WASHINGTON STATE UNIVERSITY
VANCOUVER

CS 320 Course Project Final Report

for

Journaler

Prepared by

Group Name: Dream Team

Malcolm Anderson	**011707961**	**malcolm.i.anderson@wsu.edu**
Antonio Maniscalco	**011467861**	**antonio.maniscalco@wsu.edu**
Chase Jamieson	**011732855**	**chase.jamieson@wsu.edu**

Date: 16 December 2020

Table of Contents

1 Introduction

This document describes what the Journaler web application is and how it was developed. Specific parts include:

- The processes by which user operations are completed and the structure of class objects (Section 2.1)
- The design of the software's user interface screens (Section 2.2)
- The languages, frameworks, and technologies that were used for the software (Section 3.1)
- The development environments and tools used by the developers when developing the software (Section 3.1)
- The distribution of tasks amongst team members while developing the software (Section 3.2)
- Tests that were planned and performed on the software, and their results (Section 4)

1.1 Project Overview

Journaler is a web application with a very simple goal: allow users to write and keep journal entries for reliving past moments. Users may view the journal entries they have written in a list, and click on individual entries to view their contents. Journal entries are private, so they are only accessible to the user who created them. They can be edited by the user at any time.

1.2 Definitions, Acronyms and Abbreviations

Note: This section is based off of section 1.4, “Definitions, Acronyms, and Abbreviations”, from the Milestone 1 document, and has been modified to fit this document.

This is a list of some of the more common technical terms that may be found in this document.

- **Backend** - The processes that run on internal servers, completely separated from user input. Secure database information, such as journal entries and user authentication information, may also be stored here.
- **Collection** - A related group of data in a database. For example, the group of all journal entries stored in the database would be considered one collection, as would the group of all user login information [1].

- **Frontend** - The processes that run on the user's computer. Processing here is limited to that which the user is authenticated to see by the backend, since in theory the user may be able to modify the code run by the frontend.
- **Identification (ID) Number** - A number or string meant to uniquely identify a piece of data.
- **Integrated development environment (IDE)** - A computer program that concentrates multiple aspects of software development into one interface, making development easier.
- **JavaScript** - A programming language used for creating interactive web applications.
- **Meteor** - A JavaScript framework for building web applications with a semi-unified frontend and backend code base [2].
- **React** - A JavaScript framework for creating responsive user interfaces [3].
- **Request** - The process by which the frontend asks the backend to take a specific action or provide a specific piece of information.
- **Response** - The process by which the backend responds to a request from the frontend.
- **Uniform Resource Locator (URL)** - An address that points to a specific location/page in the software. For example, "https://www.journal.app/entries " may point to a list of the currently logged in user's journals, and "https://www.journal.app/edit/0123 " may point to a page for editing a journal entry with the ID number 0123.

1.3 References and Acknowledgments

Note: This section is based off of section 1.6, “References and Acknowledgements”, from the Milestone 1 document, and has been modified to fit this document.

[1] “Collections and Schemas,” Meteor Guide. [Online]. Available: https://guide.meteor.com/collections.html. [Accessed: 07-Nov-2020].

[2] Meteor API Docs. [Online]. Available: https://docs.meteor.com/. [Accessed: 07-Nov-2020].

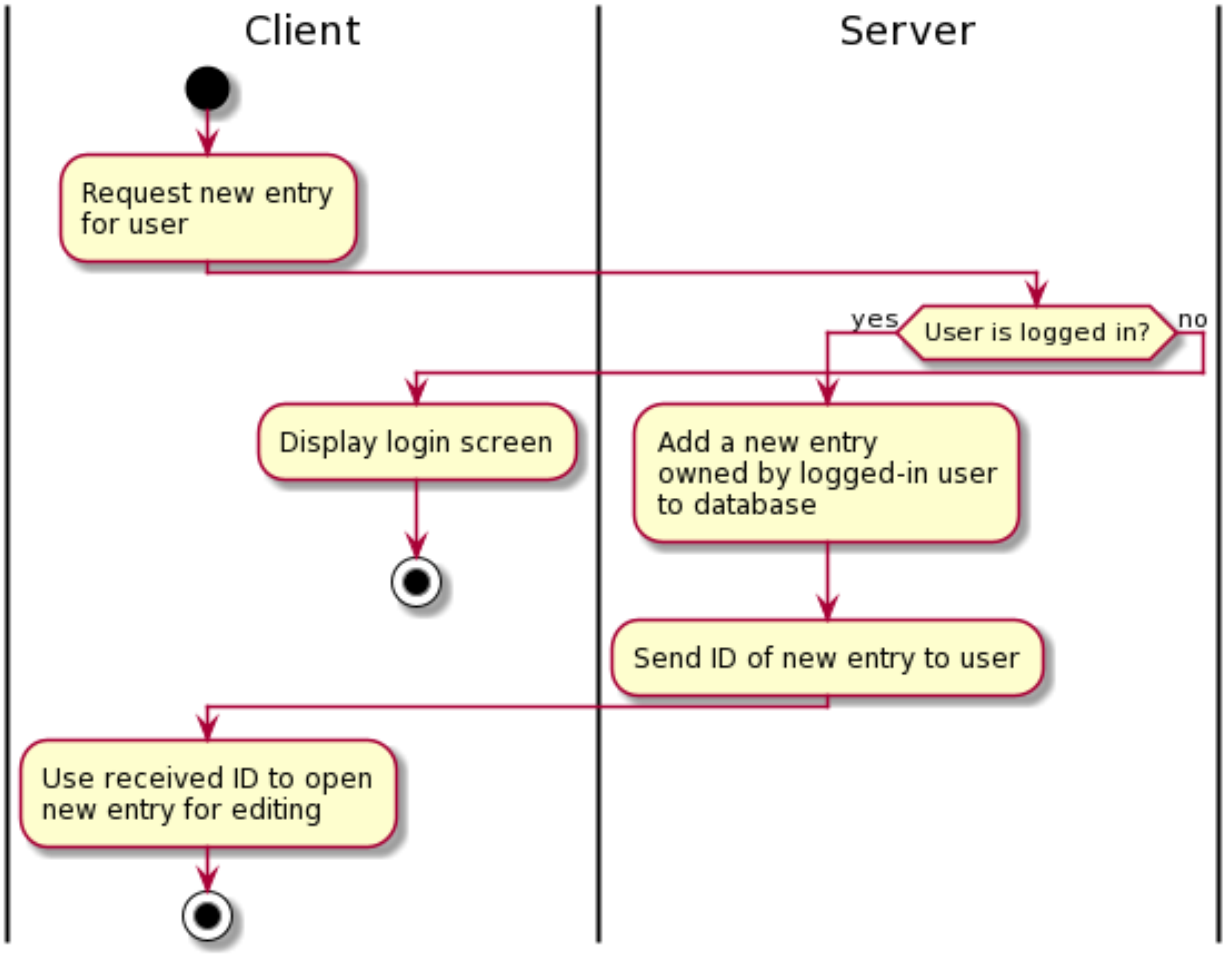
[3] “React,” Meteor Guide. [Online]. Available: https://guide.meteor.com/react.html. [Accessed: 17-Dec-2020].

