Communication Labs

Spring 2023

Assignment 2

“Before the coffee gets cold”

Interactive Comic

Prepared by:
Janindu Nanayakkara (jin3303)
Lincoln Adomako (lna279)
Swostik Pati (sp6441)
Aaron Wajah(anw2034)

Instructor:
Dr Evi Mansor

Table of contents

LIST OF FIGURES

1. INTRODUCTION

To love is to hurt. When we as humans explore the raw and quite primal emotion known as love, we have to be repaid to pay the price of how it fails to stay up to our expectation of it being perfect and even more deeply hurting, losing it. We decided to explore the beauty of this emotion through our story.

1.1 Project Name

We choose the name “Before the coffee gets cold” in respect to the Toshikazu Kawaguchi novel of the same title that also explores the theme of Love.

1.2 Project Description

The story of the project is that of an introverted male antagonist who finds himself talking to another female introvert one day at the cafe. This incident occurs as there happens to be no seats left in the cafe for the girl to sit and she has to sit next to him. They slowly break down their social awkwardness and start talking to each other and realize that they have many things in common, especially their love of a certain novel. They realize that they get along very well and decide to meet again. After a certain amount of time has passed, they meet up in a restaurant setting where the girl informs the guy that she does not feel too well, he puts this off as stress from work. However, after a short time she experiences a sharp pain and collapses, after which she is rushed off to an Emergency room. Here the doctors realize that she cannot be saved and inform the guy of this. Heartbroken and ridden with guilt he blames himself for losing what could have been the love of his life. This passes and we see the guy sitting in the same cafe as another woman approaches him, he begins thinking about what's going to happen as she proceeds to ask him if his table is vacant. He looks up and closes the book hess reading, which is revealed to be titled “Before the coffee gets cold”.

1.3 Overall Concept and Theme

The overall concept is about love and loss and how we as humans process love and grief.

1.4 Experience

On part of the website and interactivity, the comic is laid out in the conventional webtoon style. However, the stacked up layers sometimes form a “stop-motion” like effect when scrolling down. We wanted the user to fully be immersed in the story and therefore, the main form of interaction is the scrolling. While the user scrolls up and down the story panels they can experience the sounds and effects as they occur in the storyline.

2. Timeline and Roadmap

We began the project by firstly brainstorming the ideas for a webtoon. We came to an agreement that our work should explore a romantic theme and have human characters. Simultaneously we came up with a timeline of how things should be done. Figure 1 shows the final roadmap we adhered to when making the project.

Figure 1: Project roadmap

3. Implementation

3.1 Phase 1: Story development

We developed the story first before proceeding to draw the images/panels

3.1.1 Brainstorming

After the group established the genre, we decided to come up with the general theme and direction of the story. The group decided on making a story about a meet-cute (a term referring to an incidental meeting of two people in a romantic setting). We decided that the story should be about the meeting of two very introverted individuals. During the brainstorming session we talked about very deep emotional experiences that we had experienced. We realized that heartbreak is an emotion that we were all very familiar with and therefore we could draw inspiration from.

3.1.2 Script

After we brainstormed the ideas for the general path that the story will take we came up with the the script as seen below:

Type of Story

Tragic Love Story

Characters

Human

The Setting

Cafe, hospital, room

Layout/props

-

Storyboard Idea:

"Before The Coffee Gets Cold"

Frame 1: We see a young man sitting alone in a coffee shop, reading a book - "Before the coffee gets cold" (The book's not visible here)

Frame 2: A young woman walks in and looks around nervously, trying to find a seat. But the only seat available is next to this guy.

~~Frame 3: She is very uncomfortable approaching him but in the end, decides to give it a shot.~~

~~Frame 4: The woman asks the man if she can sit with him, and he nods.~~

~~Frame 4: After some minutes of awkwardness, she realizes that he is reading her favorite book. This becomes the starting point of their conversation.~~

Frame 3: She realizes that he is reading her favorite book. This becomes the starting point of their conversation. Soon they find out more commonalities between themselves. They exchange phone numbers and make plans to meet up again.

