Interactive Museum Timeline
4P02 Progress Report 2

COSC 4P02
Software Engineering II
April 30, 2023

Instructor: Nasser Ezzati-Jivan
Brock University

Alex Duclos
ad19ua@brocku.ca
6738884

Eduardo Saldana
es18za@brocku.ca
6612626

Goktug Cirag
gc19cy@brocku.ca
6776678

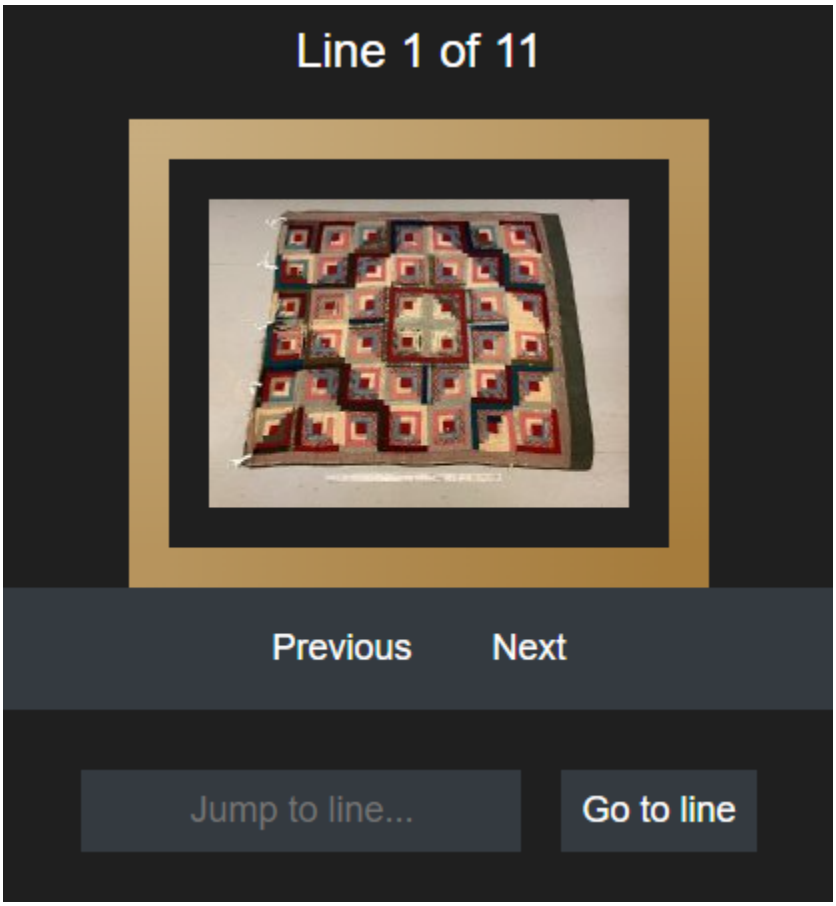
Jashan Pannu
jp18jj@brocku.ca
6505861

Rishabh Rai
rr19pa@brocku.ca