2 Design

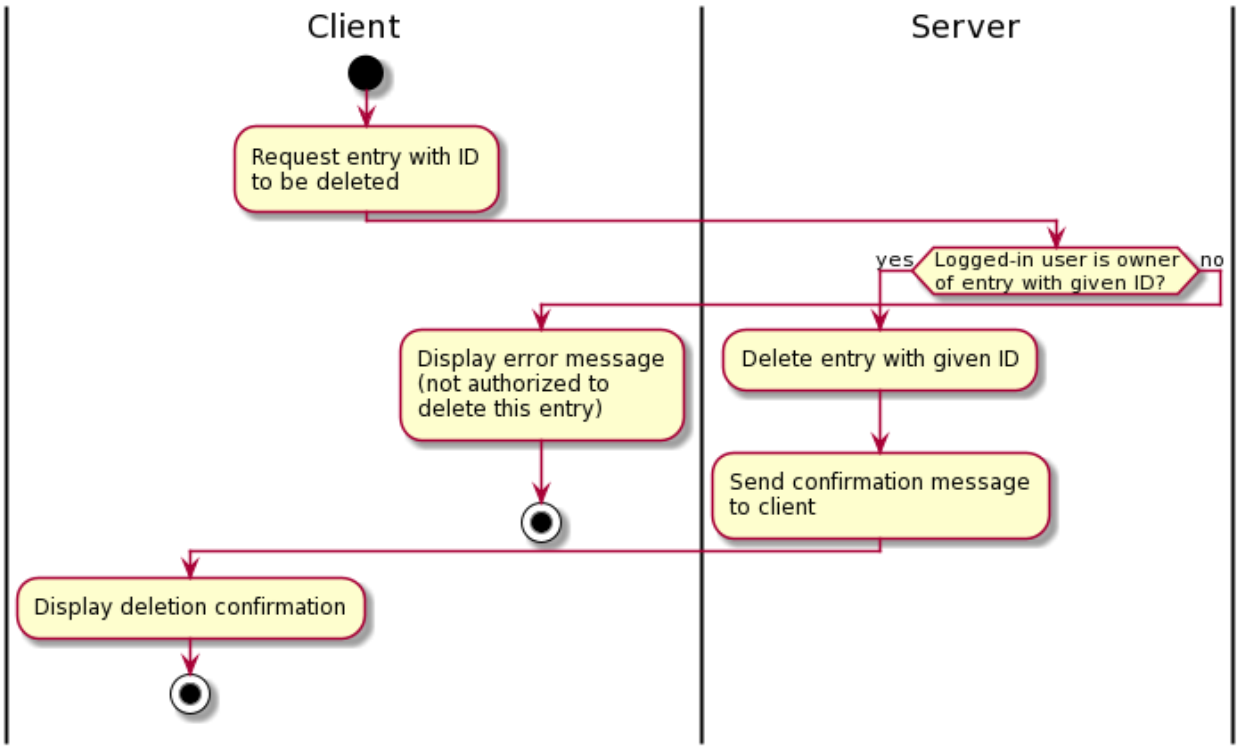
2.1 System Modeling

2.1.1 Activity Diagrams

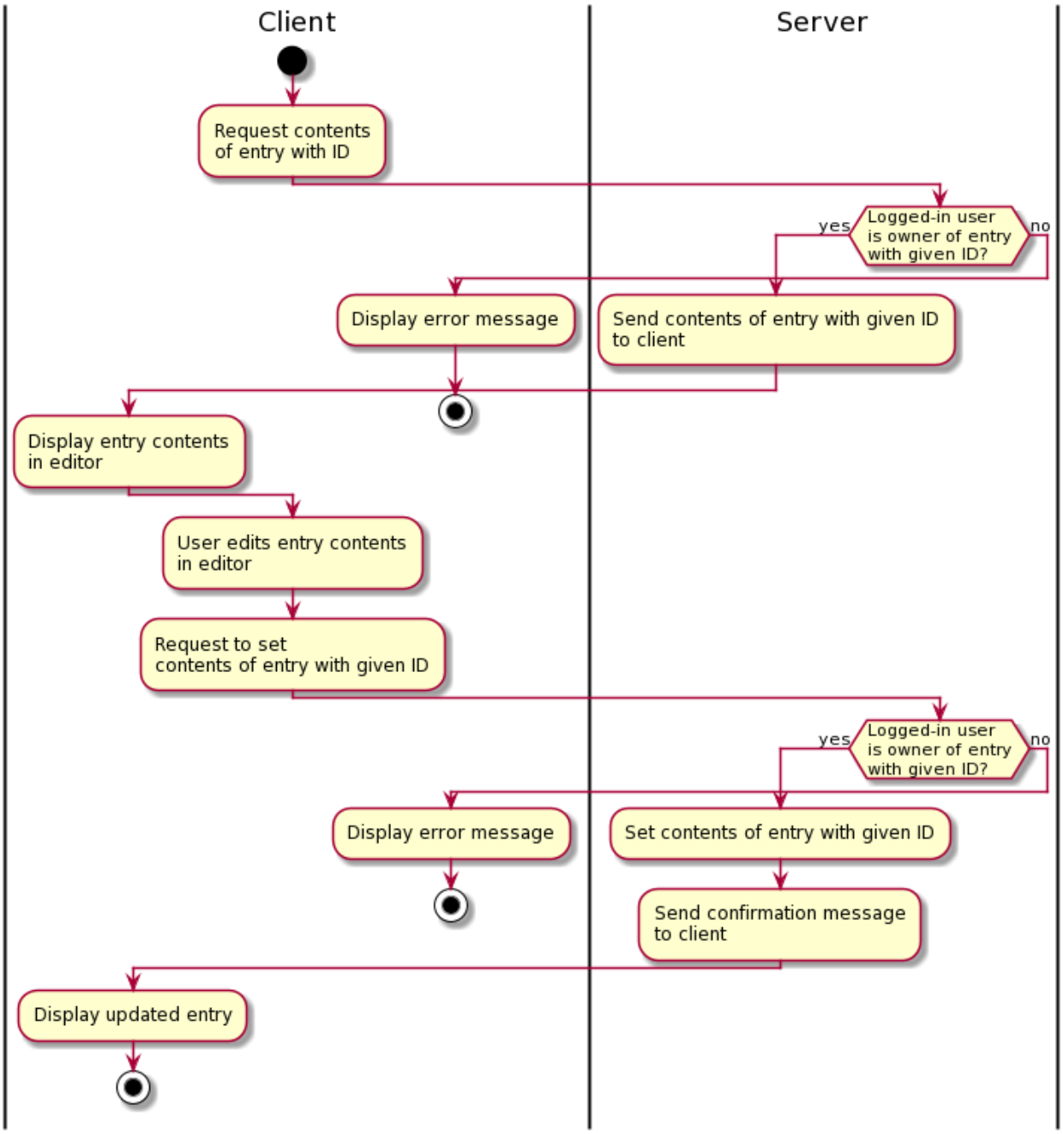
The activity diagrams for the Journaler software are mostly unchanged from those presented in Milestone 2. However, the decision was made to remove entry sharing functionality from this initial prototype. Instead, only the creator of a journal entry has the ability to view or edit it. The diagrams were modified to account for this change in scope.



