

CS 320 Course Project Final Report

for

JournalJay

Prepared by

Team Sizzle Snap

|  |  |  |
| --- | --- | --- |
| Laurel Anderson | 11638131 | laurel.o.anderson@wsu.edu |
| Irina Bejan | 11613022 | Irina.bejan@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | 12/16/2020 |
|  |  |
|  |  |

Table of Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 5

3 Implementation 8

3.1 Development Environment 8

3.2 Task Distribution 8

3.3 Challenges 8

4 Testing 9

4.1 Testing Plan 9

4.2 Tests for Functional Requirements 9

4.3 Tests for Non-functional Requirements 10

4.4 Hardware and Software Requirements 10

5 Analysis 11

6 Conclusion 12

Appendix A - Group Log 13

# Introduction

The project is JournalJay, a web-based journaling site where users will be able to make and keep track of journal entries. The user can also customize their entries. This is a digitalization of a traditional medium.

## Project Overview

JournalJay is a web-based journaling site where users can make an account, sign into their account, make journal entries, review those journal entries, get help, or review their account information. This site will work on modern browsers such as Google Chrome and Mozilla Firefox. This convenient web journaling app will give more people the ability to air their thoughts in writing.

## Definitions, Acronyms and Abbreviations

Here are the definitions of terms used in this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Admin/  Administrator | Someone who is given specific permissions to manage and control the system. |
| App | Application. |
| Browser | A software application for accessing information on the Web. |
| CSS | (Cascading Style Sheets) A programming language used to style HTML. |
| HTML | (Hypertext Markup Language) The standard markup language for documents designed to be displayed in a web browser. |
| HTTPS | (Hypertext Transfer Protocol Secure) The data transfer protocol used on the World Wide Web. |
| IEEE | Institute of Electrical and Electronics Engineers. |
| JavaScript | High level programming language used with HTML and CSS on the Internet. |
| Web | A collection of websites stored in web servers. |

## References and Acknowledgments

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended

Practice for Software Requirements Specifications”, October 20, 1998.

# Design

## System Modeling

The JournalJay system no longer has admins as one of the types of users. Therefore, Figures 2 and 6 from the software design document are no longer valid. Figure 1 remains the same. Figure 4 and 5 were from the design document were changed to the following:

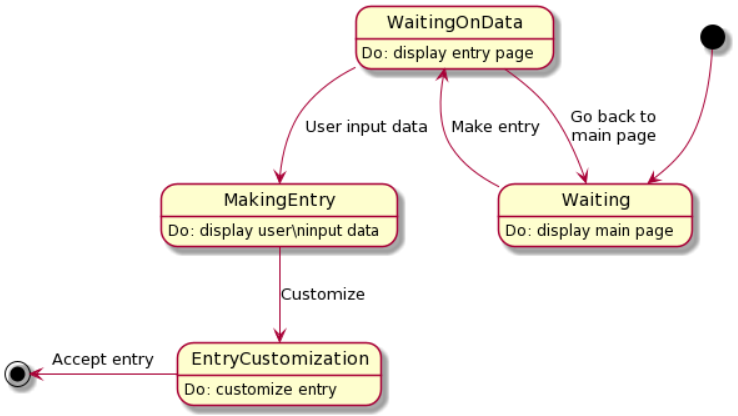


Figure 1 – Making a Journal Entry

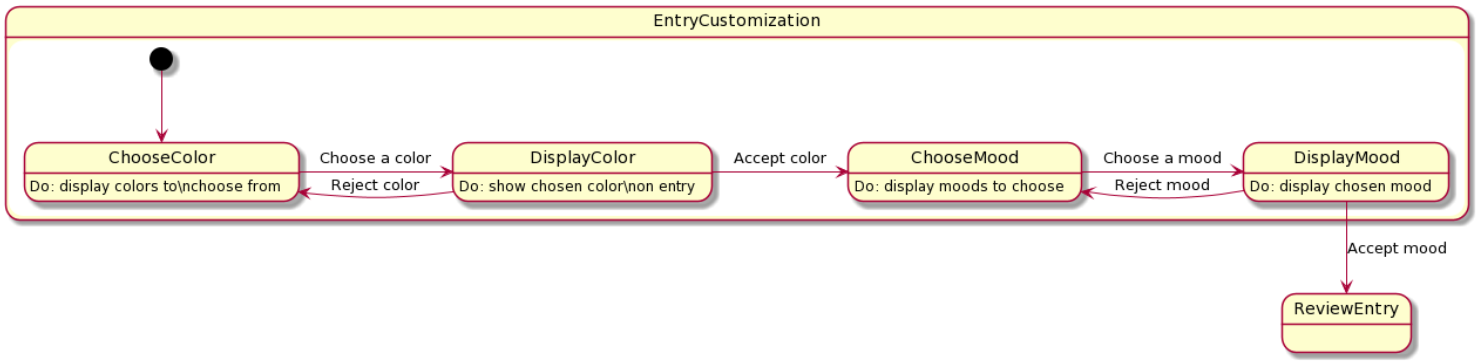


Figure 2 - Entry Customization

Table 1: Making a Journal Entry

|  |  |  |  |
| --- | --- | --- | --- |
| **Start** | **Transition** | **End** | **Description** |
| Waiting | Make entry | WaitingOnData | The system displays the main page until the user chooses to make an entry. |
| WaitingOnData | Go back to main page | Waiting | The user can choose to return to the main menu without creating a journal entry. |
| WaitingOnData | User inputs data | MakingEntry | Here, the user is actively inputting data into the journal entry. |
| MakingEntry | Customize | EntryCustomization | Once the user is done inputting data into the entry, they can then choose to customize the entry. The EntryCusomization table has the transitions for the class. |
| EntryCustomization | Accept entry | Accept State | User accepts the changes that they have made and save them. |

Table 2: Entry Customization

|  |  |  |  |
| --- | --- | --- | --- |
| **Start** | **Transition** | **End** | **Description** |
| ChooseColor | Choose a color | DisplayColor | The user can choose a color for the journal entry and the system then goes to display it. |
| DisplayColor | Accept color | ChooseMood | The user then can choose a mood and the system displays it on the journal entry. |
| DisplayColor | Reject color | ChooseColor | The user can choose to reject the color and go to choose a new color. |
| ChooseMood | Choose a mood | DisplayMood | The user can choose a mood and it will display on the journal entry. |
| DisplayMood | Reject mood | ChooseMood | The user can return to choose a different mood by rejecting the current mood. |
| DisplayMood | Accept mood | FinalState | The user can accept all customization options and go to the next state outside of customizing. |

Implementing JournalJay also resulted in some changes to the class diagram. The program organization is best summarized in the following updated diagram and table:



Figure 3 - JournalJay Classes

|  |  |
| --- | --- |
| **Class** | **Use** |
| UserProfile | Each user profile will be of this type; the class stores basic information including their username, password, an answer to a security question in “forgot,” true or false if they are an admin, and a list of their entries. |
| UserList | This is the grand list of all users in the system. We’ll need this to search at login to be able to report “wrong username or password,” as well as to retrieve a profile at correct login credentials. |
| Entry | Each user entry will be of this type; the class stores all the features of a journal entry. This class also contains all the functionality to work with the frontend in creating and displaying entries. |
| Login | This class contains all the functionality for logging a user in. |