Frame 6: Out of nowhere, she suddenly collapses and is rushed to the hospital.

~~Frame 4: In one of these dates, she suddenly collapses and is rushed to the hospital.~~

Frame 5: In a tragic twist, we see the woman lying in a hospital bed. The man is standing beside her, looking heartbroken. They realize that she has a few days to live. The woman dies later.

Frame 6: We see the man walking away from the hospital, alone and desolate.
Frame 7: Months have passed from the day. The guy becomes more accepting of the tragic incident in his life.

Frame8: This final panel shows him sitting in the same coffee shop, reading the same book. Another girl approaches him in the same exact way. The guy is left spellbound.

Frame 9: The story ends with the book's close up on the frame which reveals: "Before the coffee gets cold"

3.1.3 Dialogues

We then went ahead to design the conversations that we thought the two individuals would have and how they would interact with the other characters. However, one challenge that we did not anticipate was that while we were doing the design process we would have to change the text boxes and dialogues in a manner not pre-planned. Below is a sample of the first set of dialogues we came up with:

Script and Dialogues

Scene 1 (1-14)[in image number 10]: The Girl (Normal Speech bubble with italics text, to insinuate nervousness) - "Hey, is this seat available?"

[in image number 7]: thought bubble for the guy: "I wonder why she's heading this way'

[in image number 11]: the guy: "Ummm yeah I guess so" (again italics to show nervousness)

In image number 13 : the guy: so i'm super into this book right now…

Scene 2 (15-26): While they are walking: "HAHAHA" (on top as a panel to indicate both are laughing)

In image 18: Guy: I didn't know you were into romance novels as well

Girl: Well I've always kept to myself and my books…

While sitting on bench
Guy: Do you want to hang out at my place for a bit?
Girls: "yeahh" in italics small font to show nervousness

Scene 3 (27-34): Guy: "So how was your day?"
Girl: It was umm…

RINGING sound with heavy pitch has to be integrated here somehow. On top include : AHHHHHHHH!!!!
In image 34: include THUDD!!! As a panel above characters.

Scene 4 (35-41) Ambulance sounds. WEE-OW WEE-OW in a panel on top

Scene 5 (42-49): Thought bubble for guy: Oh No, I could have made sure this didn't happen. I should have checked up on her…

Also include sobbing sounds.

Doctor: Hi, I understand you're in distress.
But you have to understand.
She will not be able to make it…

Scene 6 (50-60): Thought bubble for guy: "I finally thought I found someone…"
Next slide: "But she had to leave me too. What did I do wrong?"
Scene 3 (61-71): New Girl: "Umm hey are you free to talk?" Guy: "Yeah yeah I was just reading my favorite book but it's fine.."
[Flipping sounds]

3.1.4 Story-Boarding

After we had a general idea of the conversations our characters were going to have, we decided to create a wireframe of the comic strip and while researching the tools to take the storyboarding process step by step, we discovered the tool Storyboard That [2]. Our group used the tool to create 9 panels to uniquely express an outline of the story. This part is what consumed a majority of the time during the development of the project. As it was very time consuming but fit the project from a design perspective and allowed us a good degree of freedom with what to do with the characters.

We see a young man sitting alone in a coffee shop, reading a book - "Before the coffee gets cold"

A young woman walks in and looks around nervously, trying to find a seat. But the only seat available is next to this guy.

