more “exciting” design and a “sales pitch” to sell visitors as to why they need to use the system.
    e. Be intentional about colors in graphs - match Snapchat color scheme (e.g. Snaps: red, chats: blue, videos: purple, etc.).
    f. Hide any views that the data files are not available for.
    g. Make “Story Views” chart horizontally scrollable.
4. Additional Features:
    a. Incorporate memories one-click downloader (https://github.com/ToTheMax/Snapchat-All-Memories-Downloader).
    b. Add bar chart to Memories View to display how many memories saved per year (and separate on media type).
    c. Return user location at corresponding time whenever a date/time is selected in any view in the system (not only in the location view).
5. Additional views (with data files they use):
    a. Shop History
        i. netsuite_orders.json
        ii. purchase_history.json
        iii. shop_history.json
    b. Customer Communication

i. email_campaign_history.json

ii. in_app_reports.json

iii. in_app_surveys.json

iv. support_note.json

v. terms_history.json

c. Connected Apps

i. bitmoji_kit_user.json

ii. bitmoji.json

iii. connected_apps.json

d. Data for Additional Features

i. cameos_metadata.json

ii. community_lenses.json

iii. scans.json

iv. snap_ads.json

v. snap_games_and_minis.json

vi. snap_map_places_history.json

vii. snap_pro.json

viii. snap_tokens_order_history.json

6. Expand compatibility to support user data from other internet platforms.

a. Facebook

b. Amazon

c. Apple

d. Google

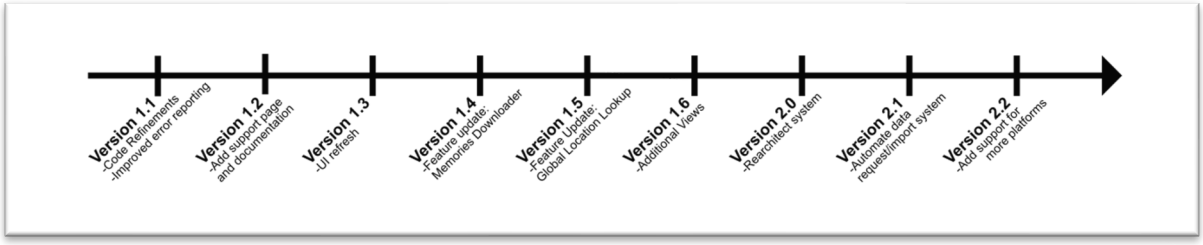
e. Inuit

f. Instagram

g. etc.

7. Automate request and import process for user data files.

Roadmap



Conclusion

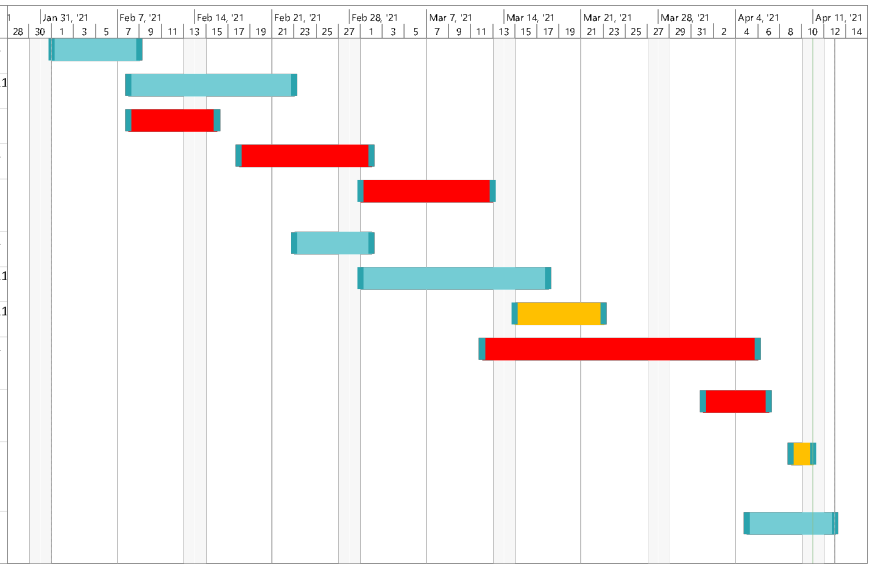
The project (User Data Visualizer: Snapchat) was successful in completing its primary mission: create a simple way for users to understand exactly what data Snapchat is collecting on them when the use the app. The system is deployed and both stable and fully functional (can be seen at http://r-haase-cu.github.io) and user responses indicate that the system is easy to use and increases their understanding of what data is being collected on them. There are many improvements planned for future releases and the system is expected to gain compatibility with more internet services in the coming years.

Appendix A

Repository

Planning Phase

ID	Task Name	Duration	Start	Finish
1	Preliminary Investigat	6 days	Mon 2/1/21	Mon 2/8/21
2	Problem Analysis	11 days	Mon 2/8/21	Mon 2/22/21
3	Javascript training refresh	6 days	Mon 2/8/21	Mon 2/15/21
4	Prototyping proof of concept	8 days	Thu 2/18/21	Mon 3/1/21
5	Prototyping to test different algorithims and datastructures	10 days	Mon 3/1/21	Fri 3/12/21
6	Requirements Analysi	5 days	Tue 2/23/21	Mon 3/1/21
7	Decision Analysis	13 days	Mon 3/1/21	Wed 3/17/21
8	Final Design	6 days	Mon 3/15/21	Mon 3/22/21
9	Construction (integrate prototypes with final design)	17 days	Fri 3/12/21	Mon 4/5/21
10	Implimentation (Get web server with application running)	4 days	Thu 4/1/21	Tue 4/6/21
11	Opperations & Support (create documentation for future release	2 days	Fri 4/9/21	Sat 4/10/21
12	Final Report (Compile documentation into	6 days	Mon 4/5/21	Mon 4/12/21



Haase, Richard Prepared: February 6, 2021

Project Charter

Project Name: User Data Visualizer | **Project Start/End:** 2/1/2021 – 4/12/2021

Project Overview:

Internet companies collect substantial amounts of data on their users. When users request to view the data collected on them, the services typically give them a set of files that are formatted in a manner that is very difficult for most users to efficiently interpret. This project seeks to address this issue and provide a system which formats user data in a way that is graphical and easy to interpret.

Problem:

Many internet companies make it difficult for users to understand what data is being collected on them and for what purpose.

Objective:

Format user data exported from other platforms into concise and easy to interpret visualizations.

Scope:

The system will be implemented as a web application where users can import the data export that they receive from a given internet platform (initial compatibility limited to Snapchat) after requesting the data file. Once imported, the data will be processed and output as visualized graphs and charts with other datatypes being organized in expandable lists. Five key areas of data collection that the system will focus on visualizing are:

1. Location History: Put the user location history on a map with a time-lapse feature to show the collection of location data over the passage of time.
2. Communication History: Show the users who were most communicated with and the medium of communication that occurred (text, video, photo, voice, etc.).

Haase, Richard Prepared: February 6, 2021

3. Contacts: Show that the platforms keep a record of everyone that a user interacts with, even if a contact is "deleted" by the user.
4. Interests: Show how the platform is profiling users (usually with intent of showing more accurate interest-based ads).
5. Content Interaction: Show how the platform is keeping track of every action taken – every post viewed, liked, log-in, etc.

Project Worth/Assumptions:

The value proposition that this project makes assumes that users will find value in being able to see in a clear and concise manner how much/what data big tech companies are collecting on them.

Development Strategy:

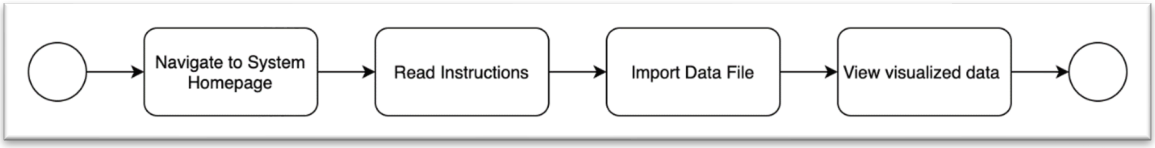
1. Create and analyze logical design options.
2. Prototype the data structures, algorithms, and visualization methods to be used.
3. Finalize system requirements.
4. Finalize system design.
5. Construct the system using web technologies (HTML, CSS, JS, and required JS frameworks and libraries). Integrate the prototypes with the final system design.
6. Complete implementation of product on a functioning web server.