## Interface Design

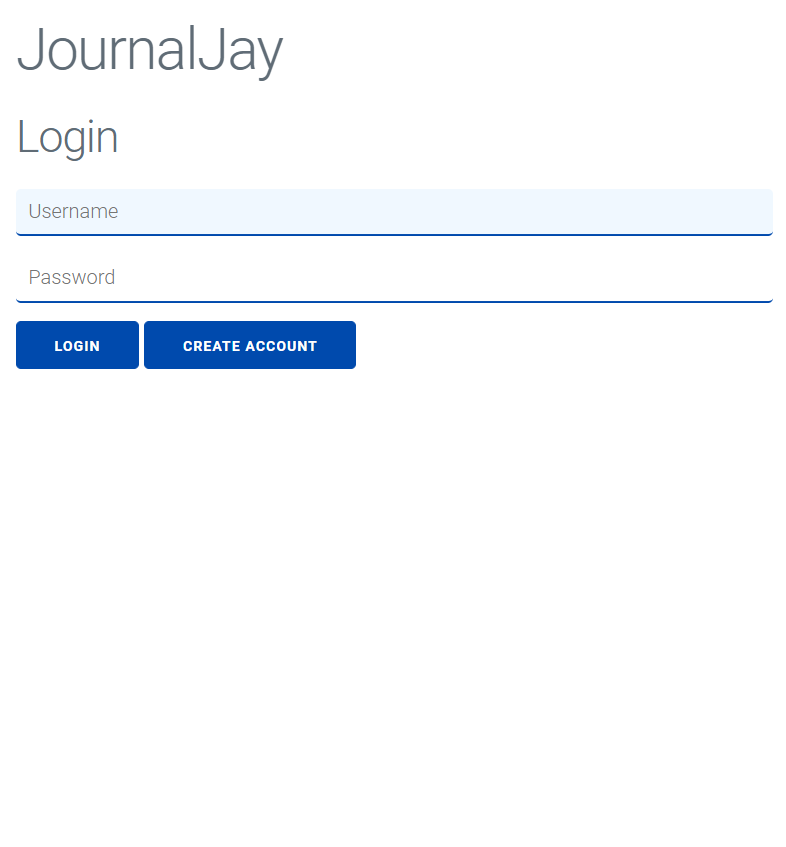
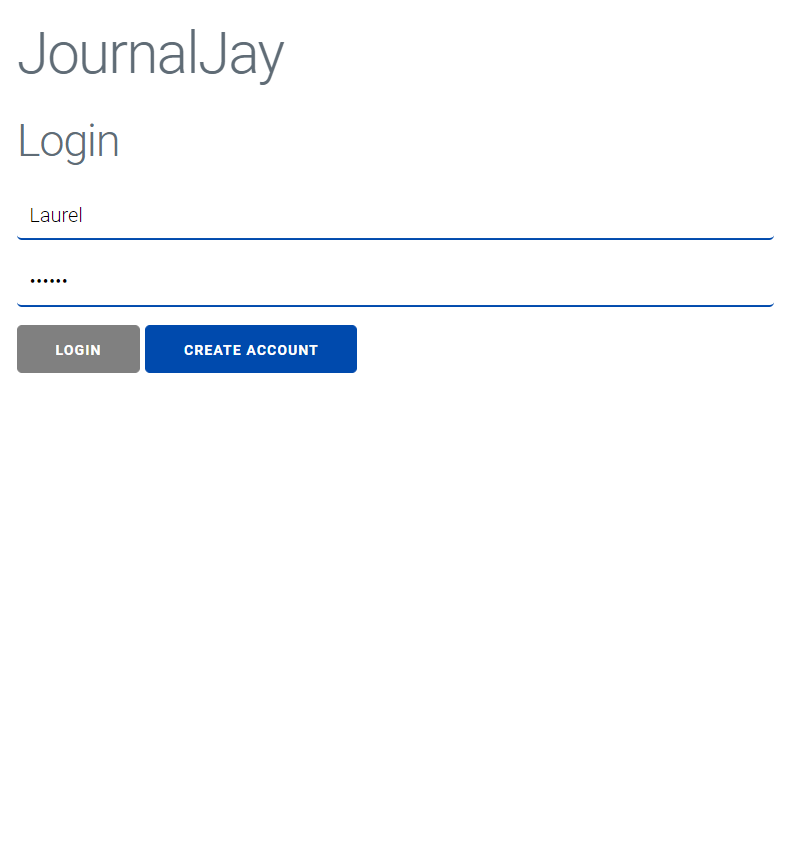
Here are the screenshots that show the JournalJay interface.