6847156

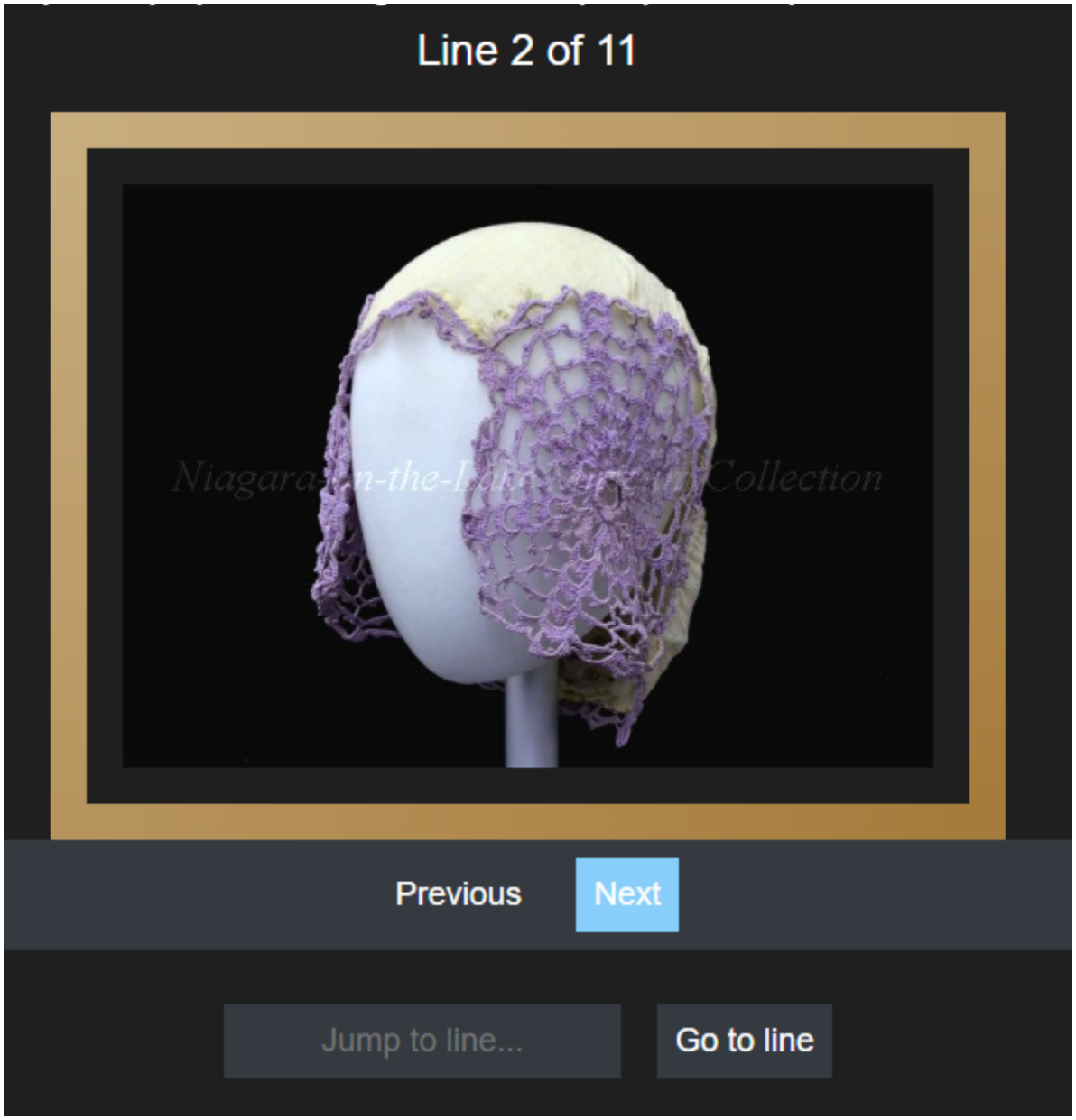
1. User Manual

Visit a URL the webpage is hosted on:
https://htmlpreview.github.io/?https://github.com/Rishabh9742/COSC4P02/blob/main/Web/index.html

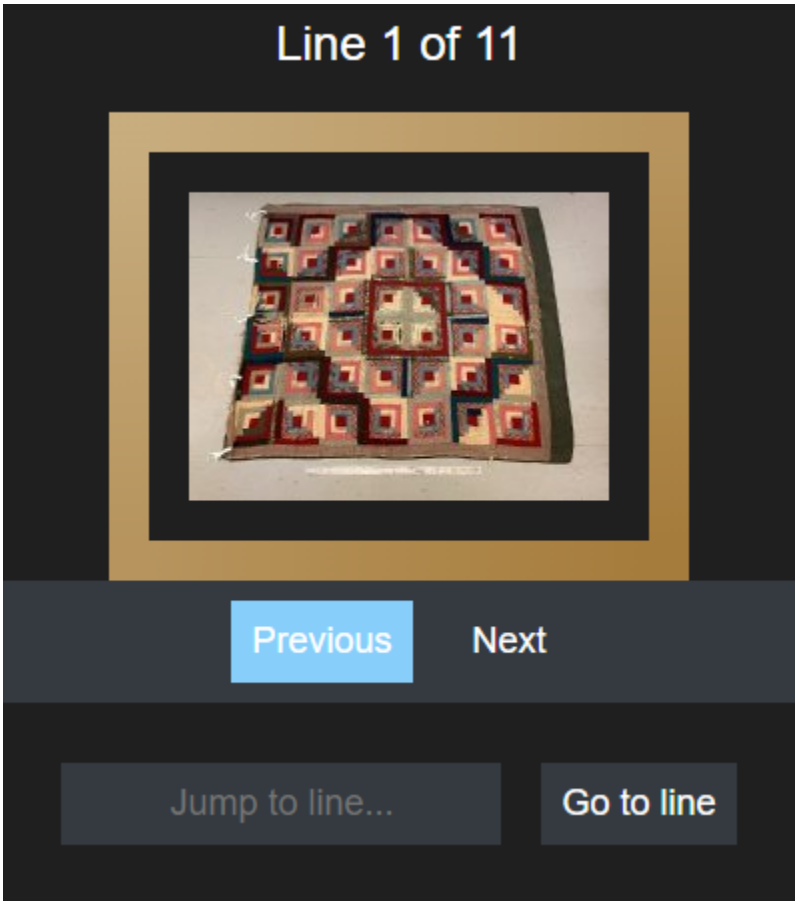
Node traversal functions: One of the main components of the website which allows the user to go to navigate different NOTL artifacts.