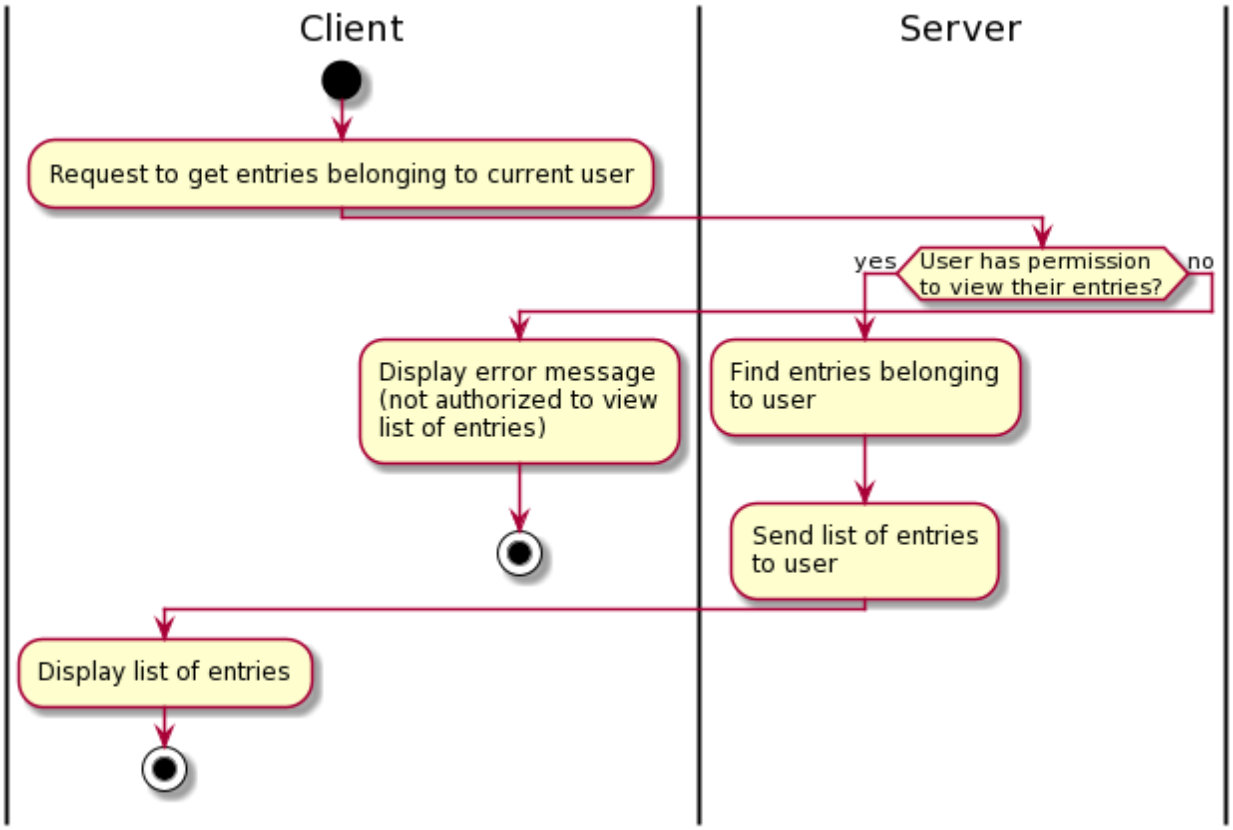
Creating a new journal entry. If a user is logged in, a new journal entry belonging to that user is created; if not, the user is redirected to the login screen.



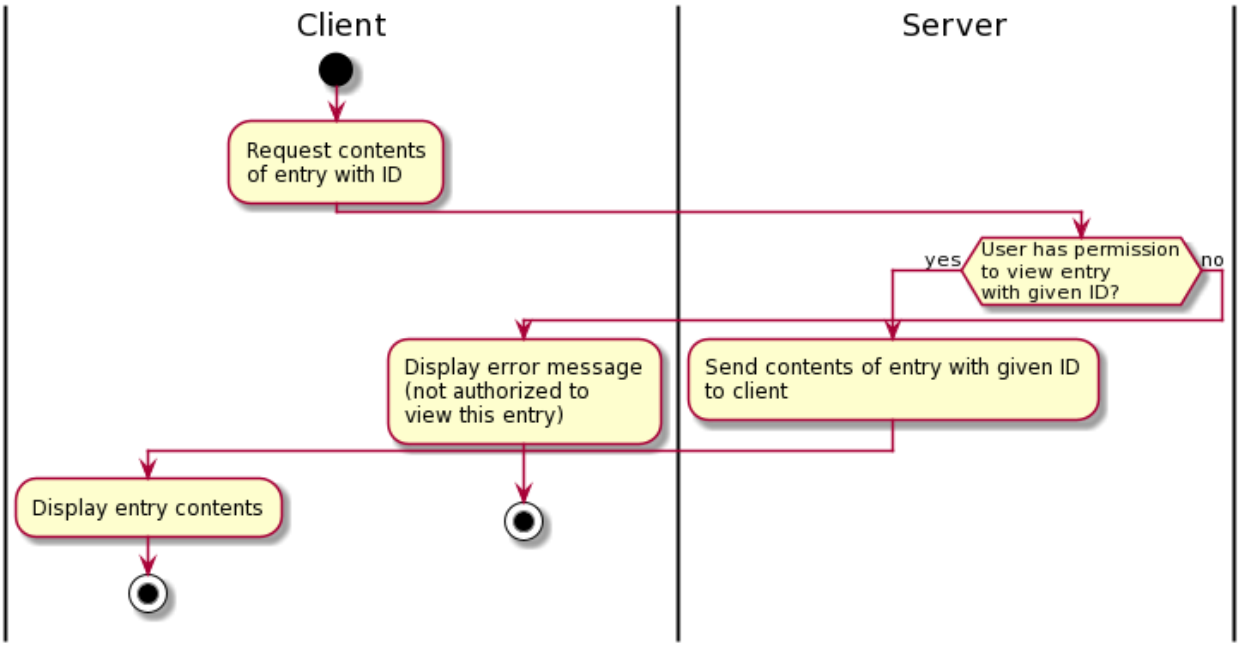
Deleting an existing journal entry. If the logged-in user is the owner of the given entry, the entry is deleted; otherwise, an error screen is shown.



Modifying an existing journal entry. If the currently logged-in user is the owner of the given journal entry, they are allowed to view and modify the contents of the journal entry; otherwise, an unauthorized error screen is displayed.