*See Gantt Chart included in the Figures section at the end of the report

**Charter Revised on April 11, 2021

Analysis Phase

Business Process Model

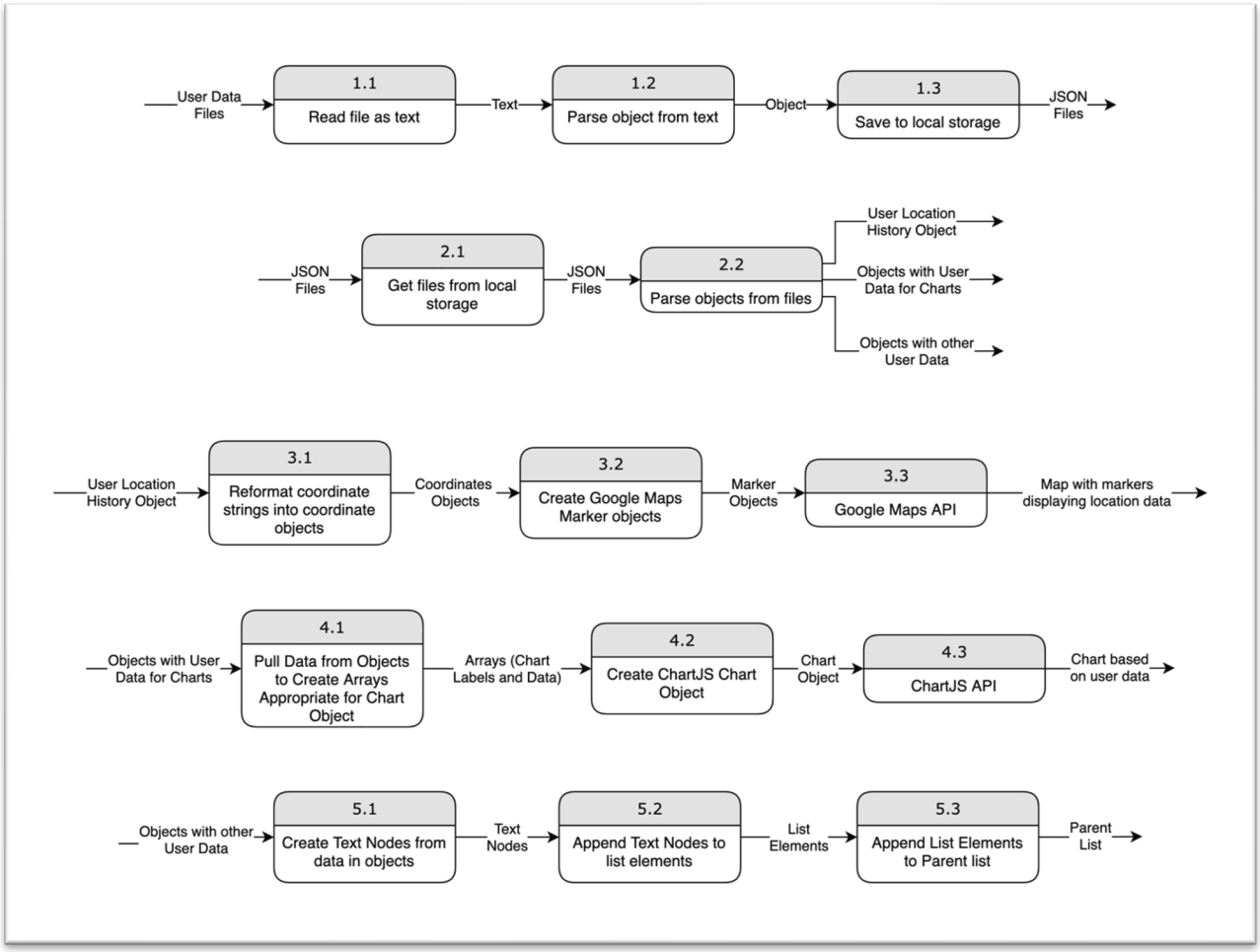


System Use Case

Use Case Title: View Collected Data
Primary Actor: User
Level: Blue
Stakeholder: User
Precondition: User has downloaded data file from Snapchat
Minimal Guarantee: The system will alert the user as to why the files cannot be read
Success Guarantee: User will be redirected to a new page where they can view visualizations of their data
Trigger: User imports data file into system
Main Success Scenario: 1. User's data file imported into system. 2. System parses data file into proper data structure. 3. System outputs default view for location data. 4. User is able to browse other views containing visualizations of other data types.
Extensions: 1a. Data file is incompatible with the system. 1a1. System outputs message informing user of the acceptable data files. 2a. Location data is not contained in data file. 2a1. System hides location view and displays the next prioritized data type by default.

Design Phase

Functional Decomposition Diagrams



Algorithms and Data Structures

Algorithms for File Import (Process 1.0: Save data to local storage)

```
SET constant "input" equal to the files imported by user
SET constant "supportedFiles" equal to an array of all acceptable file names

This function saves the files from the HTML input element to the browser local storage
before redirecting the user to the Location View
Function handleFiles is called when there is a change in the HTML input element and
all files stored in the element are passed into the function

    function handleFiles(files) //constant "input" is passed into the function
        TRY
            SET variable "totalSize" equal to 0
            FOR EACH element in the passed parameter "files"
                ADD the size of the element to totalSize
```

```
            FOR EACH element in the passed parameter "files"
                IF element type is a .json file
                    IF element name matches string in "supportedFiles" array
                        IF "totalSize" is less than capacity of local storage
                            SAVE element to local storage
                                SET Key equal to file name
                                SET Value equal to file contents
                        ELSE
                            THROW error "Files too big"
                    ELSE
                        THROW error "Unsupported data file"
                ELSE
                    THROW error "Wrong file type"

            IF only one file imported
                THROW error "Only one file Imported"

            REDIRECT to Location View after 500MS delay

        CATCH error
            IF "Wrong file type"
                ALERT user of error
            ELSE IF "Unsupported data file"
                ALERT user of error
            ELSE IF "Only one file imported"
                ALERT user of error
            ELSE IF "Files too big"
                ALERT user of error
            ELSE
                ALERT to remind user to review instructions
```

Algorithm for Data Retrieval (Process 2.0)

```
This function is used to get files from local storage and parse the text to an object
"file" parameter is a string of the key of the file used in local storage
       function getAndParse(file)
              GET file from local storage
              PARSE object from file contents
              RETURN parsed object
```

Algorithms to Generate Map Markers (Process 3.0)

```
An instance of "hist" is an array consisting of:
    Objects consisting of:
        Time and
        Latitude, Longitude

This function reformats the variables contained in the string found in attribute
Latitude, Longitude
and adds an object "coords" which is readable by the Google Maps API
    function reformatCoordsHist()
        FOR EACH object in "hist"
            SPLIT Latitude, Longitude at (,)
            CREATE object inside the current object
                SET lat equal to PARSE float from first half of split string
                SET lng equal to PARSE float from second half of split string
```

```
Now, an instance of "hist" is an array consisting of:
    Objects consisting of:
        Time and
        Latitude, Longitude and
        Coords which consists of:
            Lat and
            Lng

The this function moves the slider incrementally over time and creates a marker object
based on the slider position
This creates a time-lapse effect when the toggle button is in the ON position:
    1. Control the toggle position of the play/pause button
    2. Create a new location marker every 1 MS
    3. Move the slider as these markers are created
    4. Executes number 2 only while the toggle is set to "GO"

    Variable j keeps track of the index in the array "hist" and the position of the
slider

    async function toggle()

        SET variable j equal to slider position

        IF button value is set to STOP
            set button value to GO
            set button style to pause
        ELSE IF button value is set to GO
            set button value to STOP
            set button style to play

        DO
            SET slider position equal to j

            SET variable output to the HTML element dateIndicator
            SET variable time equal to the string contained in the Time attribute in
array "hist" at index equal to j
            ADD time to output

            SET variable coords equal to the coords object in "hist" at index j
            SET variable style equal to the URL for red map marker icon
            PASS coords, time, and style through function addMarker
                (add marker creates a marker object given the previous inputs and
pushes the object
                        to the array of visible markers to be displayed on the map)

            increment variable j

            WAIT 1MS

        WHILE j < length of "hist" AND button value is set to GO

        IF j equals the length of "hist"
            SET j equal to 0
            SET button value to STOP

This function creates a marker based on slider position when user interacts with
slider

    slider.oninput
```

```
        SET play/pause button value to STOP

        SET variable time equal to time value from "hist" at the index equal to the
slider position
        Output time as text element in the HTML

        SET variable coords equal to the coordinates from "hist" at the index equal to
the slider position
        SET variable style equal to the blue dot icon URL

        PASS coords, time, and style into the addMarker function

An instance of "Areas you may have visited in the last two years" is an array
consisting of:
    Objects consisting of
        City and
        Postal Code and
        Region and
        Time

This function uses Google's Geocode API to find coordinates based on a given Postal
Code

    async function getAreaCoords()
        FOR EACH element in "Areas you may have visited in the last two years"
            TRY
                FETCH Google's geocode API request using Postal Code attribute of the
current object
                SET attribute coords in current object equal to the object "location"
that was returned from the API response
            CATCH
                log failed API request to the console

Now, an instance of "Areas you may have visited in the last two years" is an array
consisting of:
    Objects consisting of
        City and
        Postal Code and
        Region and
        Time and
        Coords which consists of
            Lat and
            Lng

An instance of markers is declared as an empty array
    (The Google Maps API reads this array to get the properties of the markers it
needs to display)

This function is called to add a marker to the markers array
    function addMarker(coords, content, style)
        SET constant infowindow equal to the value passed in through the content
parameter

        SET constant marker equal to a new Google Maps Marker object
            SET attribute position equal to the coords object passed through the
function's parameters
            SET attribute icon equal to the style parameters passed through the
function
            SET map attribute equal to the map instance displayed on the page
```

```
        ADD event listener to the marker object to display the content in constant
infowindow on mouseover
        ADD event listener to the marker object to hide the infowindow on mouseout

        PUSH object to markers array

Now, an instance of "markers" consists of:
    One or more marker object consisting of:
        Style and
        infoWindow and
        Coords consisting of
            lat and
            lng
        and event listener to show infoWondow on mouseover
        and event listener to hide infoWindow on mouseout
```