Next button: Takes the user to the next artifact in the JSON file.



Previous Button: Takes the user to the previous artifact in the JSON file.



Jump to line function: Allows the user to go to any point in the timeline without having to navigate through them individually..

Line 1 of 11

Previous Next

7 Go to line

Line 7 of 11

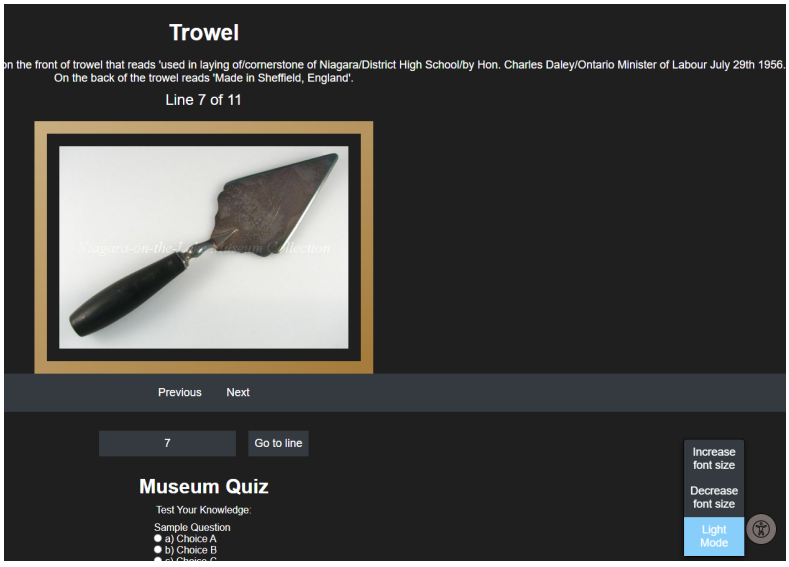
Previous Next

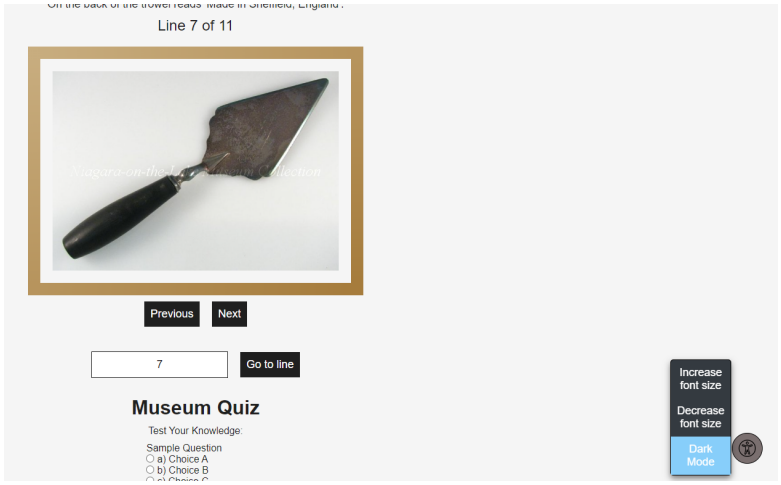