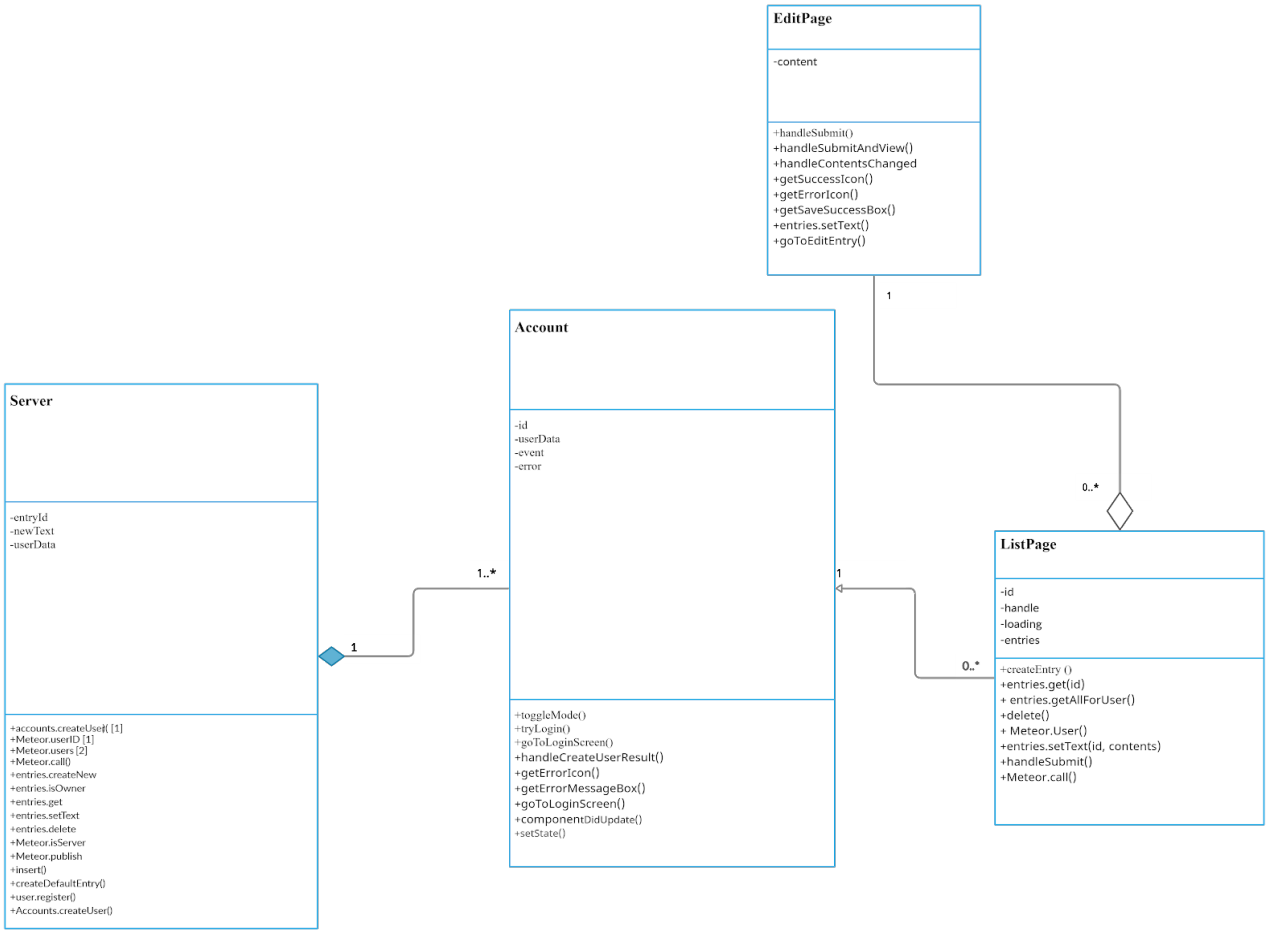


Retrieving a list of entries belonging to the currently logged-in user. If no user is currently logged in, an error message is displayed; otherwise, a list of all of the entries belonging to the currently logged-in user is retrieved.



Viewing an existing entry. If the currently logged-in user is the owner of the given entry, they are provided with the contents of the entry; otherwise, an error screen is shown.

2.1.2 Class Diagram

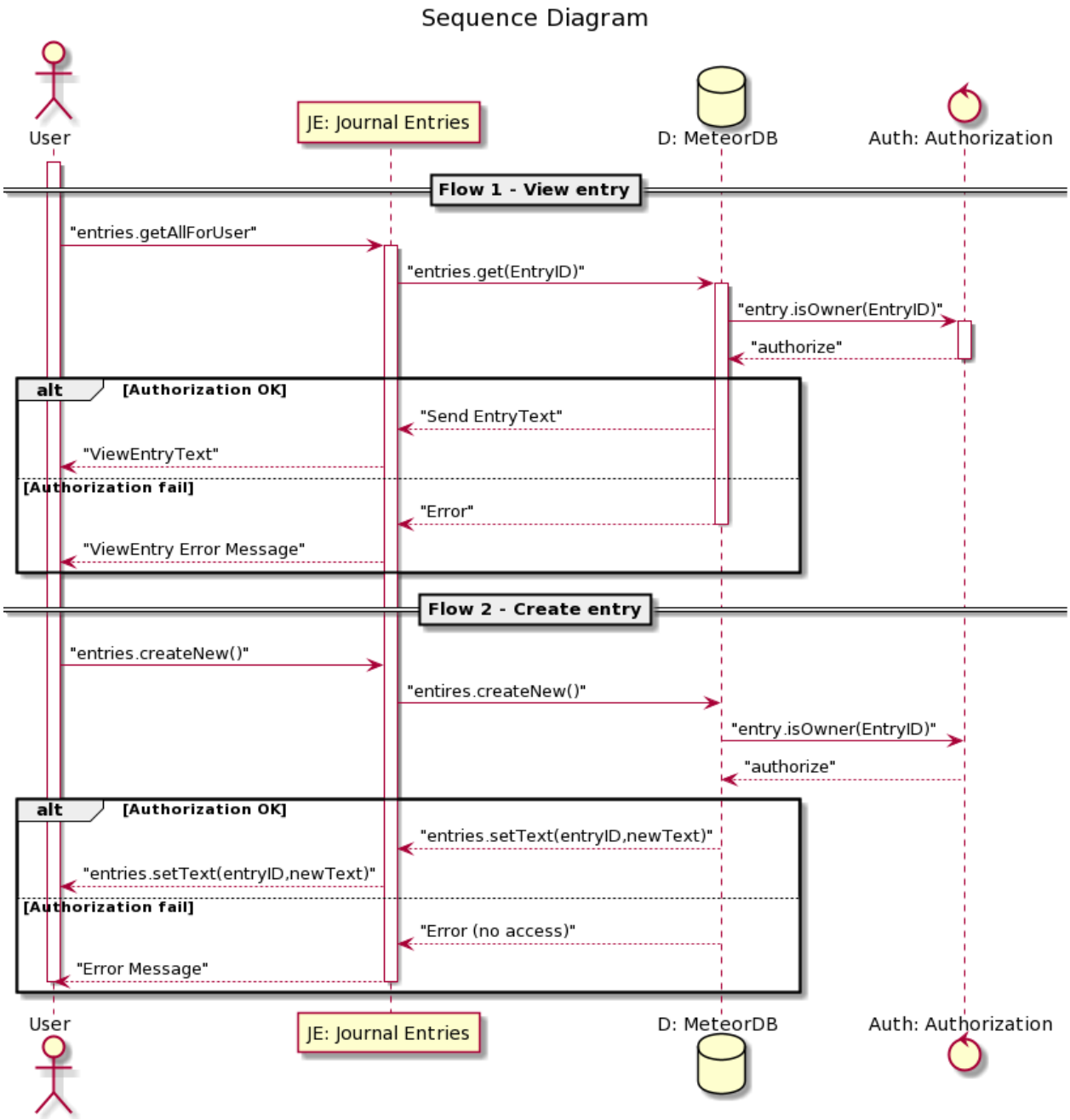


The class diagram is a bit different from the previous document. More specific functions and classes have been added. It also gives a better description of the structure of the app.

The Server class is composed of user accounts and entries. The server class deals with retrieving, and updating collections, as well as registering new accounts and checking user credentials. Account covers logging in, and creating a new account. The Account class inherits the user entries from the ListPage class where entries are retrieved from the database and displayed on the screen. The user can also select a specific entry from the list and view it in this class. The EditPage class takes a selected entry from the list of entries, and allows the user to edit the entry. EditPage also allows the user to create a new entry and updates the database.