She realizes that he is reading her favorite book. This becomes the starting point of their conversation. Soon they find out more commonalities between themselves. They exchange phone numbers and make plans to meet up again

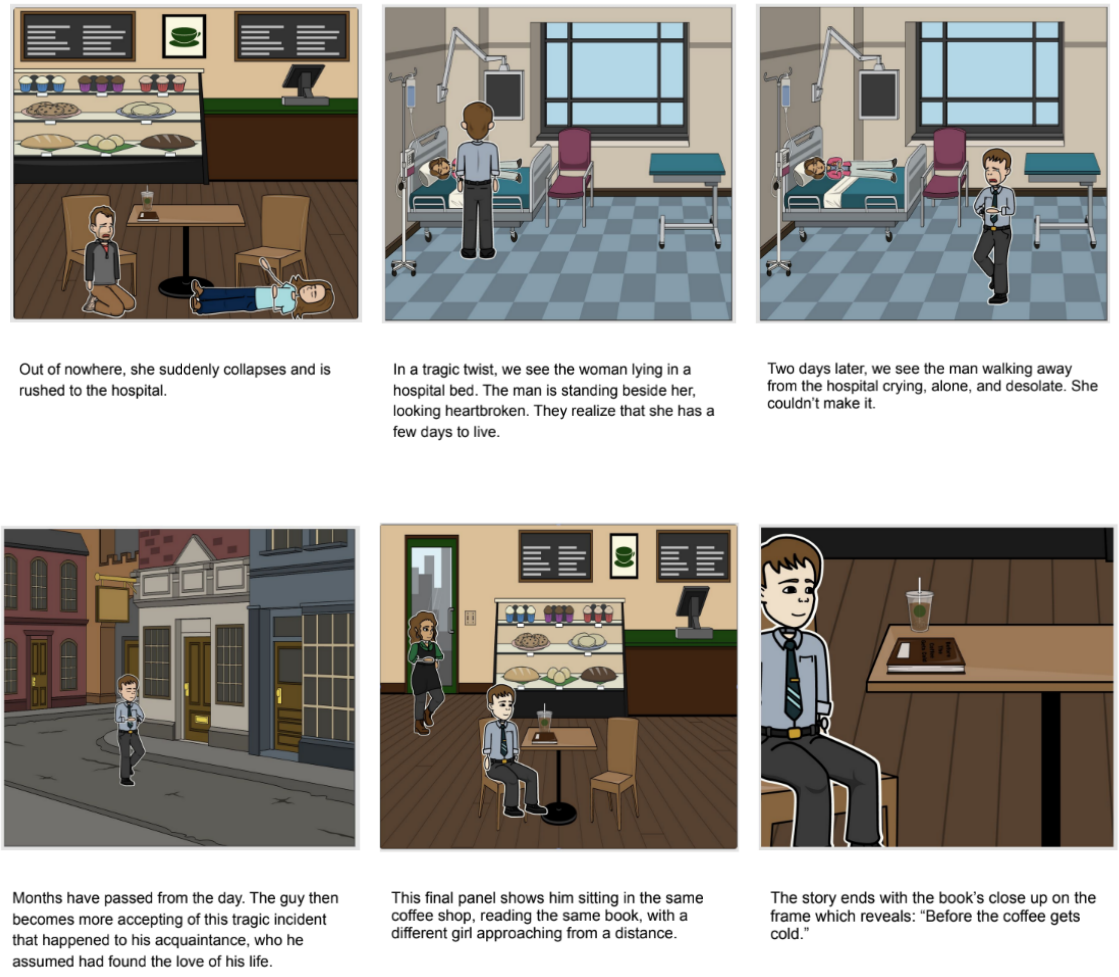

Figure 2: StoryBoard

initial letters capitalized and 6-points of white space above the subsubsection head.

3.2 Phase 2: Design and Art

The design and art phase of our project encompasses two main facets: visual art and sound.

We started by making pencil sketches of our ideas, then moved on to create the designs digitally on storyboardthat.com. It is an online platform that allows users to create and customize storyboards, which are visual representations of a story or narrative. We chose from a variety of characters, backgrounds. We then

moved to illustrator to add text, speech bubbles, and other graphic elements to the design. There were four distinct steps to the design process:

- Step 1: Storyboard to character design

  Our group initially wanted to use the storyboard backgrounds and custom characters to allow us more freedom in character design. Figure 3 shows what the first character depictions were like.