7 Go to line

Old Accessibility button

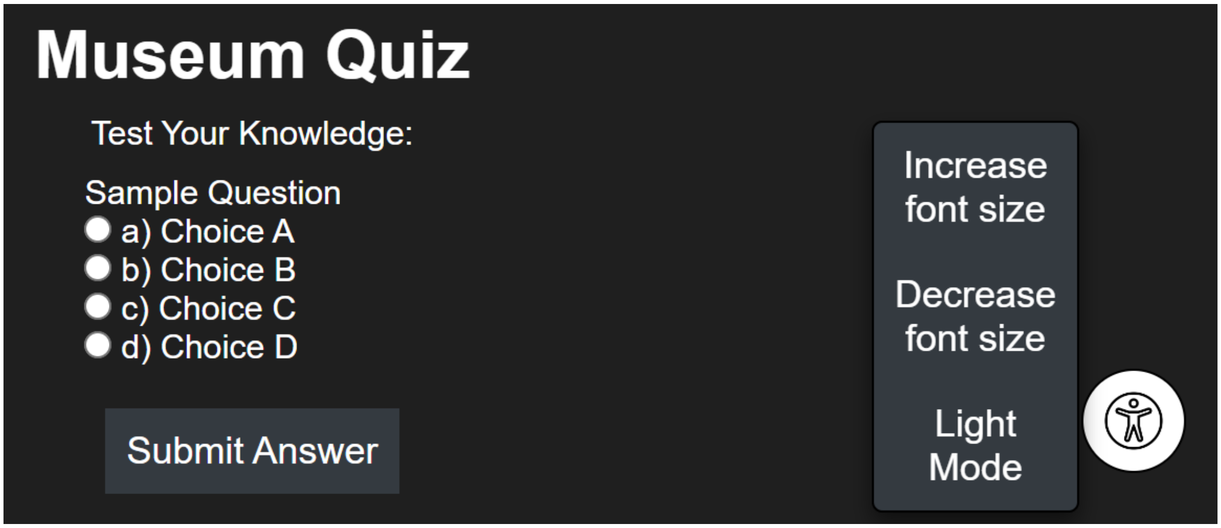
Old Accessibility menu

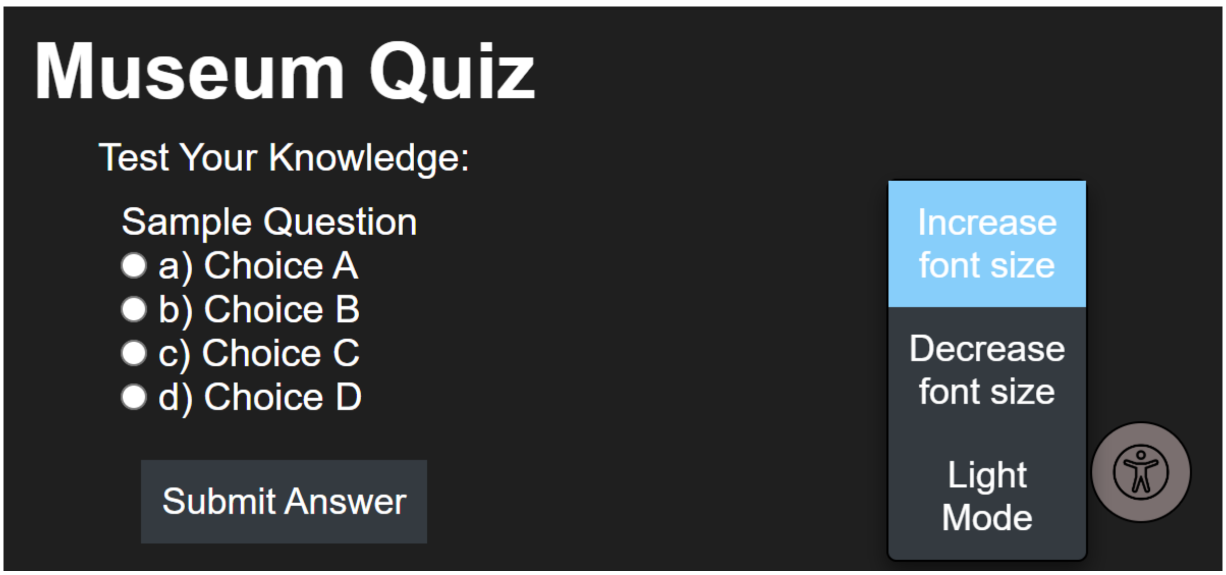
Light/Dark modes: Display option that lets the user choose a color scheme for the website with the purpose to customize the viewing experience to be more comfortable for the user.. 2 options available, dark scheme of colors and light scheme of colors.