Algorithms to Generate Charts (Process 4.0)

```
Sample algorithm for creating a data structure that is readable by the Chart.js API
from the user data file

Raw data imported from user data file:
    An instance of "story" consists of:
        One of more Objects consisting of:
            View and
            Media Type and
            View Date

The Chart.js API requires 2 arrays, one for the dataset and one with the labels.
In the data, a "View" represents 1 story view and contains the username of the poster
of the story.
For the chart, each username will get it's own count of views.
These views will be split up by Media Type.
There are 3 possible Media Types so each username will have 3 bars, each representing
the number of views of corrosponding Media Type.

The resulting labels array needs to have 1 instance of each unique username appearing
in "story"
    SET constant "usernames" equal to a sorted array of each unique instance of
item.View in the story array

    An instance of "usernames" consists of:
        One or more String values

Create a 2-dimensional array where the first index corrosponds to each username in
"usernames" and the arrays within each index contain the objects that contain the
corresponding usernames.
    SET constant "objArrays" as an empty array

    This function pushes the results of getObjArray() into the corresponding array
index in "objArrays"
    function create2dArray()
        FOR EACH element in "usernames"
            PUSH the result of getObjArray(element) to "objArrays"

    This function returns an array of every object from "story" that has a matching
username as the value that is passed through the parameter.
    function getObjArray(name)
        SET constant "tempArray" as an empty array
```

```
        FOR EACH element in "story"
            IF attribute "View" of the current object matches passed value "name"
                PUSH the object to tempArray

        RETURN tempArray

"usernames" is acceptable for the labels array in the Chart.js chart object.
3 arrays for each media type are still required where the data is in the same order as
"usernames."
This means that the data in index 0 of the data arrays corresponds to the username
contained at index 0 of "usernames."

This function returns an array where each index in the array is a count of the number
of times ao object with given Media Type (Media Type matching the passed parameter) is
found in each index of "objArrays" (1st dimension)
    function countDataArray(mediaType)
        SET constant "countDataArray" as an empty array

        FOR EACH array in "objArrays"
            FILTER the array for objects where the Media Type matches the "mediaType"
passed through the function
            PUSH the number resulting from the filter search to "countDataArray"

        RETURN "countDataArray"

    SET "dataArray1" equal to the result of countDataArray('media type 1')
    SET "dataArray2" equal to the result of countDataArray('media type 2')
    SET "dataArray3" equal to the result of countDataArray('media type 3')

    each instance of dataArray is an array consisting of:
        one or more intenger values

The 4 arrays ("usernames" and the 3 data arrays) all contain the data in the proper
data structure to pass into the Chart.js object (template found in Appendix E)
```

Algorithms to Generate Expandable Lists (Process 5.0)

```
An instance of "acctHist" consists of:
    "Display Name Change" which is an array consisting of:
        Objects consisting of:
            "Date" and
            "Display Name"
    and "Email Change" which is an array consisting of
        Objects consisting of:
            "Date" and
            "Email Address"
    and "Mobile Number Change" which is an array consisting of:
        Objects consisting of:
            "Date" and
            "Mobile Number"
    and "Password Change" which is an array consisting of:
        Objects consisting of:
            "Date"
    and "Snapchat Linked to Bitmoji" which is an array consisting of:
        Objects consisting of:
            UNKNOWN
    and "Spectacles" which is an array consisting of:
        Objects consisting of:
```

```
            UNKNOWN
    and "Two-Factor Authentication" which is an array consisting of:
        Objects consisting of:
            "Date" and
            "Event"
    and "Account deactivated / reactivated" which is an array consisting of:
        Objects consisting of:
            UNKNOWN

This function is an example of the algorithms used to create the expandable lists
Other variations of this algorithm exist in the code for compatibility with other data
structures
    function listAcctHist()
        SET "parent" equal to the details element in the HTML

            FOR EACH attribute of acctHist
                SET "arrayName" equal to the name of the attribute
                SET "array" equal to the value of the attribute
                IF "array" has a length greater than zero
                    SET "list" equal HTML element <ul>
                    APPEND "arrayName" as a text node to "list"

                    FOR EACH index in "array"
                        SET "item" equal to HTML element <li>
                        SET "text" equal to the strings contained in the index
                        APPEND "text" to "item"
                        APPEND "item" to "list"

                    APPEND "list" to "parent"
```

User Responses

User Survey 1: Claudiu Oros 04/13

Were the instructions easy to understand and follow?

Yes, the instructions were very straightforward and simple to follow!

Was the system easy to use?

The system was easy to use up to the point of error.

Did you find the visualized data to be easy to understand?

Yes! The way the web-app was set up, I could easily locate all sections of the data I inserted, and interpret what the web-app outputted.

Did you find the data shown to be useful or interesting?

The data was very interesting to read and see, it surprised me that Snapchat stored so much of my location data.

What data was missing that you felt should be shown?

I don't feel like anything was really missing, most important items were shown, if any more were shown it may have felt cluttered and messed up the UI.

What additional information would you like to know about the data shown?

N/A

What additional services would you like to see supported by the system?

Instagram - I would like to see support for Instagram and how they use my data, particularly for advertising. I know Apple and google store my location and I prefer that, but I never know how Instagram uses my location and advertising interests.

Did you experience and problems or unexpected behavior when using the system?

Yes, in a way. It was not that the system/app was at issue, it was rather that my location file was too big to be interpreted and stored by the native browser local storage/file reader. If this were an app that was developed to run as a free-standing application on windows, it would likely run the entire file as opposed to being unable to store and read my large data file.

The developer helped me understand the issue and helped try to anticipate and fix the issue on his end too.

How would you recommend improving the system?

The system is fine as it is, what could be done forward from this point is anticipating what bugs could happen, and having a way to report errors as they happen.

Any additional feedback?

N/A

User Survey 2: Keegan Chit Khin 04/13

Were the instructions easy to understand and follow?

Yes

Was the system easy to use?

Yes

Did you find the visualized data to be easy to understand?

Yes

Did you find the data shown to be useful or interesting?

Both

What data was missing that you felt should be shown?

N/A

What additional information would you like to know about the data shown?

N/A

What additional services would you like to see supported by the system?

Google

Did you experience and problems or unexpected behavior when using the system?

No

How would you recommend improving the system?

N/A

Any additional feedback?

Yes, I used my Samsung Galaxy 21 ultra 5g and it worked fine

User Survey 3: Grant Semke 04/13

Were the instructions easy to understand and follow?

The instructions were easy to follow

Was the system easy to use?

The system was easy enough to use that I could Execute without having to watch the video at the same time

Did you find the visualized data to be easy to understand?

I find the data useful for examining how are use Snapchat as well as interact with other people on the platform

Did you find the data shown to be useful or interesting?

I find the data useful for examining how are use Snapchat as well as interact with other people on the platform

What data was missing that you felt should be shown?

There wasn't any data that I thought should've been there that wasn't represented

What additional information would you like to know about the data shown?

There wasn't any additional data that I thought should've been there

What additional services would you like to see supported by the system?

I would like it to be used for Amazon Google Twitter and Instagram

Did you experience any problems or unexpected behavior when using the system?

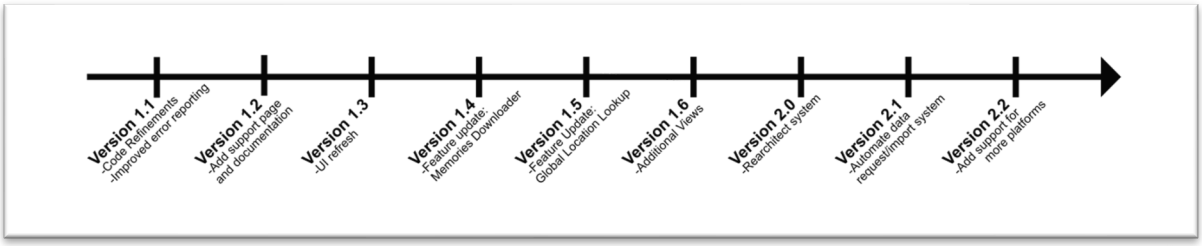
I did not encounter any unexpected problems

How would you recommend improving the system?

The interface could be mildly improve by changing spacing and visual layout of the website once on the computer screen

Any additional feedback?

System Roadmap[3]



Appendix B

Sample Input

Sample Input from Google Maps Geocode API Request


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "29707",
               "short_name" : "29707",
               "types" : [ "postal_code" ]
            },
            {
               "long_name" : "Indian Land",
               "short_name" : "Indian Land",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Lancaster County",
               "short_name" : "Lancaster County",
               "types" : [ "administrative_area_level_2", "political" ]
            },
            {
               "long_name" : "South Carolina",
               "short_name" : "SC",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "United States",
               "short_name" : "US",
               "types" : [ "country", "political" ]
            }
         ],
         "formatted_address" : "Indian Land, SC 29707, USA",
         "geometry" : {
            "bounds" : {
               "northeast" : {
                  "lat" : 35.0766659,
                  "lng" : -80.788518
               },
               "southwest" : {
                  "lat" : 34.897044,
                  "lng" : -80.91799
               }
```

```
            },
            "location" : {
               "lat" : 34.967837,
               "lng" : -80.84331450000001
            },
            "location_type" : "APPROXIMATE",
            "viewport" : {
               "northeast" : {
                  "lat" : 35.0766659,
                  "lng" : -80.788518
               },
               "southwest" : {
                  "lat" : 34.897044,
                  "lng" : -80.91799
               }
            }
         },
         "place_id" : "ChIJ_zQnAVeCVogROsdMfIGxW-8",
         "postcode_localities" : [ "Fort Mill", "Indian Land" ],
         "types" : [ "postal_code" ]
      }
   ],
   "status" : "OK"
}
```


Sample User Data Input (Input via user file import process)

account_history.json


```
{
    "Display Name Change": [
        {
            "Date": "2019-01-07 03:08:39 UTC",
            "Display Name": "Test User B"
        }
    ],
    "Email Change": [
        {
            "Date": "2019-01-07 03:15:03 UTC",
            "Email Address": "email@example.net"
        }
    ],
    "Mobile Number Change": [
        {
            "Date": "2019-01-07 03:08:56 UTC",
            "Mobile Number": "+5555555555"
```

```
        }
    ],
    "Password Change": [
        {
            "Date": "2020-03-31 21:18:34 UTC"
        }
    ],
    "Snapchat Linked to Bitmoji": [],
    "Spectacles": [],
    "Two-Factor Authentication": [
        {
            "Date": "2020-03-31 21:19:29 UTC",
            "Event": "SMS enabled"
        }
    ],
    "Account deactivated / reactivated": []
}
```


account.json