Figure 3: Initial character design

However, due to the simplicity of the storyboard application our group decided to stick with the storyboard character design. We spent a considerable amount of time redrawing some scenes and cropping scenes before we were able to finish almost 60 slides of unique png images.

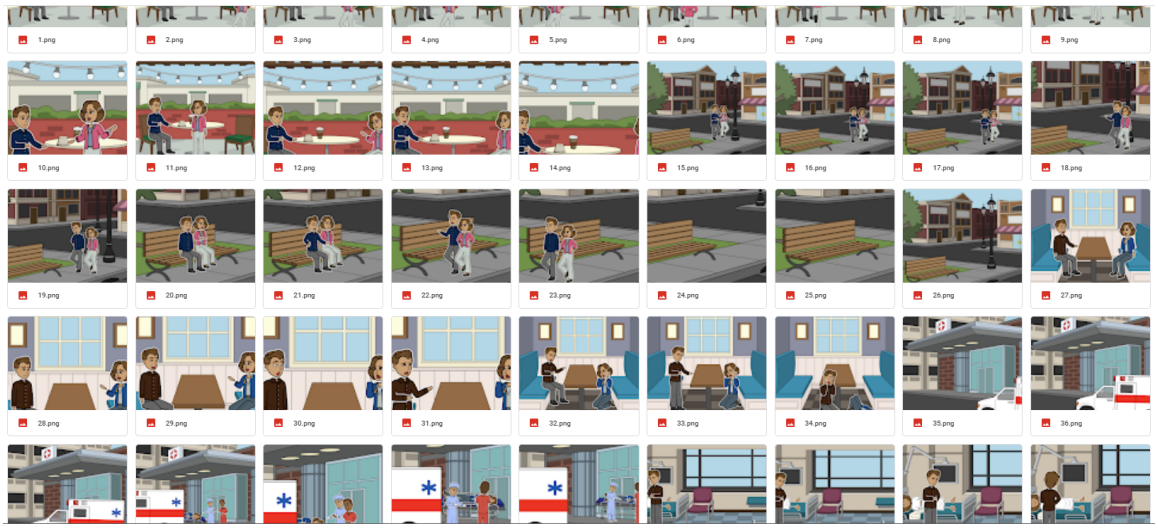

Figure 4: Flexbox implementation

- Step 2: Illustrator

We used illustrator to compile these individual png images into a single stack. Doing this on illustrator was done by declaring each frame as a layer.

- Step 3: Speech bubbles and fonts

  We then added in layers of text composed of text boxes where the dialogues were amended based on the amount of space we could dedicate for the speech bubbles. We used the comic sans font and tried to maintain a size of 14 throughout the project. We wanted the speech bubbles to have a comic feel to them so we used a website called freepik to obtain the designs (link) [3]