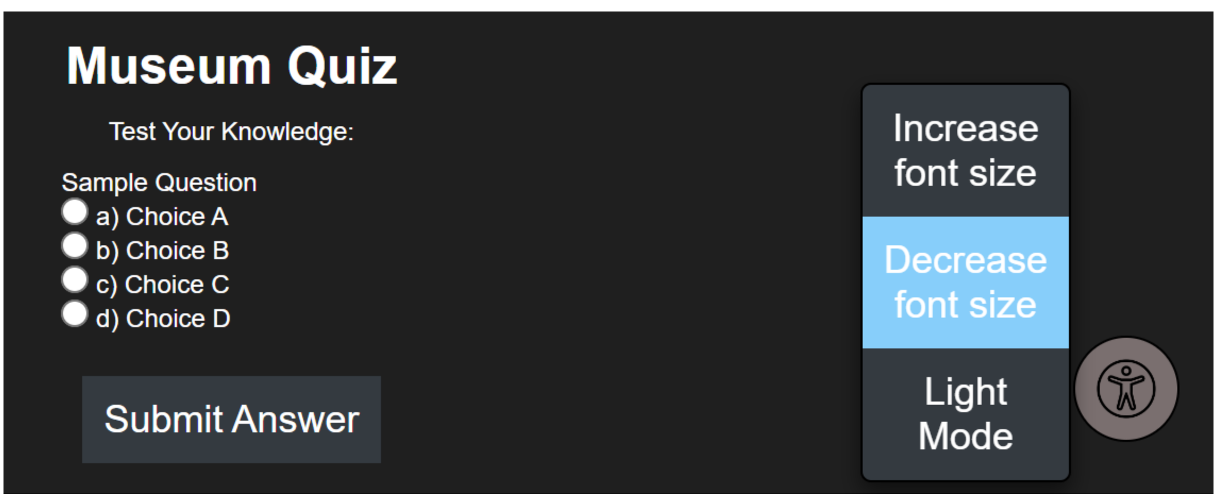




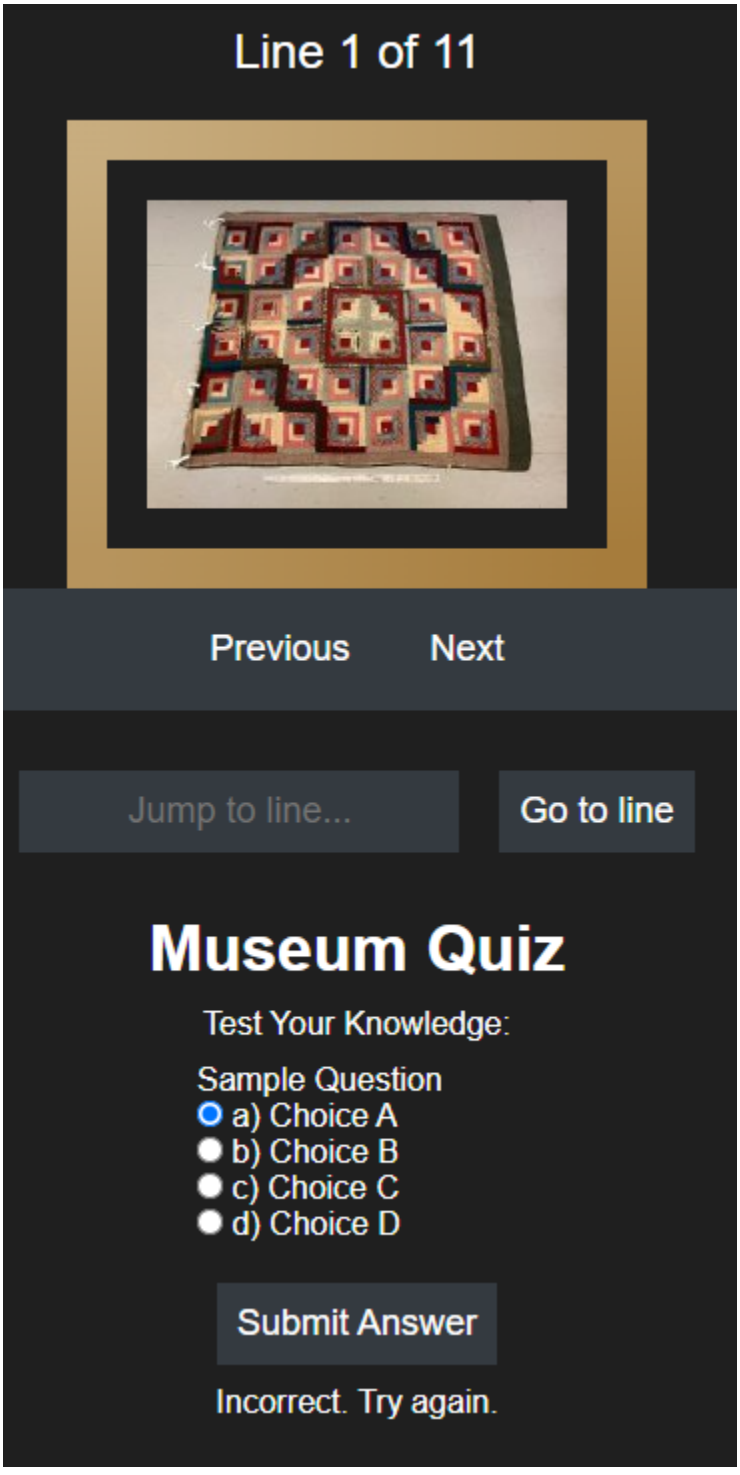
Font size changes: Accessibility option to allow the user to change the font size for better readability.