```
{
    "Basic Information": {
        "Username": "Username B",
        "Name": "Test User B",
        "Creation Date": "2019-01-07 03:08:37 UTC"
    },
    "Device Information": {
        "Make": "Apple",
        "Model ID": "iPhone11,2",
        "Model Name": "iPhone XS",
        "User Agent": "Snapchat/10.79.0.70 (iPhone11,2; iOS 13.3.1; gzip)",
        "Language": "en",
        "OS Type": "iOS",
        "OS Version": "iOS 14.4",
        "Connection Type": "WIFI, CELL"
    },
    "Device History": [
        {
            "Make": "Apple",
            "Model": "iPhone11,2",
            "Start Time": "2019-01-07 03:00:00 UTC",
            "Device Type": "PHONE"
        },
        {
            "Make": "Apple",
            "Model": "iPad5,3",
```

```
            "Start Time": "2019-10-23 03:00:00 UTC",
            "Device Type": "PHONE"
        }
    ],
    "Privacy Policy and Terms of Service Acceptance History": [],
    "Custom Creative Tools Terms": [],
    "Login History": [
        {
            "IP": "2600:1004:b08b:c6e6:185:ed6d:a917:8147",
            "Country": "US",
            "Created": "2021-03-26 00:47:57 UTC",
            "Status": "two_fa_first_step",
            "Device": "Mozilla/5.0 (iPhone; CPU iPhone OS 14_4 like Mac OS X)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Mobile/15E148 Safari/604.1"
        },
        {
            "IP": "152.38.111.116",
            "Country": "US",
            "Created": "2021-02-20 04:20:20 UTC",
            "Status": "success",
            "Device": "Snapchat/11.15.0.38 (iPhone11,2; iOS 14.4; gzip)"
        },
        {
            "IP": "152.38.111.116",
            "Country": "US",
            "Created": "2021-02-20 04:18:42 UTC",
            "Status": "success",
            "Device": "Snapchat/11.15.0.38 (iPhone11,2; iOS 14.4; gzip)"
        }
    ]
}
```


chat_history.json


```
{
    "Received Chat History": [
        {
            "From": "s.eng04",
            "Media Type": "",
            "Created": "2021-02-26 05:00:29 UTC"
        },
        {
            "From": "niela.bc",
            "Media Type": "",
            "Created": "2021-02-26 13:40:44 UTC"
        },
```

```
        {
            "From": "sarah104379",
            "Media Type": "",
            "Created": "2021-02-26 14:52:40 UTC"
        },
        {
            "From": "gav.naffy",
            "Media Type": "",
            "Created": "2021-02-26 15:41:31 UTC"
        },
        {
            "From": "no name",
            "Media Type": "",
            "Created": "2021-02-26 16:55:52 UTC"
        }
    ],
    "Sent Chat History": [
        {
            "To": "soccergal2014",
            "Media Type": "",
            "Created": "2021-02-27 02:57:35 UTC"
        },

        {
            "To": "m1lo1234",
            "Media Type": "TEXT",
            "Created": "2021-03-06 00:50:01 UTC"
        },
        {
            "To": "m1lo1234",
            "Media Type": "TEXT",
            "Created": "2021-03-06 00:49:31 UTC"
        },
        {
            "To": "m1lo1234",
            "Media Type": "TEXT",
            "Created": "2021-03-06 00:21:49 UTC"
        },
        {
            "To": "no name",
            "Media Type": "",
            "Created": "2021-03-06 01:20:21 UTC"
        },
        {
            "To": "no name",
            "Media Type": "",
            "Created": "2021-03-06 01:21:51 UTC"
```

```
        },
        {
            "To": "carolineeeblair",
            "Media Type": "TEXT",
            "Created": "2021-03-06 01:42:36 UTC"
        },
        {
            "To": "carolineeeblair",
            "Media Type": "TEXT",
            "Created": "2021-03-06 02:25:05 UTC"
        },
        {
            "To": "carolineeeblair",
            "Media Type": "TEXT",
            "Created": "2021-03-06 02:45:46 UTC"
        },
        {
            "To": "no name",
            "Media Type": "",
            "Created": "2021-03-06 03:55:24 UTC"
        },
        {
            "To": "no name",
            "Media Type": "",
            "Created": "2021-03-06 15:25:55 UTC"
        }
    ]
}
```


friends.json


```
{
    "Friends": [
        {
            "Username": "belmore59",
            "Display Name": "Barbara Elmore",
            "Timestamp": "2019-06-15 04:03:31 UTC",
            "Source": "added by suggested"
        },
        {
            "Username": "cheese123321",
            "Display Name": "Alison Keck",
            "Timestamp": "2019-03-14 03:51:13 UTC",
            "Source": "added by phone"
        },
        {
```

```
            "Username": "ssenter80",
            "Display Name": "Summer Senter",
            "Timestamp": "2019-09-29 20:44:23 UTC",
            "Source": "added by added me back"
        },
        {
            "Username": "sadcamp",
            "Display Name": "Claude Oros",
            "Timestamp": "2021-01-13 15:17:54 UTC",
            "Source": "added by username"
        },
        {
            "Username": "jefflipack",
            "Display Name": "Jeff Lipack",
            "Timestamp": "2019-03-14 03:52:04 UTC",
            "Source": "added by phone"
        },
        {
            "Username": "kennedytartt1",
            "Display Name": "Kennedy Tartt",
            "Timestamp": "2019-11-14 01:01:30 UTC",
            "Source": "added by username"
        },
        {
            "Username": "zoeyc13",
            "Display Name": "Zoey Cole",
            "Timestamp": "2019-10-01 02:48:41 UTC",
            "Source": "added by username"
        }
    ],
    "Friend Requests Sent": [
        {
            "Username": "kmart_02",
            "Display Name": "\ud83d\udd25Katie Martens",
            "Timestamp": "2021-02-13 22:21:24 UTC",
            "Source": "added by phone"
        },
        {
            "Username": "e_edwards40",
            "Display Name": "Emily Edwards",
            "Timestamp": "2021-01-03 02:21:43 UTC",
            "Source": "added by suggested"
        }
    ],
    "Blocked Users": [],
    "Deleted Friends": [
        {
```

```
            "Username": "lovablebre1245",
            "Display Name": "Bre\ud83d\udc95",
            "Timestamp": "2019-07-28 19:27:01 UTC",
            "Source": "added by added me back"
        },
        {
            "Username": "catherine_1026",
            "Display Name": "Catherine",
            "Timestamp": "2019-07-28 17:08:26 UTC",
            "Source": "added by added me back"
        },
        {
            "Username": "jmmxoxo",
            "Display Name": "Jdawg",
            "Timestamp": "2019-05-24 04:52:14 UTC",
            "Source": "added by suggested"
        }
    ],
    "Hidden Friend Suggestions": [
        {
            "Username": "showit_proud",
            "Display Name": "Show It Proud",
            "Timestamp": "",
            "Source": "added by unknown"
        },
        {
            "Username": "nirup9494",
            "Display Name": "Niru Patel",
            "Timestamp": "",
            "Source": "added by unknown"
        }
    ],
    "Ignored Snapchatters": [],
    "Shortcuts": [
        {
            "Shortcut Name": "\ud83d\udd25",
            "Created": "2020-09-01 03:13:59 UTC"
        }
    ]
}
```


location_history.json


```
{
    "Frequent Locations": [
        {
```

```
            "City": "fuquay varina",
            "Country": "usa",
            "Region": "nc"
        },
        {
            "City": "fuquay varina",
            "Country": "usa",
            "Region": "nc"
        }
    ],
    "Latest Location": [
        {
            "City": "fuquay varina",
            "Country": "usa",
            "Region": "nc"
        }
    ],
    "Home & Work": {
        "Home": "lat 35.59 \u00b1 39.09 meters, long -78.835 \u00b1 39.09 meters",
        "Work": "lat 35.407 \u00b1 78.38 meters, long -78.74 \u00b1 78.38 meters"
    },
    "Daily Top Locations": [
        [
            {
                "Date: 2021-03-22 UTC": "lat 35.589 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            },
            {
                "Date: 2021-03-22 UTC": "lat 35.59 \u00b1 39.09 meters, long -78.835
\u00b1 39.09 meters"
            },
            {
                "Date: 2021-03-22 UTC": "lat 35.589 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            },
            {
                "Date: 2021-03-22 UTC": "lat 35.59 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            }
        ]
    ],
    "Top Locations Per Six-Day Period": [
        [
            {
                "Date: 2021-02-26 UTC": "lat 35.589 \u00b1 78.17 meters, long -78.835
\u00b1 78.17 meters"
            },
```

```
            {
                "Date: 2021-02-26 UTC": "lat 35.59 \u00b1 39.09 meters, long -78.835
\u00b1 39.09 meters"
            },
            {
                "Date: 2021-02-26 UTC": "lat 35.589 \u00b1 39.09 meters, long -78.835
\u00b1 39.09 meters"
            },
            {
                "Date: 2021-02-26 UTC": "lat 35.59 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            },
            {
                "Date: 2021-02-26 UTC": "lat 35.589 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            },
            {
                "Date: 2021-02-26 UTC": "lat 35.59 \u00b1 19.54 meters, long -78.835
\u00b1 19.54 meters"
            }
        ],
        [
            {
                "Date: 2021-03-04 UTC": "lat 35.589 \u00b1 78.17 meters, long -78.835
\u00b1 78.17 meters"
            },
            {
                "Date: 2021-03-04 UTC": "lat 35.409 \u00b1 78.38 meters, long -78.74
\u00b1 78.38 meters"
            },
            {
                "Date: 2021-03-04 UTC": "lat 35.59 \u00b1 39.09 meters, long -78.835
\u00b1 39.09 meters"
            }
        ]
    ],
    "Location History": [
        {
            "Time": "2020/09/23 21:11:01 UTC",
            "Latitude, Longitude": "35.408 \u00b1 39.66 meters, -78.737 \u00b1 39.66
meters"
        },
        {
            "Time": "2020/09/23 21:14:02 UTC",
            "Latitude, Longitude": "35.408 \u00b1 39.66 meters, -78.737 \u00b1 39.66
meters"
        },
```