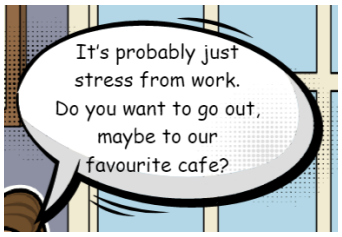

Figure 5: speech bubble style

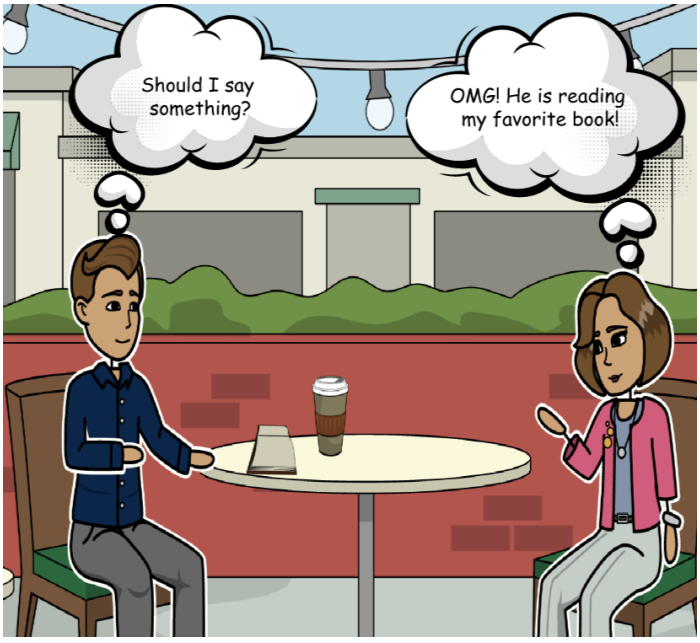

Figure 6: final visual product

- Step 4: Sound design and implementation

  The final part of the project was implementing the sounds that were associated with the individual scenes. He created a lot of those by mixing stock sounds from pixabay and Mixit to produce a library of probable sounds to choose from as seen in figure 7.

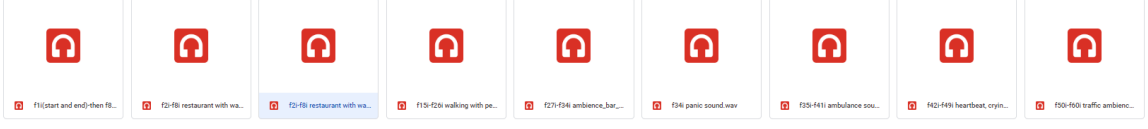

Figure 6: final visual product

After the selection of sounds that the group deemed appropriate we proceeded to the website implementation where our sounds could be actually integrated to the final product.

3.3 Phase 3: Website Implementation

The first step in implementing the website was learning how to make the host website responsive as we wanted our comic to be accessible on different devices. The team explored css flexbox [1] as a tool that can shape the DOM of our prospective website. Implementation of a test website (link) was then done as shown in the below figure.

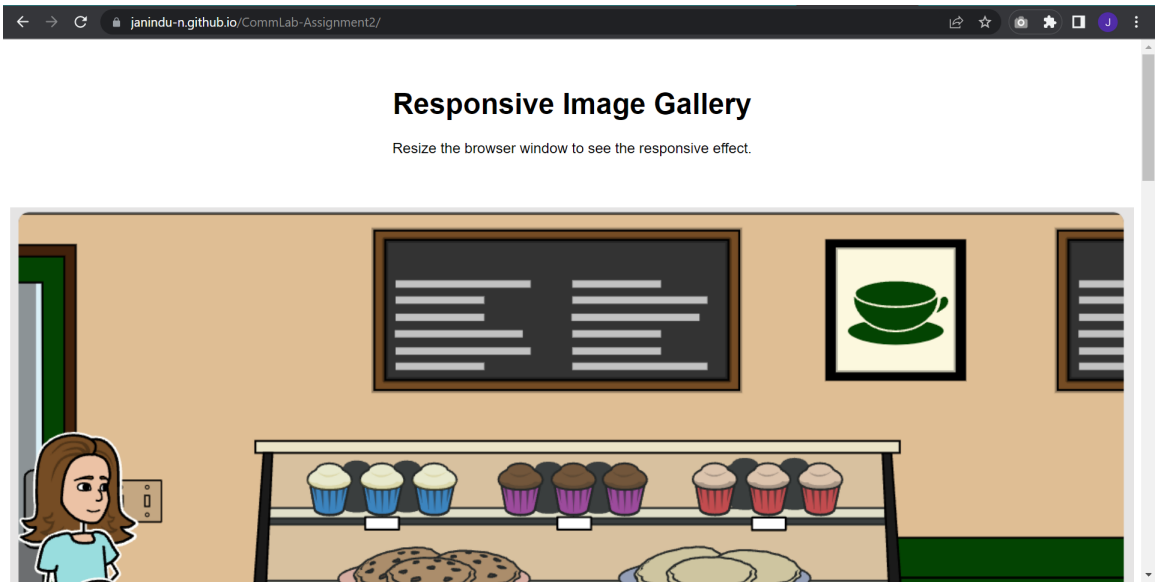

Figure 8: Flexbox implementation

This first implementation was done when the group was still in the story-boarding phase and the way that the graphics looked on the comic strip really satisfied our artistic requirements. In the subsequent implementations, the team realized that pictures and SVGs are great at resizing themselves. Since the website would have one central element at any point of time that could easily resize according to the dimensions of the screen - upon using view heights rather than pixels - the additional implementation of responsiveness was no longer necessary.