Figure 4 - Login



Figure 5 - Create a Journal Entry



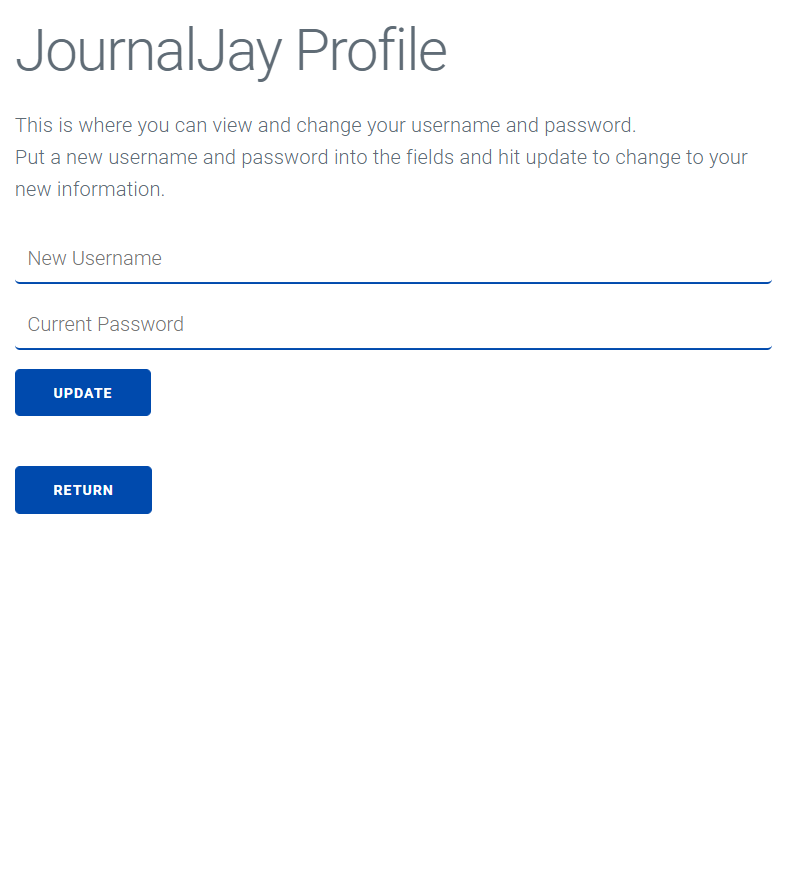
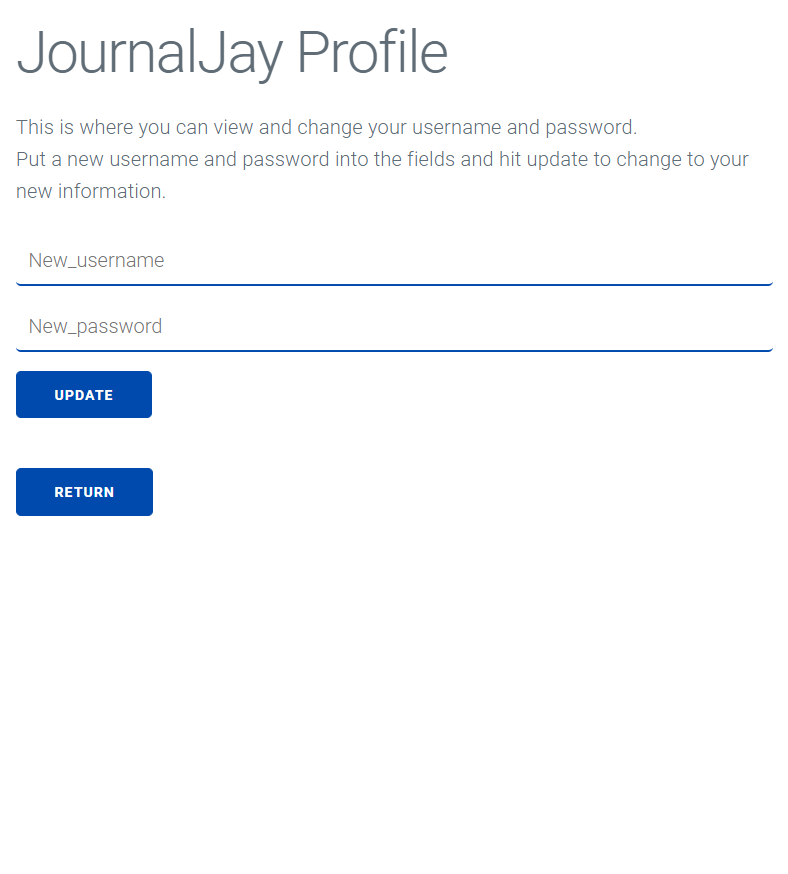


