CART 451 Final Project Prototype

Regale Me With Tales of Joy by Sharon Ku

Table of Contents

PART A: Progress Report

Summary of Project Proposal

Regale Me With Tales of Joy (or *Regale Me* for short) is an online website that allows people to collaborate and build stories together, with the starting prompt being people's happiest memories. My goal is to design a space of beauty and joy for other people. I think a lot about my little cousins when I decide on topics for my creative projects because I want to create a better world for them, where they are free to learn, explore, and have fun. With this theme, I want to remind them to always seek happiness and to be surrounded by rich and enthralling stories. On a large scale, I want to empower people through collective creativity by providing them with an easily accessible space where they can share their identities and personalities in writing.

I envision my project as an online bookshelf filled with old, musty storybooks—think the small bookshop at the start of *Beauty and the Beast*. Visitors to the library click on books to open them. Then they read through the pages and can anonymously continue the story by adding a page of text to the end of the book. Visitors can only add one page per session to allow other people to contribute to the story. The cover page that reveals the topic of the book is based on someone's happiest memories. For these prompts, I will begin with my own happiest memories, and if there is time remaining in the project, I will allow visitors to submit their happiest memories as starting prompts.

My audience is storytellers of all ages. If I target this project's audience further, the audience is my two baby cousins, my family members, and my friends. The website's aesthetics will be heavily based on illustrations that convey themes of autumn coziness, beauty, joy, and peace. I want visitors to reminisce about the story books they read as children.

Description of Current Stage

For my prototype, I have set up all the technology needed for client-server-database communication: Node.js, Sockets.io, MongoDB, PixiJS, jQuery, HTML, CSS, and JavaScript. I created a schema to store the book information and pages.

On the client side, I set up a single book whose pages can be flipped using Back and Next buttons. When the page is refreshed, the book's content (number of pages, prompt and page text) is dynamically updated from the MongoDB via sockets. When you turn the page, there is also an page-flipping animation that is triggered.

On the last page of the book, there is an input field and submit button that is supposed to allow visitors to continue the story. This feature does not work yet since I am having trouble saving the input string to MongoDB. After the visitor fills out the input field, a prompt was also supposed to appear to ask them to guide the next writer. This also has not been implemented due to the issue of array-saving. I included a more in-depth description of the other features left to do, such as adding a bookshelf to select books and a pond to add the happiest memories.

Detailed Images/Diagrams of the Overall System

Breaking down the code for seeing books and updating stories (Stage 2):

1. Check the database for number of existing book entries
    a. Display that number of books as book images on the bookshelf
2. If a user clicks on a book, grab the book title and page contents from the database, then open book view so that the user can flip through pages.
3. Once the user reaches the last page, show an input box to continue the story.
4. If the user types in that box and saves, update the database with the new text string.
    a. Then update books for all visitors: all visitors should see this new page when they view the book.

Breaking down the code for creating a pond of happiest memories that can be added to (Stage 4):

1. Check the database for number of existing memories
    a. Display that number of memories as leaf images on the pond
2. If a user clicks on a leaf image, randomly grab a memory entry from the database, then display the memory string, name of writer, and date of entry.
3. If a user adds a new memory, provide input box and once saved, store the memory string in the database and add a new leaf in the pond.
4. Once it is time to create a new book, randomly select a memory from the database that hasn't already been used as a book topic.
    a. In the database, create a new book entry with this memory.
    b. Visitors can now add to this book.

Description of the Book Schema in MongoDB:

Schema setup	Description
● Book ○ title: String, (title of book containing happiest memory) ○ currentlyEditing: Boolean, (true if a visitor is editing the book at the moment) ○ cover: Object ■ { ● color: String, ● type: String, ■ } ○ pages: Array ■ { ● num: String, ● prompt: String, ● pageText: String, ■ }	● Explanations: ○ title: title of book containing happiest memory ○ currentlyEditing: true if a visitor is editing the book at the moment ○ cover: object storing properties of the cover page ■ color: cover page color ■ type: type of cover page (will have selection of images to choose from) ○ pages: array that stores all the page content. For each page, there is: ■ num: page number, start counting from 0

■ { ● num: String, ● prompt: String, ● pageText: String, ■ }	■ prompt: string of text that guides the writer for this page ■ pageText: actually page text

Appropriate selection of API's and 3rd party libraries

I intend to create a website. The client side will show a 2D canvas made using PixiJS, HTML, and CSS. I chose to use the PixiJS library because it renders quickly (based on my own experiences, it renders quicker than p5.js) and it is free.