The idea behind the website was to have one single SVG with all the frames present as layers in it. As the user scrolled, the frames would change in sequence and the appropriate sound effects would start at particular frames. Scrolling was the central user interaction that everything was centered around. This allowed the user to get a very immersive experience of the comic.

The single SVG that was created using Adobe Illustrator with each layer having an unique id based on a particular naming convention was first included in the HTML file directly. There was an immediate problem with this approach. None of the elements inside the SVG code were visible in the DOM when the SVG was included as a direct image. With some research, the solution proposed was to include the file as an object. This approach showed all the elements of stack inside the HTML document along with the Ids we had created.

```
<!-- The SVG stack present as an object-->

    <object data="./assets/FinalDraft3.svg" type="image/svg+xml"
class="comic" id="comic_object">

        Your browser does not support SVG

    </object>
```

A new problem was found soon after. The stack object was present inside another document inside the primary HTML document. None of the elements inside the stack were directly accessible. Therefore, no CSS could be directly applied. With some more research, the only solution was to select the document inside the HTML document first and then adding or changing any properties of any of the layers in the stack using JS.

```
// Selecting the SVG object

    svgObject = document.getElementById('comic_object');

    // Accessing the document inside the object - very important to
access any element and manipulate within the SVG

    svgDoc = svgObject.contentDocument;

    // Finally accessing the SVG element and all its children

    innerDoc = svgDoc.documentElement;


    // Selecting all the frames in the SVG - including additional
sublayers present in the layers I created
```

```
    allFrames = innerDoc.querySelectorAll('g');
```

Also, the stack was a very heavy object and took quite some time to render on page load. Therefore, everything that was implemented or manipulated using JS was only done after the page load event.

```
window.addEventListener("load", () => {

//all the implementation code

}
```

The initial array of layers also consisted of the sublayers of various SVG layers. This problem arises from the fact that some vector elements like the comic bubbles had additional layers within them. This posed a problem as the framecount went way beyond 70. The objective wasn't to weed out the extra layers but rather to avoid redundant presence of sublayers as the very selection of the top level frame layers would lead to the selection of the sublayers. To fix this, frames were first filtered using the naming convention. Then these frames were then finally put into an array of arrays where the frames were grouped according to their frame number. Each array inside the filteredFramesArray could have up to 3 frames - the image, the speech bubble, and the text.

```
// Creates the array of array of filtered frames for every scene

function createFilteredFramesArr() {

    let i = 0;

    let currFrameArr = [];

    let flag = true;

    while (flag) {

        let currFrame = filteredFrames[i];

        currFrameArr.push(currFrame);

        let frameNo;

        frameNo = getFrameNo(currFrame);

        i++;

        // checks for subsequent frames with the same frame number

        while ((i < filteredFrames.length - 1) && frameNo ==
getFrameNo(filteredFrames[i])) {
```

```
            currFrameArr.push(filteredFrames[i]);
            // checking for end condition
            if (i > filteredFrames.length - 1) {
                flag = false;
                break;
            }
            frameNo = getFrameNo(filteredFrames[i]);
            i++;
        }
        // adds the frame array to the filtered frames array
        filteredFramesArr[fFAIndex] = currFrameArr;
        currFrameArr = [];
        fFAIndex++;
        // checking for end condition
        if (i > filteredFrames.length - 1) {
            flag = false;
            break;
        }

    }
}
// gets the frame number from the frame id
function getFrameNo(frame) {
    // for one digit frame ids
    if (frame.id.length == 3) {
        return parseInt(frame.id[1]);
    }
    // for two digit frame ids
    else if (frame.id.length == 4) {
```

```
        return parseInt(frame.id[1] + frame.id[2]);

    }

}
```

The reason the SVG stack was created in this way was to reveal the speech bubble and the text on further scrolling after reaching each frame. But this posed significant problems during the implementation especially on backward scroll and after careful user testing, the group decided that it was a better implementation for every frame to be displayed together in its totality as the scroll interaction was becoming too much in the other implementation.