2.1.3 Behavioral Diagram

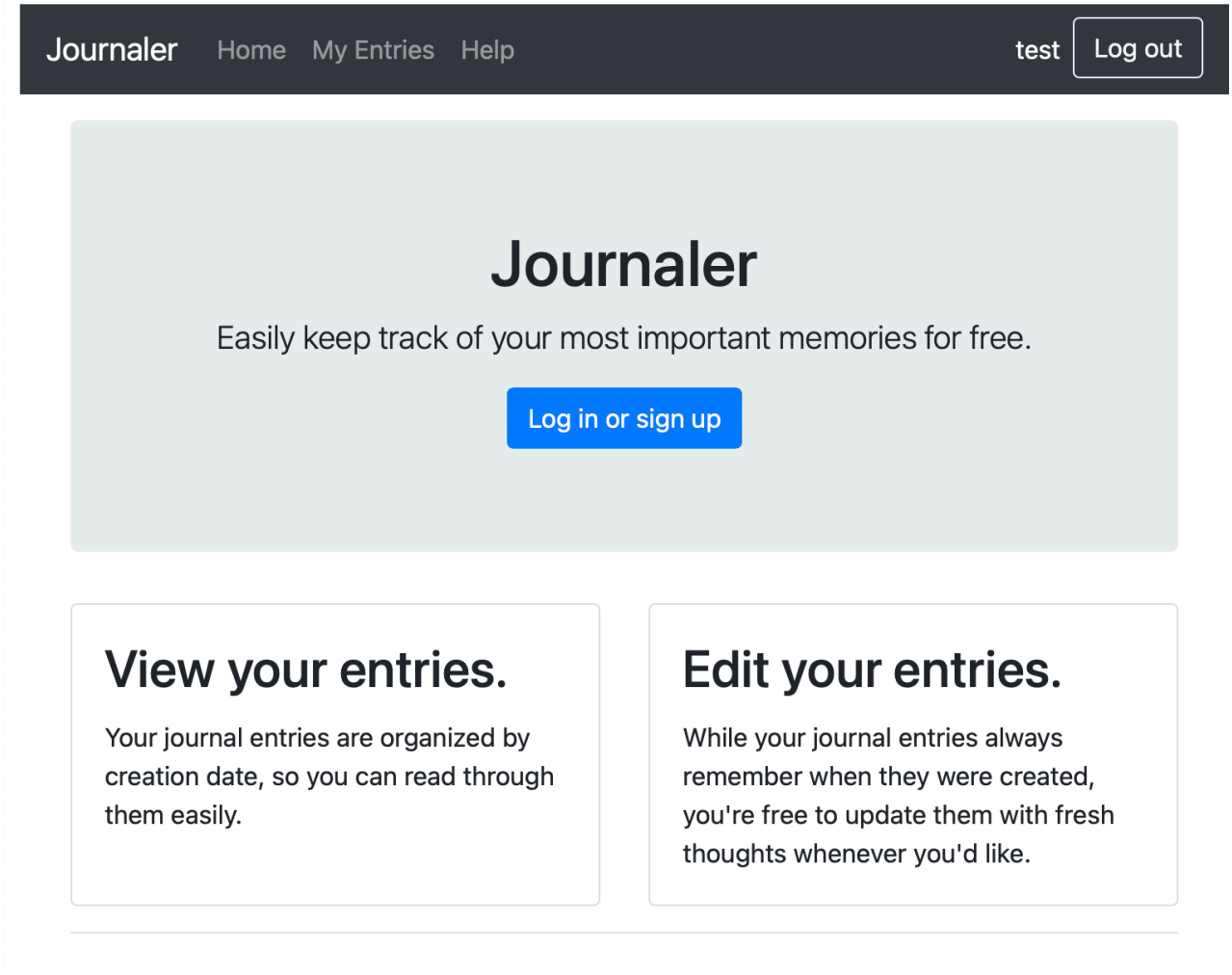
The Sequence diagram below shows interactions of objects in a time sequence. The vertical lines are different processes happening simultaneously. Our diagram is separated into two flows for viewing a pre-existing entry and for creating a new one. In each of the flows, The user must pass the authorization Ok box in order to successfully create or view an entry.



2.2 Interface Design

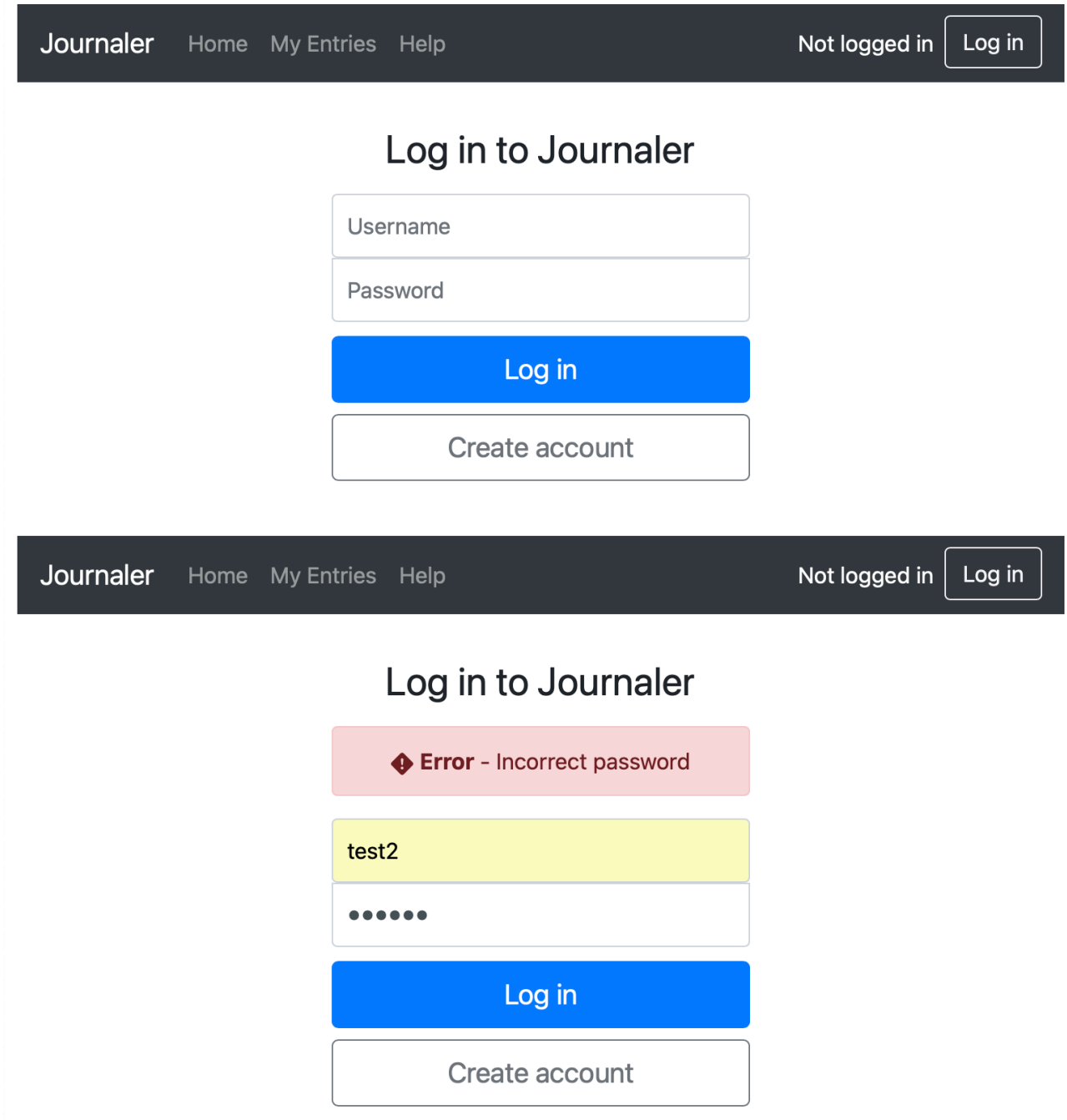
2.2.1 Home Page

A basic home/landing page was designed for the Journaler application. It describes the two main features of the software - viewing and editing entries - as well as an easy-to-use button for logging in or signing up.