Figure 6 - Change Username and Password

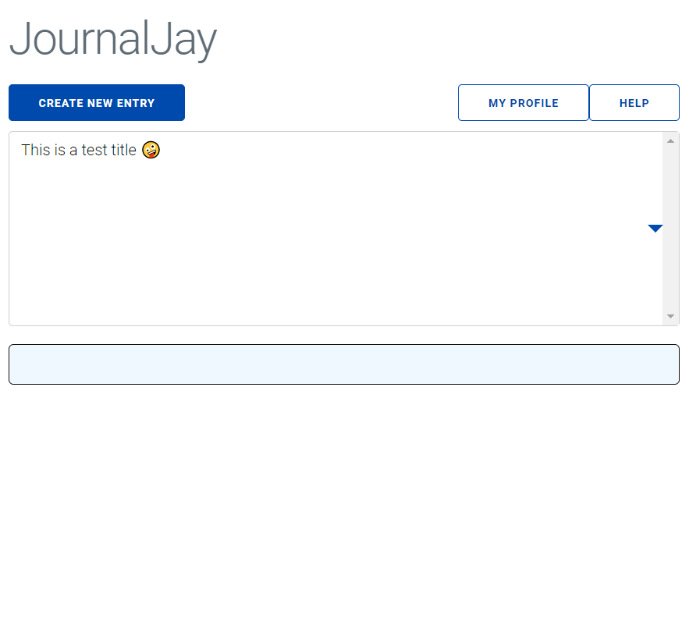


Figure 7 - Review Entries



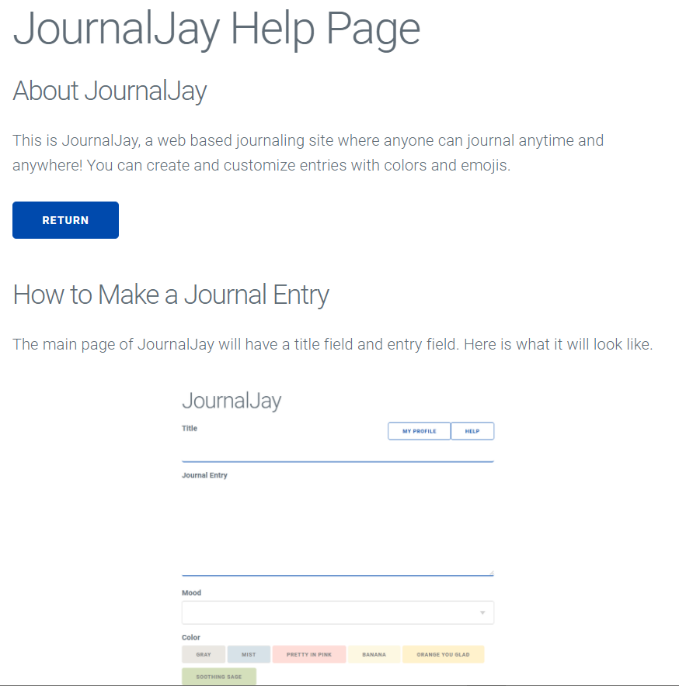


Figure 8 - Help Page

# Implementation

## Development Environment

The programming languages used for this project are:

* HTML/CSS
* JavaScript.