After the frame array was successfully implemented, the next step was to figure out the scroll animation. There were several methods that were tried out during its implementation, with each new method coming with some improvements over the previous. The first one involved creating a huge document and then checking for a difference of a specific increment (set amount). When the change in increment was greater than this in either way (forward or backward), the frame was consequently changed. The problem with this implementation was that it didn't work all the time. There were situations when the page loaded from a random scroll position in the document and everything got messed up after that. This also wouldn't work if the user decides to drag the scrollbar around as it would render the increment useless. Therefore, a more robust implementation was necessary. The best approach to this was to map every frame to a specific position in the scrollbar that would cause the frame to appear at a particular position in the website no matter what. It made the program extremely efficient as the redundant checks were all removed and the only thing that was computed was the index in the frames array that the particular scroll position corresponded to.

```
// Event listener for the scroll event

    window.addEventListener("scroll", () => {


        // Getting the current scroll percentage

        let scrollP = getScrollbarPercentage();


        //  Calculating the current frame index

        frameIndex = Math.floor(scrollP / incr);

        console.log(scrollP, frameIndex);
```

```
        // Clearing the screen of the previously displayed frame - if any
        clearScreen();
        // Displaying first frame array - image, speech bubble, and text
        displayCurrFrame();

    });
```

```
// gets the scrollbar percentage
function getScrollbarPercentage() {
    //  getting the scrolltop, scrollheight, and clientheight of the document
    const scrollTop = document.documentElement.scrollTop;
    const scrollHeight = document.documentElement.scrollHeight;
    const clientHeight = document.documentElement.clientHeight;

    // calculating the scroll percentage
    const scrollbarPercentage = (scrollTop / (scrollHeight - clientHeight)) * 100;
    return scrollbarPercentage;
}
```

The final implementation also made sure to clear the screen of all the previous frames while displaying the next frame. This helped reduce the messiness caused by the illustrator frames stacked over one another.

```
// removes all the frames from the screen including the cover page
function clearScreen() {
    for (let i = 0; i < filteredFrames.length; i++) {
```

```
        filteredFrames[i].style.display = 'none';
    }
    coverP.style.display = "none";
}
```

The next step in the implementation was to add sound effects. This was done again by mapping particular sound effects to particular frames. A potential problem that we realized early on was the overlapping of different sound effects. To mitigate this we made sure to stop the last playing sound effect before a new sound effect was played. This stage of mapping sound effects required extensive work and several rounds of trial and testing. These sound effects were first imported into the JS file and then controlled using variable objects. The sounds were started along just when a particular frame was displayed.

```
// displays the current frame and also plays the sound effect/music
associated with the frame
function displayCurrFrame() {
    // getting the current frame array
    visibleFrameArr = filteredFramesArr[frameIndex];

    // checking for the sound effect/music to be played and calling
the function with the frame number to play the sound effect/music
    if (frameIndex - 1 == 1) playMusic(1);
    if (frameIndex - 1 == 2) playMusic(2);
    if (frameIndex - 1 == 6) playMusic(6);
    if (frameIndex - 1 == 13) playMusic(13);
    if (frameIndex - 1 == 25) playMusic(25);
    if (frameIndex - 1 == 30) playMusic(30);
    if (frameIndex - 1 == 33) playMusic(33);
    if (frameIndex - 1 == 40) playMusic(40);
    if (frameIndex - 1 == 45) playMusic(45);
    if (frameIndex - 1 == 49) playMusic(49);
    if (frameIndex - 1 == 59) playMusic(59);
```

```
    if (frameIndex - 1 == 62) playMusic(62);
    if (frameIndex - 1 == 69) playMusic(69);


    // displaying the current frame array
    for (let i = 0; i < visibleFrameArr.length; i++) {


        visibleFrameArr[i].style.display = "block";


    }
}
```