```
        {
            "Time": "2020/09/24 16:08:30 UTC",
            "Latitude, Longitude": "35.410 \u00b1 39.66 meters, -78.783 \u00b1 39.66
meters"
        },
        {
            "Time": "2020/09/24 16:08:33 UTC",
            "Latitude, Longitude": "35.410 \u00b1 39.66 meters, -78.783 \u00b1 39.66
meters"
        }
    ],
    "Businesses and public places you may have visited": [
        {
            "Date": "2021-01-01",
            "Name": "Dairy Queen (CA)"
        },
        {
            "Date": "2020-12-29",
            "Name": "Aldi (US)"
        },
        {
            "Date": "2021-01-17",
            "Name": "Bojangles (US)"
        },
        {
            "Date": "2020-12-12",
            "Name": "Hardee's"
        }
    ],
    "Areas you may have visited in the last two years": [
        {
            "Time": "2020/09/23 02:50:32 UTC",
            "City": "monroe",
            "Region": "North Carolina",
            "Postal Code": "27526"
        },
        {
            "Time": "2019/06/04 21:28:41 UTC",
            "City": "monroe",
            "Region": "North Carolina",
            "Postal Code": "27529"
        },
        {
            "Time": "2020/09/21 15:53:17 UTC",
            "City": "monroe",
            "Region": "North Carolina",
            "Postal Code": "27546"
```

```
        },
        {
            "Time": "2021/01/16 01:43:02 UTC",
            "City": "monroe",
            "Region": "North Carolina",
            "Postal Code": "28110"
        }
    ]
}
```


memories_history.json


```
{
    "Saved Media": [
        {
            "Date": "2021-03-13 03:28:47 UTC",
            "Media Type": "PHOTO",
            "Download Link": "https://app.snapchat.com/dmd/memories?uid=2ba1a228-94cc-
45b4-91ad-e50ffd607579&sid=4C76BD6C-0545-4808-ABA1-D758C6D06303&mid=4C76BD6C-0545-
4808-ABA1-
D758C6D06303&ts=1616720203579&proxy=true&sig=a2cebc924a0b1296fa07c94ee8901c51997a0021d
42d0d5b7c8ab02f9a07e005"
        },
        {
            "Date": "2021-03-06 01:21:47 UTC",
            "Media Type": "VIDEO",
            "Download Link": "https://app.snapchat.com/dmd/memories?uid=2ba1a228-94cc-
45b4-91ad-e50ffd607579&sid=0E3A4641-7904-4981-A9A3-FA0C510B8CEC&mid=0E3A4641-7904-
4981-A9A3-
FA0C510B8CEC&ts=1616720259105&proxy=true&sig=0e4ff320d845b294f247ff33cf600c50fe7a13ccc
8d71f68a0755f096fa24c0a"
        },
        {
            "Date": "2021-02-19 13:17:58 UTC",
            "Media Type": "PHOTO",
            "Download Link": "https://app.snapchat.com/dmd/memories?uid=2ba1a228-94cc-
45b4-91ad-e50ffd607579&sid=8E9A5A5E-E191-4F14-81C4-2AD986BE9AC2&mid=5F59337D-BE53-
43CE-9706-
A0BABF480B4D&ts=1616720286665&proxy=true&sig=096fa216b671ec29f706585832885fa219f59e325
9642b9fca2ad569ca515da2"
        }
    ]
}
```

ranking.json


```
{
    "Number of Stories Viewed": [
        0
    ],
    "Content Interests": [
        "Spa & Beauty Treatments",
        "Celebrity Stories",
        "Fun & Trivia: Lists",
        "Dogs",
        "Music",
        "Running & Jogging",
        "Desserts & Sweets",
        "Automotive",
        "Beauty & Fashion",
        "Social Media/Viral Star Stories",
        "Relationships_Stories"
    ]
}
```


search_history.json


```
[
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "ben",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "be",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "b",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "br",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
```

```
        "Search Term": "brn",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "br",
        "Location": ""
    },
    {
        "Date and time (hourly)": "2021-02-21 21:00:00 UTC",
        "Search Term": "b",
        "Location": ""
    }
]
```


snap_history.json


```
{
    "Received Snap History": [
        {
            "From": "sgymbug",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 23:37:05 UTC"
        },
        {
            "From": "nathan_ferency",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 22:09:19 UTC"
        },
        {
            "From": "niela.bc",
            "Media Type": "VIDEO",
            "Created": "2021-03-25 20:20:26 UTC"
        },
        {
            "From": "th3causer3",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 15:46:26 UTC"
        },
        {
            "From": "niela.bc",
            "Media Type": "VIDEO_NO_SOUND",
            "Created": "2021-02-23 08:45:14 UTC"
        }
    ],
    "Sent Snap History": [
```

```
        {
            "To": "sgymbug",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 23:15:41 UTC"
        },
        {
            "To": "sarah104379",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 03:25:13 UTC"
        },
        {
            "To": "sgymbug",
            "Media Type": "IMAGE",
            "Created": "2021-03-25 03:25:13 UTC"
        },
        {
            "To": "sgymbug",
            "Media Type": "VIDEO",
            "Created": "2021-03-24 01:48:12 UTC"
        }
    ],
    "App Story History": []
}
```


story_history.json


```
{
    "Your Story Views": [
    {
        "Story Date": "2020-10-09 19:00:00 UTC",
        "Story Views": 3,
        "Story Replies": 0
    },
    {
        "Story Date": "2020-10-09 14:00:00 UTC",
        "Story Views": 1,
        "Story Replies": 0
    },
    {
        "Story Date": "2020-10-09 13:00:00 UTC",
        "Story Views": 1,
        "Story Replies": 0
    },
    {
        "Story Date": "2020-10-09 12:00:00 UTC",
        "Story Views": 3,
```

```
        "Story Replies": 0
    },
    {
        "Story Date": "2020-10-09 04:00:00 UTC",
        "Story Views": 2,
        "Story Replies": 0
    }
],
    "Friend and Public Story Views": [
        {
            "View": "carolineeeblair",
            "Media Type": "STORY",
            "View Date": "2021-03-25 14:43:18 UTC"
        },
        {
            "View": "carolineeeblair",
            "Media Type": "",
            "View Date": "2021-03-25 14:43:18 UTC"
        },
        {
            "View": "niela.bc",
            "Media Type": "STORY",
            "View Date": "2021-03-23 23:29:35 UTC"
        },
        {
            "View": "niela.bc",
            "Media Type": "",
            "View Date": "2021-03-23 23:29:35 UTC"
        }
    ]
}
```


subscriptions.json


```
{
    "Public Users": [
        "why.exsit"
    ],
    "Publishers": [],
    "Stories": [],
    "Last Active Timezone": "America/New_York",
    "Push Notifications": [],
    "Hidden Category Sections": []
}
```

talk_history.json


```
{
    "Outgoing Calls": [],
    "Incoming Calls": [
        {
            "Date & Time": "2020-12-02 18:06:25 UTC",
            "Type": "VIDEO",
            "People in Chat": 2,
            "Result": "Call Received",
            "City": "fuquay varina",
            "Country": "US",
            "Length (sec)": 30,
            "Network": "WIFI"
        },
        {
            "Date & Time": "2021-01-07 18:15:28 UTC",
            "Type": "VIDEO",
            "People in Chat": 2,
            "Result": "Call Failed",
            "City": "charlotte",
            "Country": "US",
            "Length (sec)": 4,
            "Network": "MOBILE"
        }
    ],
    "Completed Calls": [],
    "Chat Sessions": [],
    "Game Sessions": []
}
```


user_profile.json