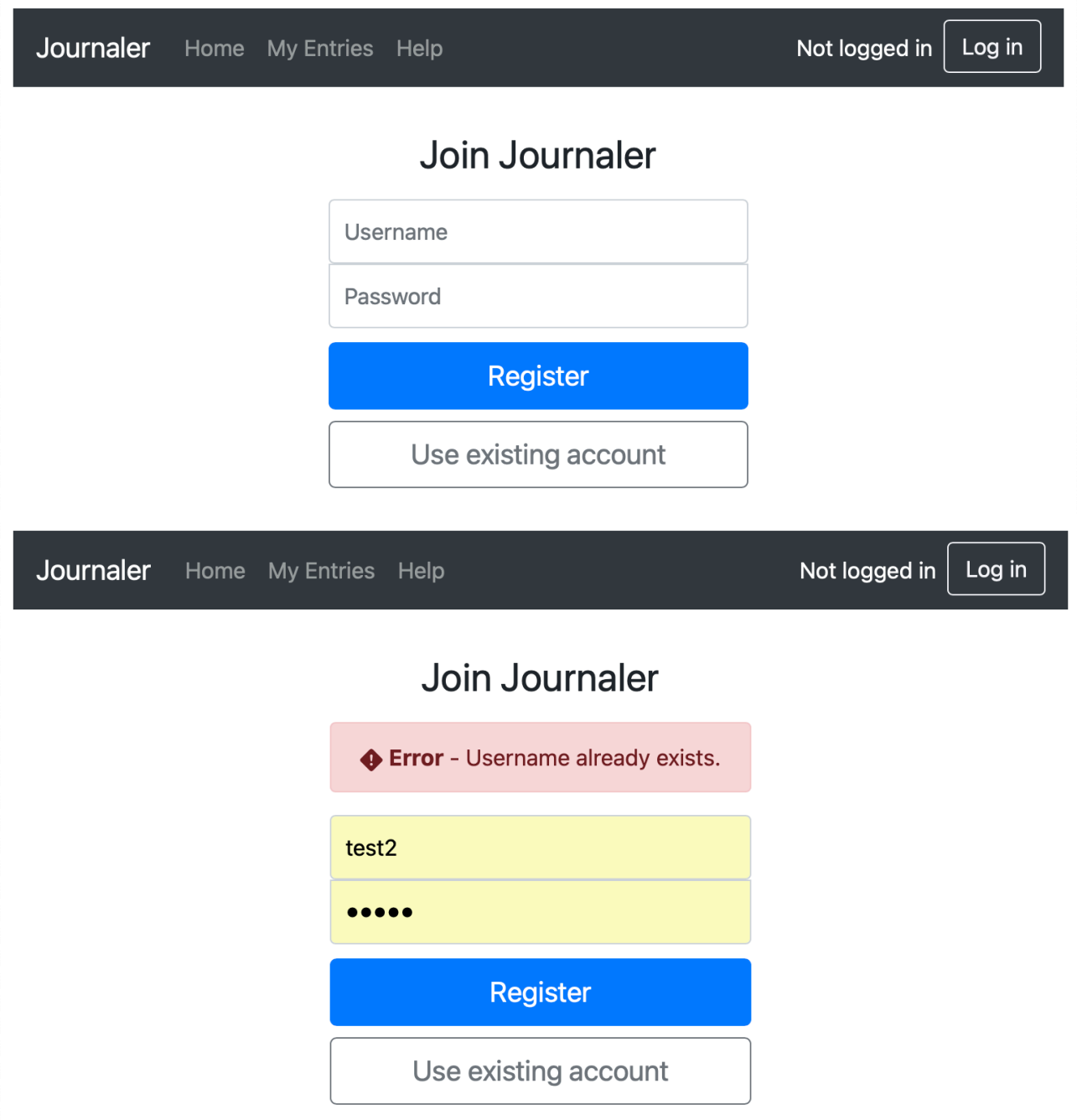


2.2.2 Login/Signup Page

The login and signup screens for Journaler were combined into a single page. Upon loading the page, the software asks for credentials to sign into an existing account. If improper credentials are entered, an error box appears telling the user that they could not be logged in.

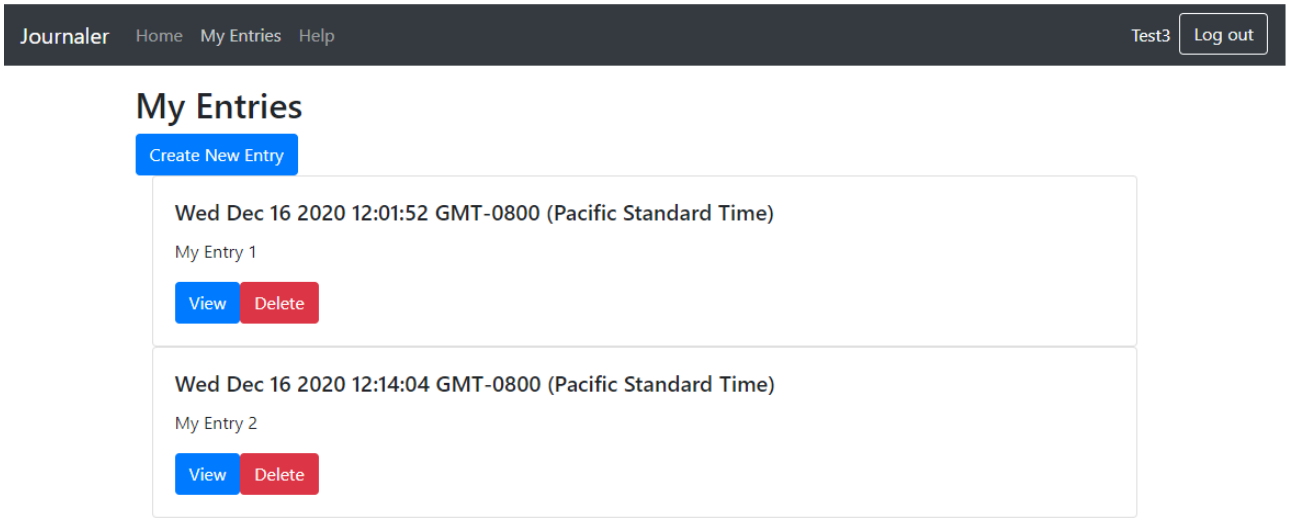


Clicking the "Create account" button changes the page to the "create account" mode. At this point, the user may enter a username and password they want their account to have and click "Register" to create their account. If the user attempts to create an account with a username that is already in use, an error message is shown and the user's account is not created.