The IDEs used by the team included:

* IntelliJ IDEA
* Visual Studio Code

Additional tools used for this project are:

* Milligram
* GitHub
* JavaScript general libraries

## Task Distribution

Both team members worked with each other to get all project aspects done. If one member had trouble, or if they did not have time to finish an aspect of their code, the other member would step in to help. Both members fully tested their own code then pushed their changes to GitHub for the other member to test. All issues found in testing were brought to the team member in charge of that section. Here is the general distribution of work:

Laurel Anderson – Login page, user account management, testing code, help page, profile page.

Irina Bejan – Journal entry creation and customization, testing code, profile page, all styling*.*

## Challenges

No major challenges were noted. We worked well and troubleshooted together to make JournalJay bring our brain-ideas to life.

# Testing

The code for this project was tested incrementally, meaning that the core components of the code were created and tested before the next components were implemented. All code was first tested by the person in charge of that code, and once that testing was done the code was pushed to GitHub and the other team member tested it.

## Testing Plan

Laurel oversees the user sign in page. Here are the functions and classes to test:

* UserProfile class – create a user profile object that assigns name and password.
* UserList class – create an array of UserProfile objects.
* The methods that were associated with the UserProfile class: addUser(user) and search(username).

Additional tests for the login page include:

* createAccount(inputName, inputPassword) – search for a user and creates an account.
* userInput(inputName, inputPassword) – decide what to do with the user input.
* confirmBox(inputName, inputPassword) – ask the user if they want to create an account.
* traverse() – open the view.html file.

Irina oversees journal creation and perusing. Here are the functions and class to test:

* Entry class – construct an “entry” type with title, body, date, mood, and color.

Additional functions for tests include:

* init() – initialize the screen; update it for the user
* writeEntry() – tell the html to go to the “write” div and ensure all the buttons are pressable.
* readEntry() – tell the html to go to the “read” div.
* displayEntry() – show the user the entry they click on when in “peruse” mode.
* saveEntry() – helper function for the save button to store the entry in a local entry array.
* changeColor() – change the background color of the entry if the user requests it.

All these tests need to be done by December 12th, 2020. Additional testing will be conducted later by both team members.

## Tests for Functional Requirements

Use case 1— login in as a current user → add a journal entry → try to create another entry → check account and change account information: The JournalJay site did pass most of use case 1. It accepted the known user, it created and displayed the journal entry, but the site was not able to display the current user. This would be fixed by having a data base implemented. The site was le to display the changed username and password.

Use case 2— login as a new user → go to help page → return → make a journal entry → review journal entry: The JournalJay site did pass all of use case 2. The user is able to create an account, go to the help page to get info on the site, go back to the main page, and review the journal entry.

Use case 3— try to login with right username but wrong password: The site passed this use case. The site checked the username and password and told the user that there one of them is wrong.

Use case 4— try to create account with username that is already taken: The site passed this use case. The site alerted the user that the username was taken.

Use case 5— try to login with no entry into the fields: The site passed this use case. The site alerts the user to please input a username or password.

Known bugs: In some cases, when you clear the browsing cache before you create a journal entry, you cannot click on the profile or help buttons. This is fixed when you first create a journal entry before press either of those buttons. Sometimes when you create a journal entry and then try to make one directly after, you are alerted that you cannot make another journal, but then it allows you to make one anyways.

The previous bugs are known, and given more time, they would have been addressed to the best of our ability.

## Tests for Non-functional Requirements

Use case 1— try to load the login page: The site passes the requirement of loading within 5 seconds. This was tested with a simple count as opposed to a test suite, but a formalized test suite is the goal to improve the availability of numerical data.