```
{
    "App Profile": {
        "Country": "US",
        "Creation Time": "2019-01-07 03:08:37 UTC",
        "Account Creation Country": "US",
        "Platform Version": "IOS",
        "In-app Language": "en"
    },
    "Demographics": {
        "Cohort Age": "",
        "Derived Ad Demographic": ""
    },
    "Subscriptions": [],
```

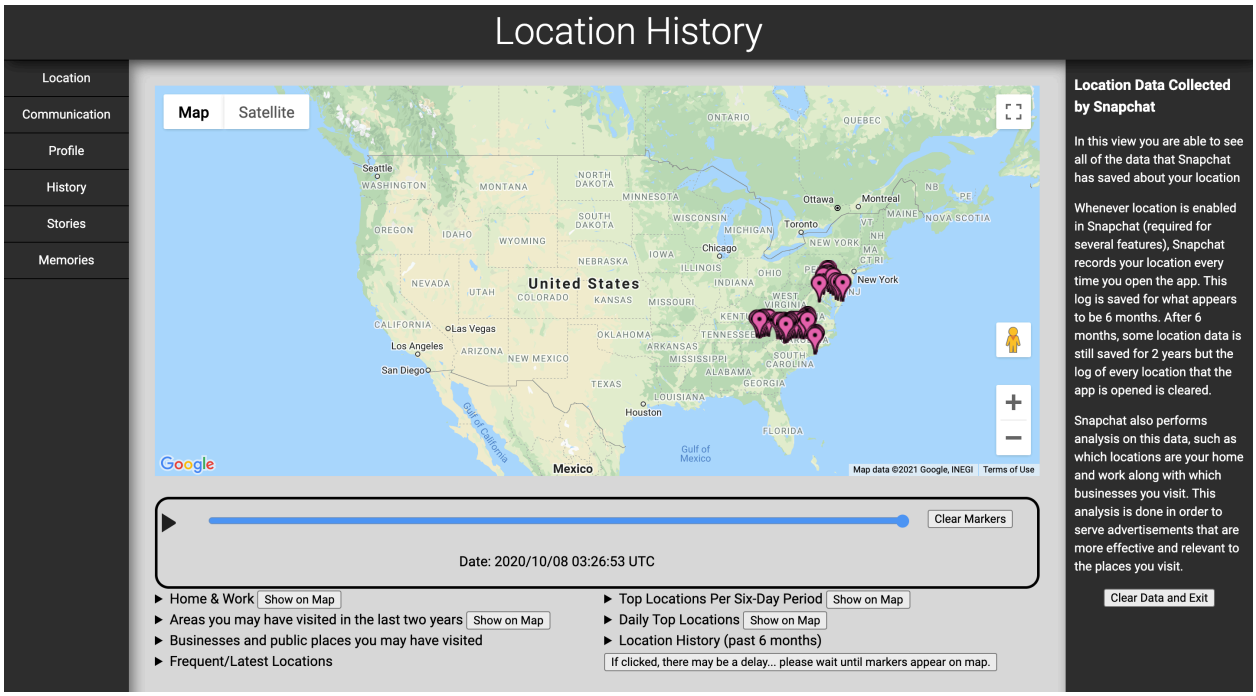
```
"Engagement": [
    {
        "Event": "Geolens Swipes",
        "Occurrences": 3
    },
    {
        "Event": "Direct Snaps Created",
        "Occurrences": 107
    },
    {
        "Event": "Chats Sent",
        "Occurrences": 138
    },
    {
        "Event": "Snap Sends",
        "Occurrences": 75
    },
    {
        "Event": "Story Ads Viewed",
        "Occurrences": 8
    },
    {
        "Event": "Application Opens",
        "Occurrences": 459
    },
    {
        "Event": "Story Views",
        "Occurrences": 45
    },
    {
        "Event": "Snaps Viewed in a Story",
        "Occurrences": 95
    },
    {
        "Event": "Geofilter Story Snaps Viewed",
        "Occurrences": 4
    },
    {
        "Event": "Chats Viewed",
        "Occurrences": 115
    },
    {
        "Event": "Direct Snaps Viewed",
        "Occurrences": 25
    },
    {
        "Event": "Snap Views",
```

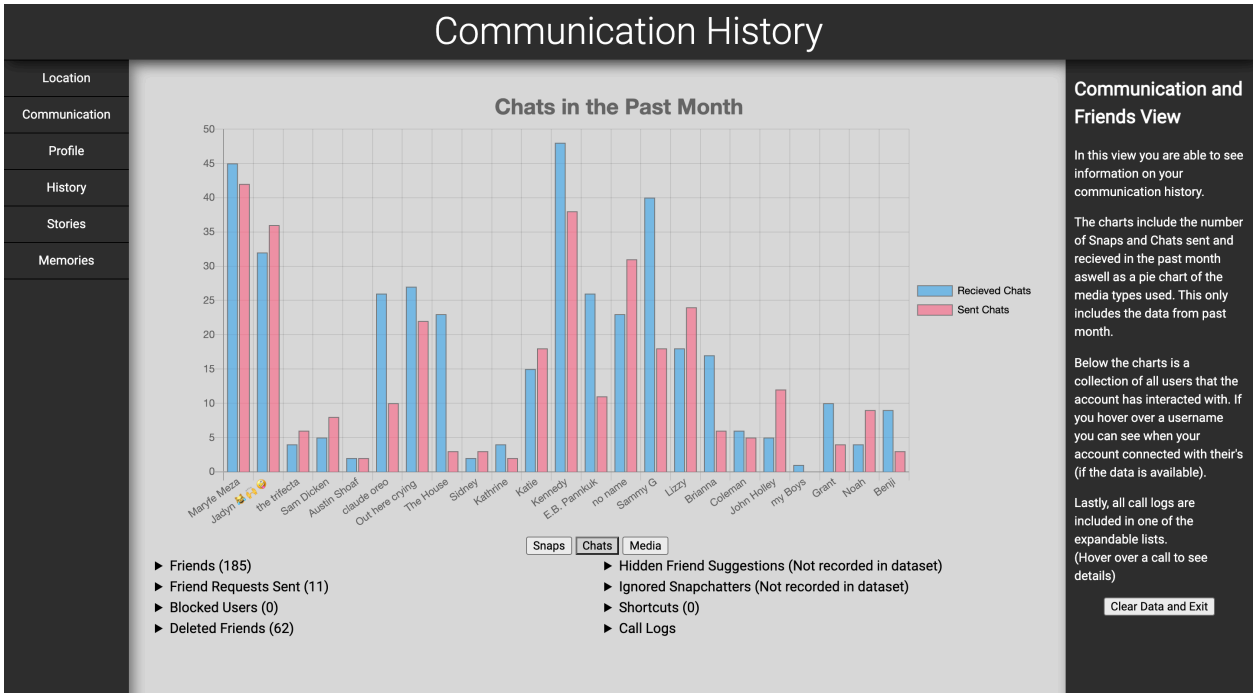
```
            "Occurrences": 280
        }
    ],
    "Discover Channels Viewed": [],
    "Breakdown of Time Spent on App": [
        "Camera: 8.04%",
        "Discover: 0.00%",
        "Map: 0.73%",
        "Memories: 0.02%",
        "Messaging: 8.34%",
        "Others: 78.32%",
        "Profile: 4.55%"
    ],
    "Ads You Interacted With": [
        {
            "Advertiser Name": "My.com B.V.",
            "Date": "2020-06-02"
        },
        {
            "Advertiser Name": "Hulu",
            "Date": "2020-10-09"
        }
    ],
    "Interest Categories": [
        "Collegiates",
        "Big Box Stores",
        "Big Box Store Buyers",
        "Techies & Gadget Fans",
        "Luxury Hotels",
        "Health Clubs & Store Buyers - Weight Watchers",
        "eCommerce Buyers - Amazon - All Buyers",
        "Jewelry Store Buyers - All Buyers",
        "Insurance Buyers - Metlife",
        "Airline Buyers - Light Buyers",
        "Credit & Rewards Card Buyers - Credit Revolvers",
        "Indie & Foreign Film Fans",
        "Department Store Shoppers",
        "TV Viewers (Christmas Specials)",
        "Last 12 Months Buyers - Walmart",
        "TV Network Viewers (Comedy Central)",
        "Telecom Buyers - ATT Products or Services",
        "Olive Garden Restaurants",
        "Fast Food Buyers - Little Caesars",
        "Travel Services Buyers - Expedia",
        "Car Part Buyers - All Car Parts"
    ],
    "Geographic Information": [],
```

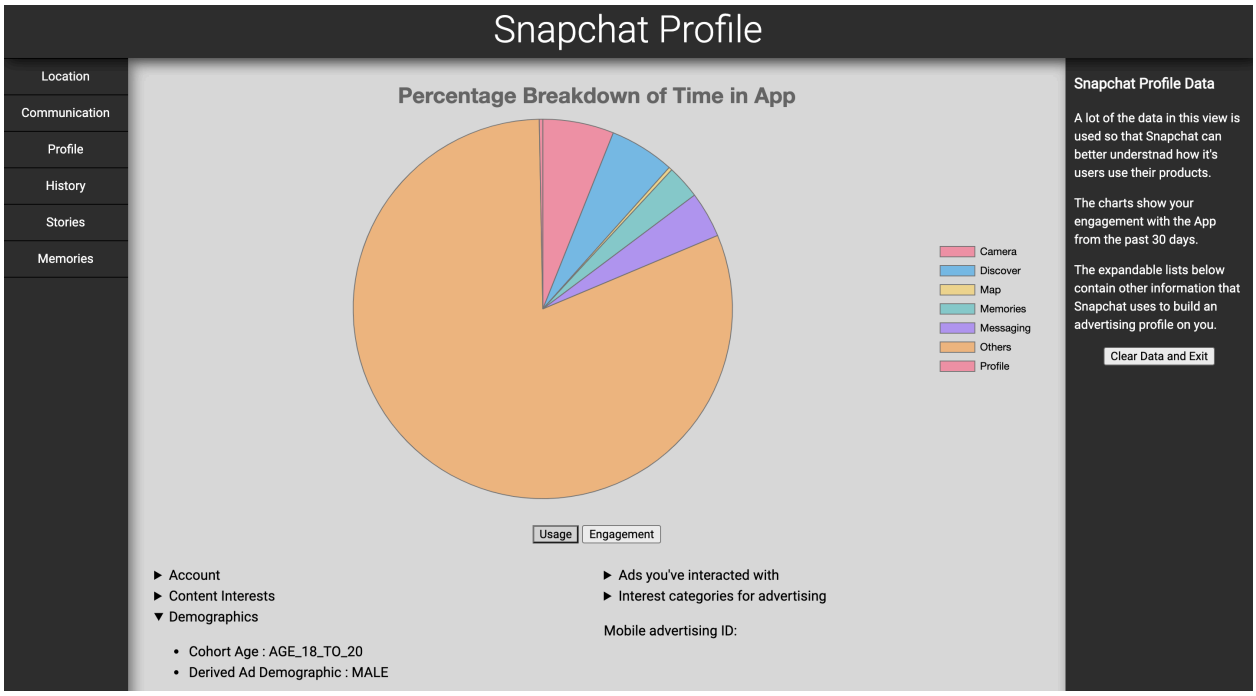
```
    "Interactions": {
        "Web Interactions": [
            "subway.com",
            "footaction.com",
            "timberland.com",
            "paramountplus.com",
            "ingrammicro.com",
            "patagonia.com",
            "mikesbikes.com",
            "monagiza.com",
            "windsorstore.com",
            "fashionnova.com",
            "graduatez.com",
            "healthyway.com",
            "crateandbarrel.com",
            "vineyardvines.com"
        ],
        "App Interactions": [
            "https://itunes.apple.com/us/app/id1473663873",
            "https://itunes.apple.com/us/app/id324684580",
            "https://itunes.apple.com/us/app/id1459260306",
            "https://itunes.apple.com/us/app/id472014516",
            "https://itunes.apple.com/us/app/id692365393",
            "https://itunes.apple.com/us/app/id546473125",
            "https://itunes.apple.com/us/app/id1193508329"
        ]
    },
    "Mobile Ad Id": "4b520a08-6434-483c-bef7-9bcf6e665b48"
}
```

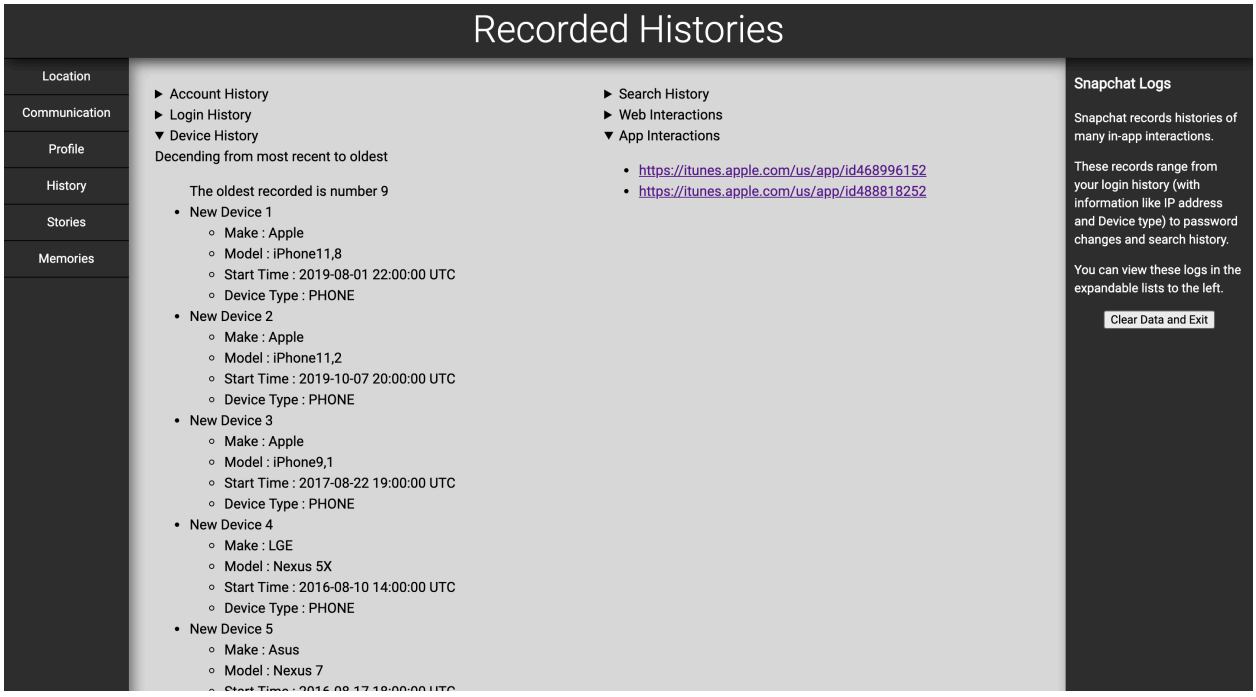
Appendix C

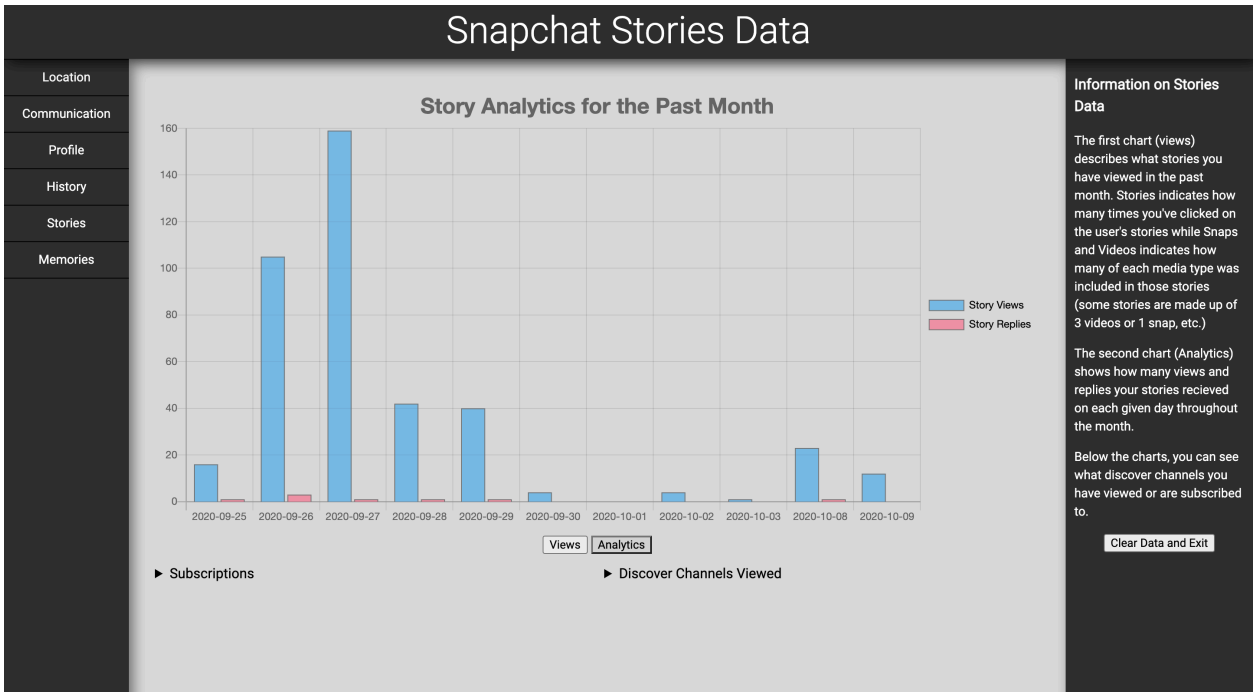
Sample Output

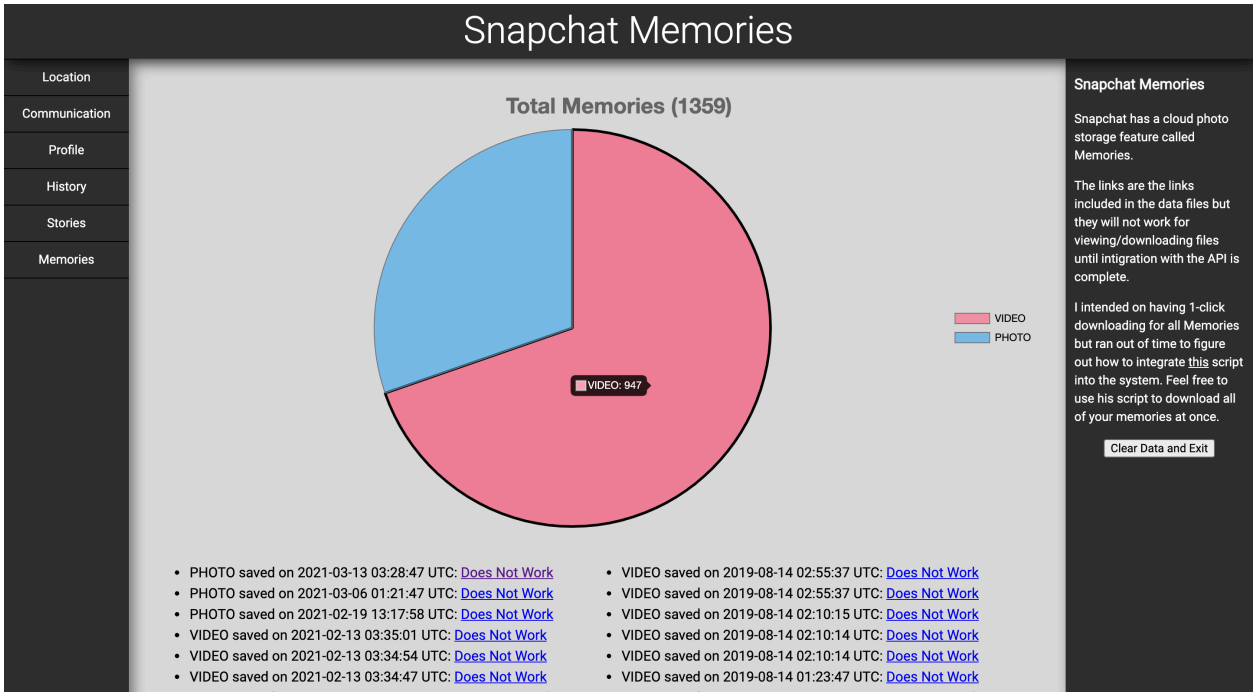





Snapchat Profile
Location
Communication
Profile
History
Stories
Memories
Percentage Breakdown of Time in App
Camera
Discover
Map
Memories
Messaging
Others
Profile
Usage
Engagement
▶ Account
▶ Content Interests
▼ Demographics
• Cohort Age : AGE_18_TO_20
• Derived Ad Demographic : MALE
▶ Ads you've interacted with
▶ Interest categories for advertising
Mobile advertising ID:
Snapchat Profile Data
A lot of the data in this view is used so that Snapchat can better understnad how it's users use their products.
The charts show your engagement with the App from the past 30 days.
The expandable lists below contain other information that Snapchat uses to build an advertising profile on you.
Clear Data and Exit


Recorded Histories
Location