The backend will require Node.js, socket.io, and MongoDB. Node.js and socket.io will be used to allow the stories to be updated with different users and communicate between the server and database. MongoDB will store the happy memory prompts, and story titles with their corresponding page content.

Description of Features

Stage 1: Aim for prototype

My original goal for this prototype was to enable visitors to add pages to one book only.

Feature 1: Fill up book content with data from MongoDB
This feature works. When I update the book schema in Mongo and refresh the book scene page in the browser, the book gets updated with the correct number of pages and with the right content.

Feature 2: Pages flip with an animation
This feature works. When visitors click on the Next and Back buttons, the pages flip with updated content on them and correct z-indices. The animation is intended to simulate the book reading experience, making it feel more authentic and physical.

Feature 3: After the final page of the book, allow the visitor to add one page per session by filling out an input field, and this page gets saved in the database for future visitors to see
This feature is in development. Visitors can see a prompt on the last page, with an input box and Submit button, but when they click on “Submit,” the input string is not getting updated to MongoDB. The bug is caused by my inability to update an array inside the book schema.

Feature 4: After entering text for a new page, ask visitors to provide a prompt for the next page (to guide the story)
This feature has not been developed. I intend on implementing it as soon as I figure out the bug in Feature 3 since I also need to update the pages array to store the prompt information.

I've listed the features I intend to accomplish for the final (at the very least Stage 2) and if time allows (Stages 3-4). None of these features have been developed yet.

Stage 2: Aim for final

- **Description:** Bookshelf scene where visitors can choose books, book scene where visitors can continue the stories
- Add a bookshelf where there are 10 different books to choose from
- All book topics will be my own memories
- Allow visitors to continue the stories with text input that gets saved to the database
- Playtest

Stage 3: If time allows

- **Description:** Bookshelf scene, book scene, UI to input happiest memory prompts
- In the bookshelf scene, add a button that when clicked, reveals an input field where visitors can add their happiest memories
- These memories will be saved to the database, and randomly chosen as starting prompts for stories

Stage 4: If time allows

- **Description:** Bookshelf scene, book scene, and outdoors scene where visitors can drop their happiest memories
- Add an outdoors autumn scene with floating leaves on a pond
- A button allows visitors to add their happiest memories to this pond of ideas
- Once a memory is added, it takes the form of a leaf that floats down on the pond
- Visitors can click on a leaf to see individual memories

PART B: Interactive Prototype

Clickable Wireframe Prototype

Mid-Fidelity Prototype

Code found here: https://github.com/sharon-ku/regale-me

Experiments & Development

Tutorials Used

- Resize Pixi canvas: https://jsfiddle.net/bigtimebuddy/oaLwp0p9/
- Make VS Code prettier on save: https://www.alphr.com/use-prettier-vs-code/#:~:text=Manually%20Format%20Document%20on%20VS%20Code%20Using%20Prettier&text=For%20those%20on%20Windows%2C%20click,Formatter%E2%80%9D%20to%20format%20the%20code
- Socket setup:https://www.npmjs.com/package/socket.io

- Mongo Array: https://www.mongodb.com/docs/manual/reference/operator/update/push/#:~:text=If%20the%20value%20is%20an,for%20%24push%20%2C%20see%20Modifiers.
- https://www.codementor.io/@prasadsaya/working-with-arrays-in-mongodb-16s303gkd3