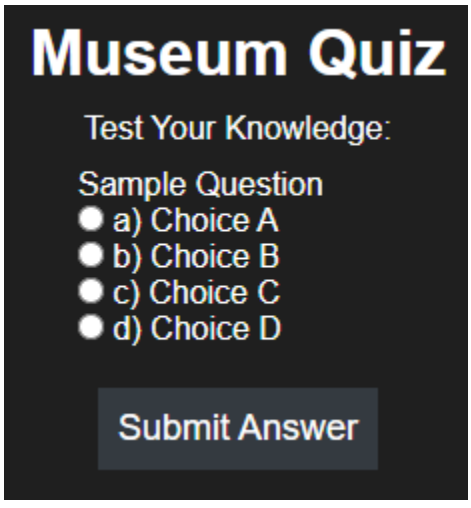






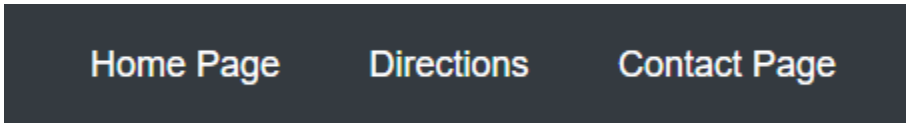
Quiz: Each artifact has its respective question and multiple answer options (currently experiencing issues with the website fetching the question/answer). Clicking the”Submit Answer” button provides feedback on the answer.





Nav bar: Allows the user to travel to any of the pages by clicking on them, it's always located at the top. The order always stays the same and takes the use to either the “Home Page”,

“Directions” and “Contact Page”



2. Software Documentation

2.1 Main and GUI

The code defines the following GUI elements:

- A main window with the title "CSV Data Appender"
- A label widget for displaying the text "Select a JSON file:"
- A label widget for displaying the path to the selected JSON file
- A button widget for browsing for a JSON file
- A label widget for displaying the text "Title:"
- An entry widget for inputting the title data
- A label widget for displaying the text "Description:"
- An entry widget for inputting the description data
- A label widget for displaying the text "Image URL:"
- An entry widget for inputting the image_url data
- A button widget for clearing the text fields
- A button widget for appending data to the selected JSON file
- A button widget for deleting a row from the selected JSON file
- An entry widget for specifying the row to delete
- A button to go to the next element in the JSON file
- A button to go to the previous element in the JSON file

2.2 Timeline Class

Definition:

`class timeline:`

The timeline class is an interface for creating and managing a timeline containing event nodes.

Variables:

`linkedList`: Linked List to contain events created using the `LinkedList` class

`currentNode`: The currently referenced node

`index`: The index of currentNode within the linkedList

Methods:

`__init__(self) -> None`

The constructor of the timeline class

`startTimeline(self) -> None`

Sets currentNode to head of linkedList and index variable to 0

`nextEvent(self) -> None`

Sets currentNode to next in the linkedList and increases index by 1

`previousEvent(self) -> None`

Sets currentNode to previous node in the linkedList and decreases index by 1

`goTo(self, index: int) -> None`

Sets currentNode to node at parameter index value in linked list and sets class index to parameter index value

`getTitle(self) -> str`

Returns title string contained by node referenced by currentNode

`getSummary(self) -> str`

Returns summary string contained by node referenced by currentNode

`getDate(self) -> str`

Returns date string contained by node referenced by currentNode

`getFacts(self) -> str`

Returns facts string contained by node referenced by currentNode

`setTitle(self) ->(self, title: str) -> None`

Sets title string contained by node referenced by currentNode to received parameter title string

`setSummary(self, summary: str) -> None`
Sets summary string contained by node referenced by currentNode to received parameter summary string

`setDate(self, date: str) -> None`
Sets date string contained by node referenced by currentNode to received parameter date string

`setFacts(self, facts: str) -> None`
Sets facts string contained by node referenced by currentNode to received parameter facts string

`addEvent(self, title: str, summary: str, date: str, facts: str) -> None`
Adds new node to linkedList constructed with received parameters title, summary, date and facts

`deleteEvent(self) -> None`
Deletes currentNode from linkedList

2.3 Linked List Class

Definition:

`class LinkedList:`
The LinkedList class serves as a linked list data structure with functions for containing and managing Node class instances.

Variables:

`head`: First node of the linkedList structure

Methods:

`addNode(self, node: Node, pos: int = -1) -> None`
Adds received parameter node into the linked list at parameter position value.