Communication
Profile
History
Stories
Memories
▶ Account History
▶ Login History
▼ Device History
Decending from most recent to oldest
The oldest recorded is number 9
• New Device 1
◦ Make : Apple
◦ Model : iPhone11,8
◦ Start Time : 2019-08-01 22:00:00 UTC
◦ Device Type : PHONE
• New Device 2
◦ Make : Apple
◦ Model : iPhone11,2
◦ Start Time : 2019-10-07 20:00:00 UTC
◦ Device Type : PHONE
• New Device 3
◦ Make : Apple
◦ Model : iPhone9,1
◦ Start Time : 2017-08-22 19:00:00 UTC
◦ Device Type : PHONE
• New Device 4
◦ Make : LGE
◦ Model : Nexus 5X
◦ Start Time : 2016-08-10 14:00:00 UTC
◦ Device Type : PHONE
• New Device 5
◦ Make : Asus
◦ Model : Nexus 7
▶ Search History
▶ Web Interactions
▼ App Interactions
• https://itunes.apple.com/us/app/id468996152
• https://itunes.apple.com/us/app/id488818252
Snapchat Logs
Snapchat records histories of many in-app interactions.
These records range from your login history (with information like IP address and Device type) to password changes and search history.
You can view these logs in the expandable lists to the left.
Clear Data and Exit


Snapchat Stories Data
Location
Communication
Profile
History
Stories
Memories
Story Analytics for the Past Month
160
140
120
100
80
60
40
20
0
2020-09-25
2020-09-26
2020-09-27
2020-09-28
2020-09-29
2020-09-30
2020-10-01
2020-10-02
2020-10-03
2020-10-08
2020-10-09
Story Views
Story Replies
Views
Analytics
▶ Subscriptions
▶ Discover Channels Viewed
Information on Stories Data
The first chart (views) describes what stories you have viewed in the past month. Stories indicates how many times you've clicked on the user's stories while Snaps and Videos indicates how many of each media type was included in those stories (some stories are made up of 3 videos or 1 snap, etc.)
The second chart (Analytics) shows how many views and replies your stories recieved on each given day throughout the month.
Below the charts, you can see what discover channels you have viewed or are subscribed to.
Clear Data and Exit


Snapchat Memories
Location
Communication
Profile
History
Stories
Memories
Total Memories (1359)
VIDEO: 947
VIDEO
PHOTO
• PHOTO saved on 2021-03-13 03:28:47 UTC: Does Not Work
• PHOTO saved on 2021-03-06 01:21:47 UTC: Does Not Work
• PHOTO saved on 2021-02-19 13:17:58 UTC: Does Not Work
• VIDEO saved on 2021-02-13 03:35:01 UTC: Does Not Work
• VIDEO saved on 2021-02-13 03:34:54 UTC: Does Not Work
• VIDEO saved on 2021-02-13 03:34:47 UTC: Does Not Work
• VIDEO saved on 2019-08-14 02:55:37 UTC: Does Not Work
• VIDEO saved on 2019-08-14 02:55:37 UTC: Does Not Work
• VIDEO saved on 2019-08-14 02:10:15 UTC: Does Not Work
• VIDEO saved on 2019-08-14 02:10:14 UTC: Does Not Work
• VIDEO saved on 2019-08-14 02:10:14 UTC: Does Not Work
• VIDEO saved on 2019-08-14 01:23:47 UTC: Does Not Work
Snapchat Memories
Snapchat has a cloud photo storage feature called Memories.
The links are the links included in the data files but they will not work for viewing/downloading files until intigration with the API is complete.
I intended on having 1-click downloading for all Memories but ran out of time to figure out how to integrate this script into the system. Feel free to use his script to download all of your memories at once.
Clear Data and Exit

Appendix D

Processing Logic: Source Code for Release Version 1.0

https://github.com/RickyHaase/PersonalPortfolio

Appendix E

Templates

HTML Template for Views

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>View Title</title>
    <link rel="icon" href="#" type="image/gif" sizes="16x16" />
    <link
      href="https://fonts.googleapis.com/css?family=Roboto:300,400"
      rel="stylesheet"
    />
    <link rel="stylesheet" type="text/css" href="../Assets/MainView.css" />
  </head>

  <body>
    <div class="header">Page Title</div>

    <ul class="sidenav">
      <li><a href="../Location View/Location.html">Location</a></li>
      <li>
        <a href="../Communication View/Communication.html">Communication</a>
      </li>
      <li><a href="../Profile View/Profile.html">Profile</a></li>
      <li><a href="../History View/History.html">History</a></li>
      <li><a href="../Stories View/Stories.html">Stories</a></li>
      <li><a href="../Shop View/Shop.html">Shop</a></li>
      <li><a href="../Memories View/Memories.html">Memories</a></li>
    </ul>

    <div class="infoPane">
      <div class="info">
        <h3>Words Words</h3>
        <p>Words words words...</p>
        <div class="buttons">
          <a href="../index.html">
            <button onclick="localStorage.clear()">
              Clear Data and Exit
            </button></a>
        </div>
      </div>
    </div>
```

```
    <div class="content">
      <script src="Template.JS"></script>
    </div>
  </body>
</html>
```