Progress Photos

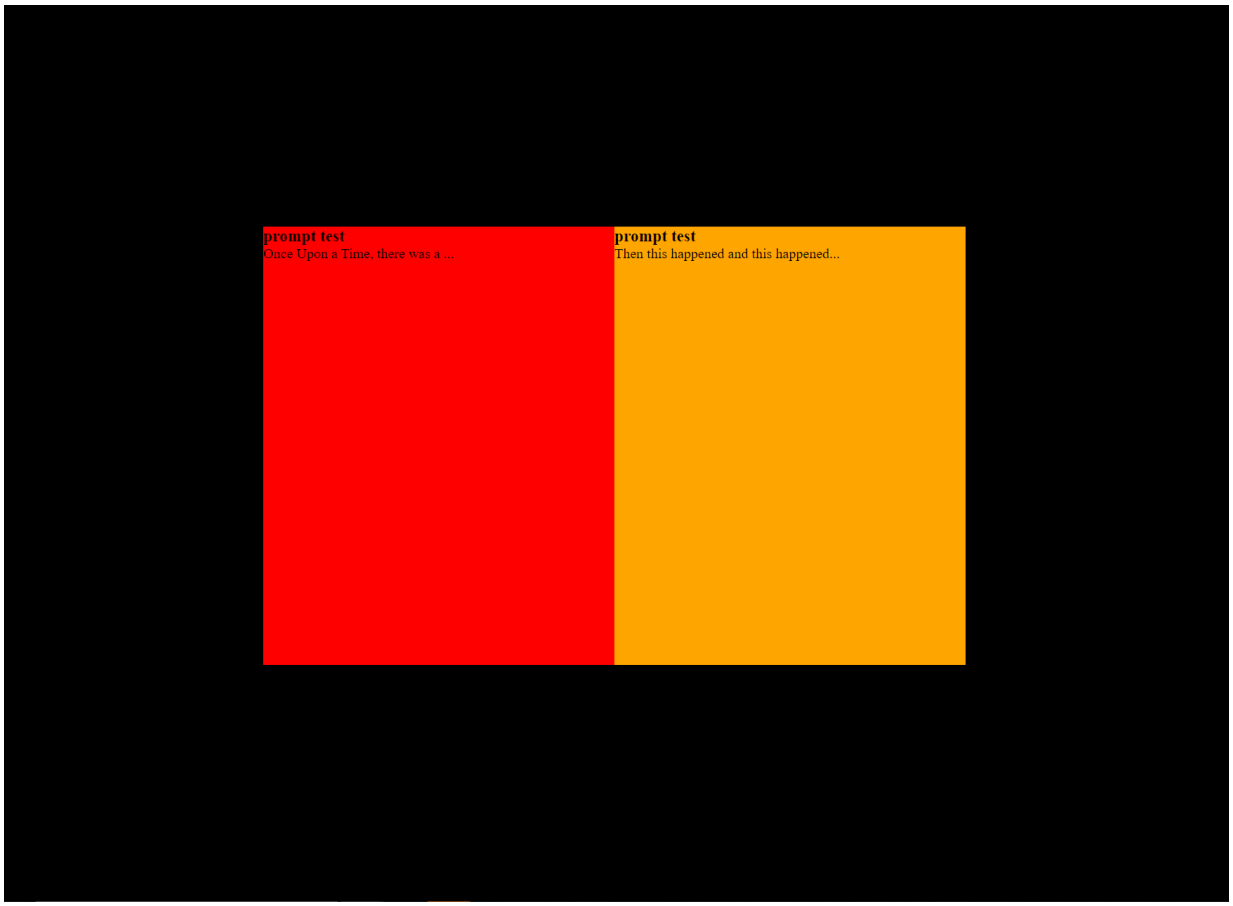

Figure 1. Created a book layout with placeholder content.

Figure 2. Added arrows for page flipping.

Figure 3. Found a tutorial online for animating the page flip. I decided to implement this earlier on to avoid doing major reworks of the page layout code later on. Now the pages have an animation.

Figure 4. Most challenging part of project: Transforming the tutorial's page code into JavaScript code that allows pages to be dynamically added. I had to do a lot of math to figure out how to integrate for-loops for the number of sheets or pages, on which page the information needs to be displayed, how the Back and Next buttons are affected with which page, and sorting out the z-index nightmare. I had to temporarily remove the arrow keys in Figure 2 since the tutorial used Back and Next buttons, though I might bring them back later.