`addNewNode(self, title: str, date: str, summary: str, facts: str, pos: int = -1) -> None`
Creates a new node using received title, date, summary, and facts string parameters and adds the node into the linked list at parameter position value.

`insertAtLast(self, node: Node) -> None`
Inserts parameter node at the end of the linked list.

`deleteLast(self) -> None`
Deletes last node in linked list

`delete(self, pos: int = -1) -> None`
Deletes node at parameter position value

`reorder(self, node, pos) -> None`
Moves parameter node from current position in linked list to parameter position.

`get(self,pos) -> Node`
Returns node at parameter position.

`length(self)`
Returns total number of nodes in linked list.

`rate_node(node_name, rating)`
Appends parameter rating to ratings of node with parameter node name.

`**submit_review(node_name, review)**`
Appends parameter review to reviews of node with parameter node name.

2.4 Node Class

Definition:

`class Node:`

The Node class is the representation of an event, containing event related data

Variables:

`title`: The title of the event

`date`: The date the event took place

`summary`: A summary of the event

`facts`: Facts about the event

`next`: The next event node

Methods:

`deleteSummary(self)`

Deletes the current summary

`addSummary(self, new_summary: str)`

Replaces the current summary with parameter new summary

`addFacts(self, new_facts: str)`

Replaces the current facts with parameter new facts

`deleteFacts(self)`

Deletes the current facts

`deleteTitle(self)`

Deletes the current title

`displayInfo(self)`

Prints title, date, summary, and facts information about the node to console

3. Software License

MIT License

3. Contributions

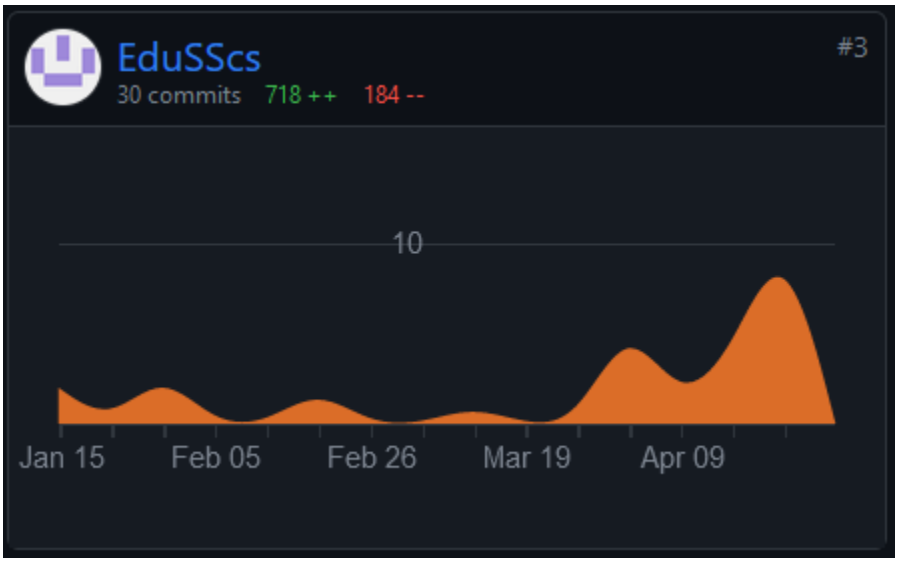
3.1 Alex Duclos - Product Owner

Provided MVP for the majority of sections of the assignment, including initial python prototype, first web build, as well as managing the requirements of the project.

Provided technological assistance as well as managed requirement understanding. Selected technologies for use in the project,