JavaScript template for Chart.js Object

(from Traversey, Getting Started with Charts.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.6.0/Chart.min.js"></script>
    <title>My Chart.js Chart</title>
  </head>
  <body>
    <div class="container">
      <canvas id="myChart"></canvas>
    </div>
    <script>
      let myChart = document.getElementById('myChart').getContext('2d');
      // Global Options
      Chart.defaults.global.defaultFontFamily = 'Lato';
      Chart.defaults.global.defaultFontSize = 18;
      Chart.defaults.global.defaultFontColor = '#777';

      let massPopChart = new Chart(myChart, {
        type: 'bar', // bar, horizontalBar, pie, line, doughnut, radar, polarArea
        data: {
          labels: [
            'Boston',
            'Worcester',
            'Springfield',
            'Lowell',
            'Cambridge',
            'New Bedford',
          ],
          datasets: [
            {
              label: 'Population',
```

```
            data: [617594, 181045, 153060, 106519, 105162, 95072],
            //backgroundColor:'green',
            backgroundColor: [
              'rgba(255, 99, 132, 0.6)',
              'rgba(54, 162, 235, 0.6)',
              'rgba(255, 206, 86, 0.6)',
              'rgba(75, 192, 192, 0.6)',
              'rgba(153, 102, 255, 0.6)',
              'rgba(255, 159, 64, 0.6)',
              'rgba(255, 99, 132, 0.6)',
            ],
            borderWidth: 1,
            borderColor: '#777',
            hoverBorderWidth: 3,
            hoverBorderColor: '#000',
          },
        ],
      },
      options: {
        title: {
          display: true,
          text: 'Largest Cities In Massachusetts',
          fontSize: 25,
        },
        legend: {
          display: true,
          position: 'right',
          labels: {
            fontColor: '#000',
          },
        },
        layout: {
          padding: {
            left: 50,
            right: 0,
            bottom: 0,
            top: 0,
          },
        },
        tooltips: {
          enabled: true,
        },
      },
    });
  </script>
 </body>
</html>
```

Template for Google Maps API Integration

(from Google Developers, Maps)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Map</title>
    <script src="https://polyfill.io/v3/polyfill.min.js?features=default"></script>
    <style type="text/css">
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }
      /* Optional: Makes the sample page fill the window. */
      html,
      body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
    <script>
      let map;

      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: { lat: -34.397, lng: 150.644 },
          zoom: 8,
        });
      }
    </script>
  </head>
  <body>
    <div id="map"></div>

    <!-- Async script executes immediately and must be after any DOM elements used in
callback. -->
    <script
src=https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap&librarie
s=&v=weekly async> </script>

  </body>
</html>
```

Template for Geocode API Request

(from Google Developers, Geocode)

```
https://maps.googleapis.com/maps/api/geocode/outputFormat?parameters
```

Example API Request from Project:

```
https://maps.googleapis.com/maps/api/geocode/json?components=postal_code:29707&key=AIzaSyCWBiNjlecS2ZLkEWnuUE4_xa9v2CvwMXw
```

References

Chart.js Contributors. (n.d.). Chart.js Documentation. Retrieved February 2021, from https://www.chartjs.org/docs/latest/

Google Developers. (n.d.). Google Geocoding API Documentation. Retrieved February 2021, from https://developers.google.com/maps/documentation/geocoding/overview

Google Developers. (n.d.). Google Maps JavaScript API Documentation. Retrieved February 2021, from https://developers.google.com/maps/documentation/javascript/overview

Mozilla Corp. (n.d.). JavaScript Reference. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Snap Inc. (n.d.). Snapchat Privacy Center. Retrieved February 2021, from https://www.snap.com/en-US/privacy/privacy-center

Traversey, B. (2017, June 12). Google Maps JavaScript API Tutorial. Retrieved February 2021, from https://www.youtube.com/watch?v=Zxf1mnP5zcw&t=1482s

Traversey, B. (2017, June 19). Getting Started with Chart.js. Retrieved February 2021, from https://www.youtube.com/watch?v=sE08f4iuOhA&t=621s

Valacich, J. S., & George, J. F. (2018). Modern Systems Analysis & Design (9th ed.). Boston, MA: Pearson.

Footnotes

[1] This assumption was found to be not true late in testing (April 9, 2021). Some user data files are too large to fit in browser local storage and thus cannot be imported into the system.

[2] See System Support Documentation for procedures on how to proceed when this message occurs.

[3] See “Future Enhancements” section for more details on each system improvement.