Use case 2— try to press the button to cancel a journal entry: The site passes the requirement of only taking one click to bring about the action (in this case, resetting the journal creation fields).

Use case 3— try to press the “Help” button: The site passes the requirement of only taking one click to bring about the action (in this case, loading help.html).

Use case 4— try to copy and paste the password on the login screen once a user inputs a password (e.g., “•••••••”): The site passes the requirement of not allowing the user to copy this. Pasting results in pasting whatever was previously saved to the clipboard, if anything.

## Hardware and Software Requirements

To perform the tests, all the source code is required, as well as a relatively recent version of a common browser. For speed testing, mocha and chai testing libraries should be used. For hardware, the user must have a functioning keyboard, mouse, and monitor.

# Analysis

Laurel Anderson

* Milestone 1: ~13 hours.
* Milestone 2: ~6 hours.
* Milestone 3: ~20 hours.

Milestone 3 took the most effort because it included building and testing the site. It was also hard to translate the software design docs to code. We had to change some things in the program and put those changes in the updated report.

Irina Bejan

* Milestone 1: ~10 hours.
* Milestone 2: ~5 hours.
* Milestone 3: ~30 hours.

Milestone 3 required more effort, but it was also most rewarding because I was able to see the code come to life. There were a lot of stylistic things that seemed easy in theory, but when attempted, the frontend CSS or HTML didn’t respond to what I thought the backend JS was telling it to do.

# Conclusion

Throughout the experience of coding JournalJay, we got the opportunity to practice and expand on technical skills as well as soft skills. We learned how to coordinate on a team, especially a small team. We got more comfortable with each other’s work style and held short “scrum meetings,” finding that quick check-ins helped us stay on track. Another soft skill we developed was working with each of our schedules; we had to be flexible with meeting times and deadlines and encouraged each other to stay on track with our timeline.

A great deal of tech skills was developed as well. The use of Git was difficult for us at first, but we soon adjusted and used it for everything. We found that the best way to collaborate through Git was to first split up the work in a naturally divisible way; in our case, that meant one person worked on login and profile management, while the other did journal entry creation and customization. Additionally, we learned how to apply JavaScript to HTML/CSS. We created fields and buttons in our “view” and connected them to a controller in JavaScript, and then the controller would send the information to the JavaScript internal logic.

Overall, it was a project that required us to step out of our comfort zones—both by coding a product that we had never coded before (in this complexity), and by having to do it together. We believe the documentation for the project, as well as JournalJay itself, placed us into a real-world work-like situation, providing a chance to exercise what we’ve learned up until now, and heavily build on that knowledge in practice.

Appendix A - Group Log

We talked via text and held meetings through Zoom and facetime. Text was great for quick questions, Facetime was good for discussing ideas, and Zoom was good for showing our code to each other.

10-2-20 10:00am–10:20am Filled out the team agreement form.

10-9-20 9:00am-9:25am Drafted a schedule and rough timeline to complete the SRS document. Discussed concerns and initial ideas in our project requirements.

10-16-20 9:00am-9:45am Sectioned out the SRS document. Set initial deadline of first draft for 10-25-20).

10-30-20 9:45am-10:30am Talked about project specifications and worked on the SRS document.

11-04-20 4:00pm-5:30pm Discussed outstanding changes and set up GitHub repo for teamwork.

11-06-20 12:00pm-1:00pm Made final necessary adjustments.

12-01-20 5:00pm-6:00pm Talked about the project and what framework we were going to use.

12-02-20 4:00pm-4:45pm Talked about who would work on what. Laurel – Sign in. Irina – Journal entries.

12-07-20 2:00pm-2:45pm Checked in and looked at each other’s work. Gave advice and encouragement.

12-11-20 3:00pm-4:30pm Worked on the site together.

12-16-20 10:am-10:45am Discussed the final touches of the report doc.