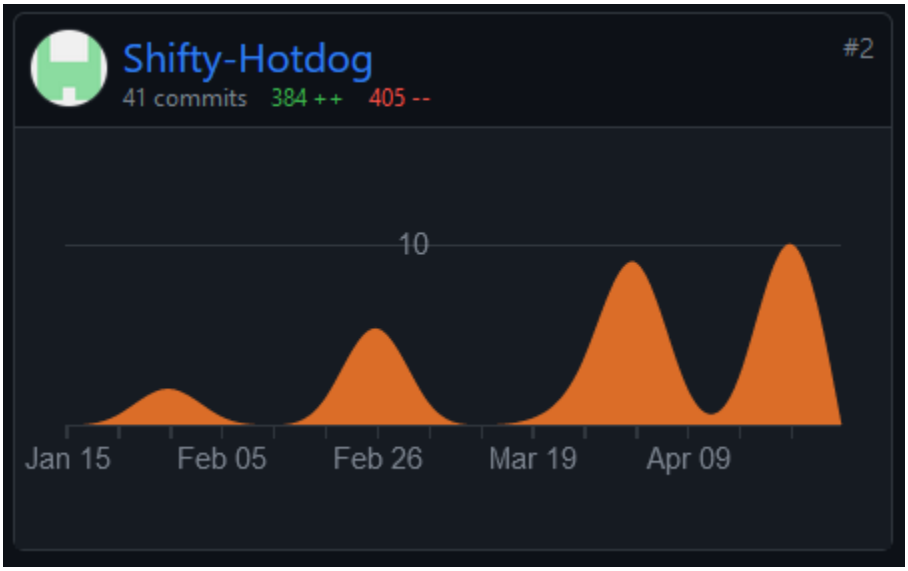
3.2 Eduardo Saldana - Developer

- Updated the JSON file to include artifacts from the NOTL museum including their name, description, images and their respective questions.
- Created a directions page which include a google map location to the museum. The address, their phone number, how accessible they are and some pictures.
- Created the Contact page and filled it with the contact information.
- Updated the CSS and JS files.
- Made the upper part of the website available to used to navigate website.
- Updated some HTML tags for a more semantic code and understandability.
- Added CSS code for mobile only view.



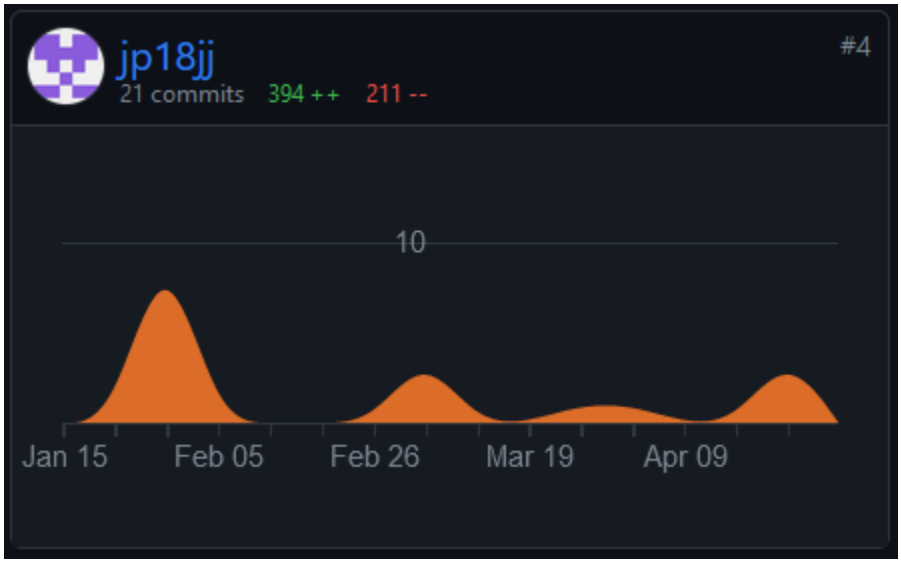
3.3 Goktug Cirag - Developer

- Added delete row button to original csv_edit.py file (later changed and moved to JSON_edit.py by other group members as I did not work on the JSON_edit.py file)
- Added jump to button in original non-website version of app (main.py)
- Fixed layout of the website
- Added a highlight when user mouses over parts of the website, such as buttons or text boxes
- Added a light mode to the website
- Added a lightmode/darkmode toggle button
- Fixed the accessibility button and added onto it.
- Made the accessibility button toggle a menu.
- Moved light mode toggle functionality to this accessibility menu
- Added a font size increase option to the accessibility menu
- Added a font size decrease option to the accessibility menu



3.4 Jashan Pannu - Scrum Master

- Worked on initial software requirements engineering and subsequent refinement
- Designed and managed the product and sprint backlogs
- Organized and recorded meeting notes to construct progress reports and presentations
- Planned initial implementation ideas of timeline structure
- Aided in researching and testing technologies for use in project
- Developed timeline class for use as object
- Created test class for linkedList structure
- Performed testing on linkedList structure to identify and resolve bugs
- Wrote software documentation for timeline, linked list, and node classes



3.5 Rishabh Rai - Developer

- Implemented a file management system to handle all viewable media for each node.
- Created a function to add or remove media files from nodes.
- Implemented a system for interacting with and traversing the timeline by traveling through nodes.
- Created input functions for "previous", "next", and received a list of all nodes.
- Added a function for selecting the current node to view its contents.
- Added a feature for users to rate and submit how interesting they find the node**.**
- Developed the timeline and node.
- Made the accessibility button on the website which was not working properly so i decided to add the accessibility button using a third party system called “EqualWeb”.