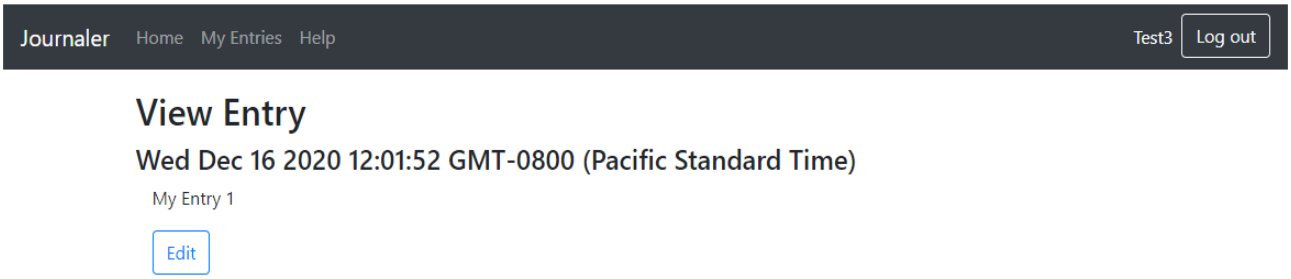
2.2.3 List Entries Page

The List Entries page provides the user with a list of the entries that they have authored. Each entry displays the date and time of its creation and its contents, along with "View" and "Delete" buttons to perform those actions. The list page also has a "Create New Entry" button.



2.2.4 View Entry Page

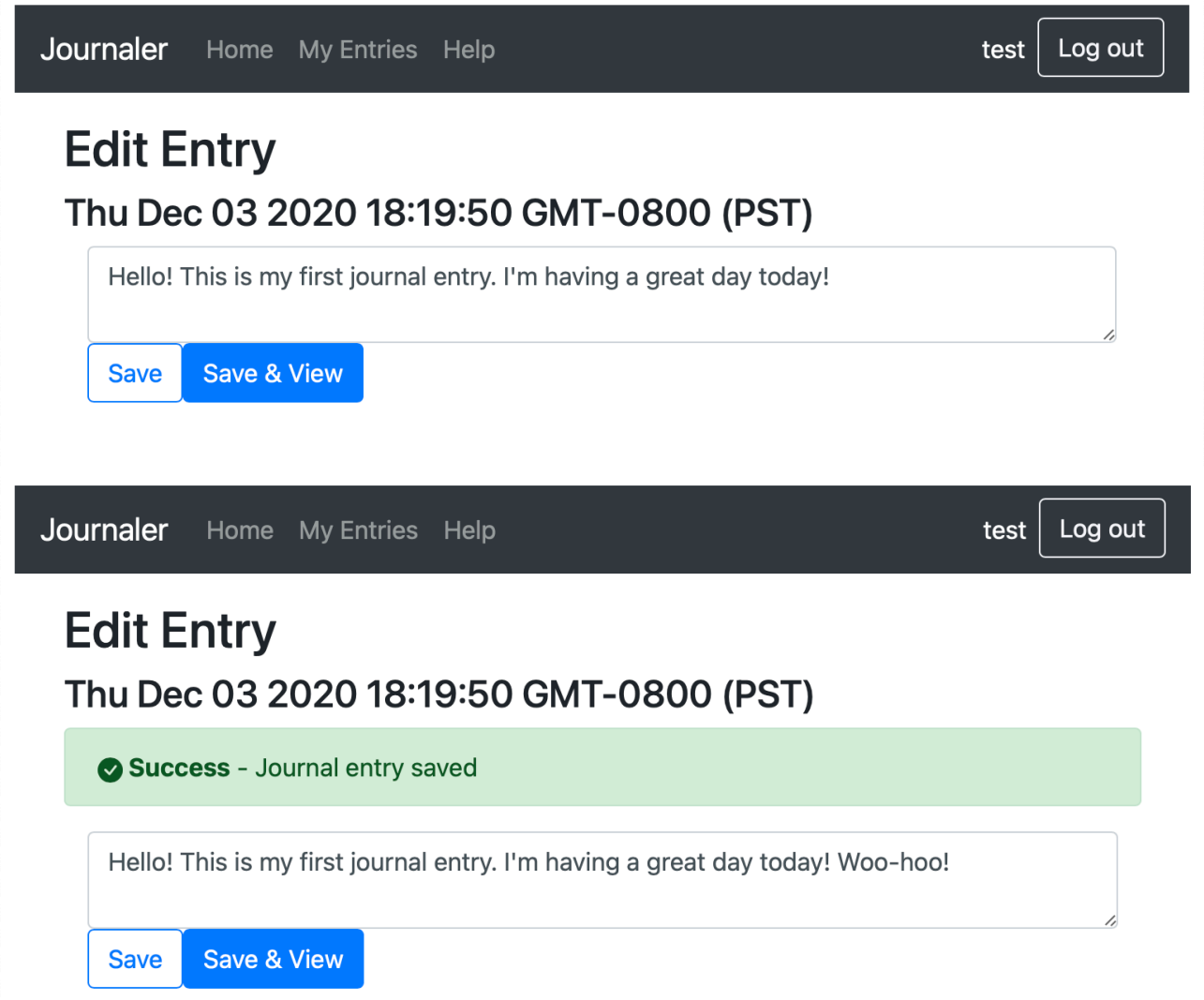
The View Entry page displays information about a single entry. This includes the date and time of the entry's creation and its current contents. The user can edit the entry's contents by clicking the "Edit" button, but doing so does not alter the date or time associated with the entry.



2.2.5 Edit Entry Page

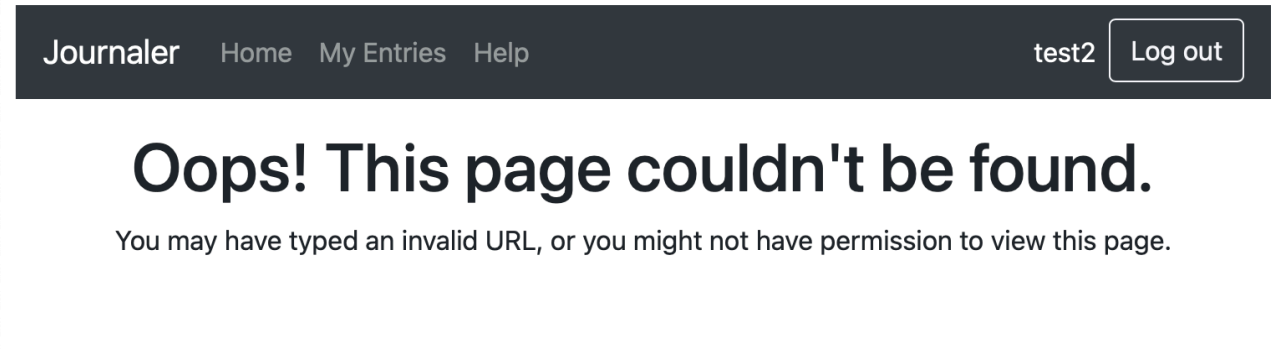
When creating or editing an entry, the user is provided with a text box to write in. The text box is pre-populated with the "current" contents of the entry. A brand new entry's contents are an empty string, which means that the text box appears empty to the user.