To play a particular music, a switch case is used to map the sound effect to its frame.

```
switch (fn) {
        // In each of these cases, the current playing music is
first paused, changed to the new sound effect, and then played again
        // This avoids overlapping of sound effects from different
frames
        case 1: {
            curr_music.pause();
            curr_music = aud_frame1;
            curr_music.play();
            break;
        }
```

The very last additional feature that the team decided on implementing was to include a screen that says “Keep scrolling” everytime the user is idle for too long. This was made possible by implementing a way mentioned in this post on StackOverFlow[4]. The frame was removed and the music was paused until the user started scrolling again. This was done in an attempt to mitigate the problem where the user skipped the cover page screen - which also mentioned that the user needed to scroll to view the comic - and was clueless in the middle of the

comic as to how to proceed. This feature can be considered a really important one to make the user experience smoother.

```
// function to detect idle time
function idleLogout() {
    //
    let t;
    // flag essential to avoid calling functions on undefined
variables in the beginning
    let flag = false;
    // calling the function of resetTimer on specific DOM elements
    window.onload = resetTimer;
    window.addEventListener('scroll', resetTimer, true);

    // calling the function to handle in the event that there screen
is idle for more than 20 seconds
    function triggerIdleHandler() {
        console.log("You have been inactive for 20 seconds");
        // music pauses
        curr_music.pause();
        // the screen is cleared and the keep scrolling message is
displayed
        clearScreen();
        keepScrollP.style.display = "block";

    }

    // function to reset the timer on specific events
    function resetTimer() {
        console.log("reset timer");
        // the setTimeout is cleared
```

```
        clearTimeout(t);

        // a new timer is set

        t = setTimeout(triggerIdleHandler, 20000);  // time is in
milliseconds


        // handling edge case on window onload

        if (flag) {

            // removing the keep scrolling message

            keepScrollP.style.display = "none";

            // displaying the current frame and restarting the frame
music/sound effect

            displayCurrFrame();

            curr_music.play();

        }

        flag = true;


    }

}

// calling the function to detect idle time

idleLogout();
```

We finally hosted the webpage by uploading the javascript, html and css files to github. Then we used the github pages functionality to render the following website.

4. Conclusion

Overall, the project was an engaging activity for all the group members and helped us gain essential skills.

4.1 Lessons Learned

Our group learnt very important lessons in the scope of design, implementing projects on the web and teamwork. We learnt how to use tools that may restrict the amount of flexibility when it comes to thinking creatively. For example, the use of storyboardThat meant that our vision of what the characters were was

pretty restricted from the beginning. Our group was able to overcome this to deliver our message with minimals misinterpretations.

4.2 Future Improvements

In terms of future improvements it may be feasible to implement the following to make the project much more appealing and interactive:

- Making panel size uniform
- Implementing interactivity for objects in the comic such as the speech bubbles or even things like the ambulance.

5. REFERENCES

[1] W3Schools Flexbox responsive (2023). *CSS Flexbox Responsive* [Code]. W3Schools: https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_image_gallery

[2] Storyboard That (n.d.). Storyboard Creator [Software]. Retrieved March 9, 2023, from https://www.storyboardthat.com/

[3] Freepix (2023). *Speech Bubbles* [Code]. Freepix comic chat bubbles: https://www.freepik.com/free-vector/comic-chat-bubbles-collection_16395946.htm#query=speech%20bubble&position=25&from_view=keyword&track=ais

[4] Stackoverflow (2023). [Code]. Stackoverflow javascript: https://stackoverflow.com/questions/667555/how-to-detect-idle-time-in-javascript