

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 |
|  |  |
|  |  |
|  |  |

Content

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 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 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** |
| Web | A collection of websites stored in web servers. |
| App | Application. |
| Browser | A software application for accessing information on the Web. |

## References and Acknowledgments

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

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

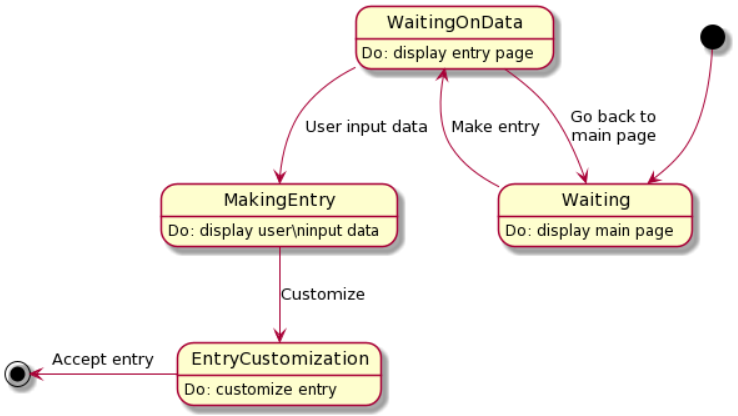
# Design

## System Modeling

< Update your UML diagrams in milestone 2, to reflect the real implementation of this software.

TO DO: Provide an updated version of the UML diagrams, including use case diagrams, sequence (or state) diagrams, activities diagrams, and class diagrams. If you don’t have an updated version, just mention: “our implementation strictly follows the design document (milestone 2)”. >

The JournalJay system no longer has admins as one of the types of users. Therefore, figure 2 and figure 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:



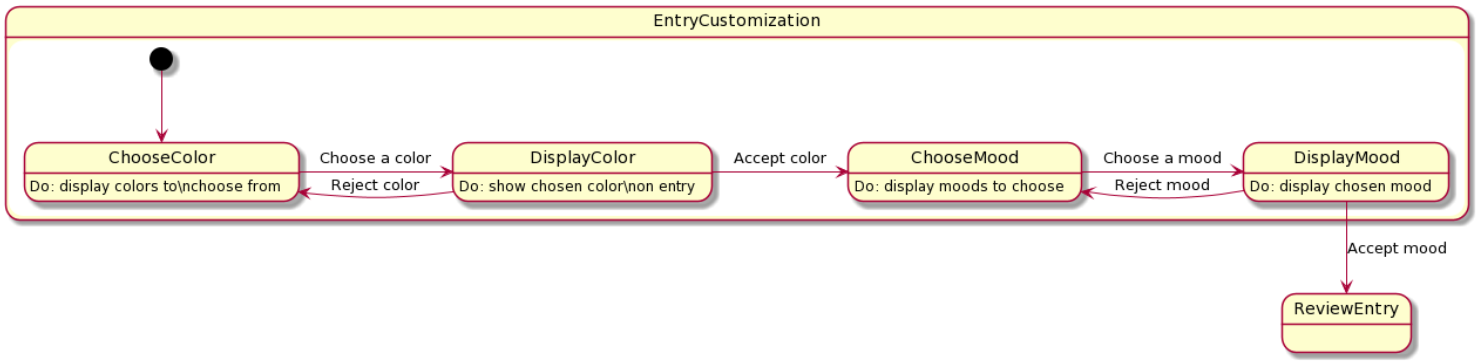


Figure #: Updated Entry Creation State Diagram

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. |

## Interface Design

<Provide several screenshots to illustrate your interface design.

TO DO:

For each subsystem, pick one or two representative screenshots and paste here.>

# Implementation

## Development Environment

The programming languages used for this project are:

* HTML/CSS
* JavaScript.

The IDEs used by the team included:

* Intellij
* Visual Studio Code

Additional tools used for this project are:

* Miligram
* GitHub
* JavaScript general libraries

## Task Distribution

Both team members worked with each other to get all projects done. If one member was having 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 and account page.

Irina Bejan – Journal entry creation and customization, testing code, account page*.*

## Challenges

We were perfect.

# 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 would test it.

## Testing Plan

<Describe your testing plan for the project.

TODO: Give a list of items or functions you want to test, and also a schedule for performing the testing. >

Laurel oversees the user sign in page. Here are the functions and classes that she would like 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) – searches for a user and creates an account.
* userInput(inputName, inputPassword) – decides what to do with the user input.
* confirmBox(inputName, inputPassword) – asks the user if they want to create an account or not.
* traverse() – opens the view.html file.

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 able 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: trying 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.

## Tests for Non-functional Requirements

<Similar to the Section 4.2, but this section is for the non-functional requirements. >

## Hardware and Software Requirements

<Describe the hardware and software requirements for performing the tests. >

# Analysis

<In this Section you need to analyze the effort that has been put on this project.

TODO: Describe how many hours (approximately) each team member spent on the project, for each milestone, which milestone took the most effort and why. >

Laurel Anderson

* Milestone 1: about 13 hours.
* Milestone 2: about 6 hours.
* Milestone 3: about 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.

# Conclusion

<Conclude the document with what you have learned through working on the project.>

We learned a lot of tech skills and soft skills. We learned how to coordinate on a team, especially a small team. We had a lot of “scrum meetings” and that helped us stay on track. Another soft skill we developed was working with the each of our schedules. We had to be flexible to have meetings or to get aspects of the project done.

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 learned how to apply JavaScript to HTML/CSS. We created fields and buttons in our “view” and connected them to a controller in JavaScript. The controller would then send the information to the JavaScript internal logic.

Appendix A - Group Log

< Describe how frequently the group members meet during the semester, and how effective the communication is. This is optional for one-person projects.>