Along with the text box are two buttons for saving the journal entry's contents. The first button, "Save", saves the updated entry by setting its contents to whatever is in the text box. A "Success" box appears above the text box when the entry's contents are successfully saved. The second button, "Save & View", saves the updated entry in the same manner, but immediately moves the user to the "View Entry" page for that entry.



2.2.6 Unauthorized Page

When a user attempts to access an invalid page on the Journaler website, or a journal entry that they are not authorized to view (i.e. they are not the creator of the entry), they are given a generic 404 error page.



3 Implementation

3.1 Development Environment

3.1.1 Frameworks and Technologies

The Journaler software is written in JavaScript with the React UI framework. Meteor is used for the software's frontend and backend logic. The webpages themselves use Bootstrap for UI components and element styling.

3.1.2 Development Environments

While working on the project, Malcolm used the Visual Studio Code editor on the macOS Big Sur operating system, and tested the software's functionality in the Safari web browser. Antonio used the IntelliJ IDE on the Linux Mint operating system and tested using the Google Chrome web browser. Chase used the Visual Studio Code editor on the Windows operating system and tested using the Google Chrome and Firefox web browsers.

3.2 Task Distribution

Malcolm was assigned to create the software's Home page, the Login/Signup page, the Edit Entry page, and the Unauthorized/Error page. Additionally, he designed the data structure representing journal entries, set up the Meteor backend, and performed unit tests for quality assurance. Malcolm tested the software on the macOS operating system to ensure it behaved correctly.

Antonio was assigned to create the software's View Entry and List Entry pages, as well as perform unit tests for quality assurance. Antonio tested the software on the Windows operating system to ensure it behaved correctly.

Chase was assigned to create the user help documentation page for the software, as well as perform unit tests on the software for quality assurance. Chase also tested the software on Windows to ensure it behaved correctly.

3.3 Challenges

There were no particularly challenging aspects of developing the software that the team encountered.

4 Testing

4.1 Testing Plan

Sections 4.2 "Tests for Functional Requirements" and 4.3 "Tests for Non-Functional Requirements" list the individual unit tests for the software. Due to the user interface-heavy nature of the software, the choice was made to forgo automated unit testing in favor of manual testing procedures.

4.2 Tests for Functional Requirements

4.2.1 The user must be able to create a new account with a given username and password.

Steps:

1. Ensure that no user is currently logged into the system.
2. From the homepage, click "Log in".
3. On the login screen, click "Create account".
4. In the username field, type "Test3". In the password field, type "hunter2".
5. Click "Register".
6. If the webpage redirects to the "My Entries" page, with "Test3" in the top right corner, the test SUCCEEDED. If an error message appears, the test FAILED.

Result(s):
15 December 2020 at 8:57 PM, Malcolm – Success
16 December 2020 at 11:30 AM, Antonio – Success

4.2.2 The user must be able to log into an existing account with a given username and password.

This test relies on test 4.2.1 having succeeded.
Steps:

1. Ensure that no user is currently logged into the system.
2. From the homepage, click "Log in".
3. In the username field, type "Test3". In the password field, type "hunter2".
4. Click "Log in".
5. If the webpage redirects to the "My Entries" page, with "Test3" in the top right corner, the test SUCCEEDED. If an error message appears, the test FAILED.

Result(s):
15 December 2020 at 9:00 PM, Malcolm – Success

16 December 2020 at 11:43 AM, Antonio – Success

4.2.3 The user must not be logged into an existing account if they provide the incorrect password.

This test relies on test 4.2.1 having succeeded.
Steps:

1. Ensure that no user is currently logged into the system.
2. From the homepage, click "Log in".
3. In the username field, type "Test3". In the password field, type "Wrong".
4. Click "Log in".
5. If an error message reading "Error - Incorrect password" appears, the test SUCCEEDED. If the webpage redirects to the "My Entries" page, with "Test3" in the top right corner, the test FAILED.

Result(s):
15 December 2020 at 9:08 PM, Malcolm – Success
16 December 2020 at 11:46 AM, Antonio – Success

4.2.4 A new account must not be created if there is an existing account with the same username.

This test relies on test 4.2.1 having succeeded.
Steps:

1. Ensure that no user is currently logged into the system.
2. From the homepage, click "Log in".
3. On the login screen, click "Create account".
4. In the username field, type "Test3". In the password field, type "newpassword".
5. Click "Register".
6. If an error message reading "Error - Username already exists" appears, the test SUCCEEDED. If the webpage redirects to the "My Entries" page, with "Test3" in the top right corner, the test FAILED.

Result(s):
15 December 2020 at 9:10 PM, Malcolm – Success
16 December 2020 at 11:47 AM, Antonio – Success

4.2.5 The user must be able to create a new entry and have themselves assigned as its owner.

This test relies on test 4.2.1 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. Navigate to the "New Entry" page (at the URL localhost:3000/new or via the user interface from the "My Entries" page).
3. If the "Edit Entry" page appears with the current date and time and an empty text box (placeholder text reading "What's going on?"), continue. If anything else happens, the test FAILED.
4. Type "Hello world, this is a journal entry!" in the text box.
5. Click "Save". If the message "Success - Journal entry saved" appears, continue. If anything else happens, the test FAILED.
6. Add "Goodbye for now!" to the contents of the text box.
7. Click "Save & View". If the software redirects to the "View Entry" page, with the date and time of the entry's creation, and the text "Hello world, this is a journal entry! Goodbye for now!" below it, the test SUCCEEDED. Otherwise, the test FAILED.

Result(s):
15 December 2020 at 9:39 PM, Malcolm – Success
16 December 2020 at 11:48 PM, Antonio – Success

4.2.6 The user must be able to view a list of entries they own.

This test relies on test 4.2.5 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, an entry should be listed with the contents "Hello world, this is a journal entry! Goodbye for now!". If not, the test FAILED.
3. Create a new entry and set its contents to "This is my second entry." (see test 4.2.5).
4. Navigate to the "My Entries" page.
5. On the "My Entries" page, two entries should be listed: one with the contents "Hello world, this is a journal entry! Goodbye for now!", and another one with the contents "This is my second entry.". If this is not the case, the test FAILED.

Result(s):
15 December 2020 at 9:43 PM, Malcolm – Success
16 December 2020 at 11:49 AM, Antonio – Success

4.2.7 The user must be able to view the contents of an entry they own.

This test relies on test 4.2.6 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, click the "View" button underneath the entry with the contents "Hello world, this is a journal entry! Goodbye for now!".
3. If the "View Entry" page appears with the same contents, the test SUCCEEDED. If not, the test FAILED.

Result(s):
15 December 2020 at 9:49 PM, Malcolm – Success
16 December 2020 at 11:50 AM, Antonio – Success

4.2.8 The user must be able to edit and save the contents of an entry they own.

This test relies on test 4.2.7 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, click the "View" button underneath the entry with the contents "Hello world, this is a journal entry! Goodbye for now!".
3. Click the "Edit" button.
4. The "Edit Entry" page should appear with the same contents, now in an editable text field. If not, the test FAILED.
5. Replace the contents of the text field with "I'm feeling great today!".
6. Click "Save". If the message "Success - Journal entry saved" appears, continue. If anything else happens, the test FAILED.
7. Add "Really great, actually!" to the contents of the text box.
8. Click "Save & View". If the software redirects to the "View Entry" page, with the date and time of the entry's creation, and the text "I'm feeling great today! Really great, actually!" below it, the test SUCCEEDED. Otherwise, the test FAILED.

Result(s):
15 December 2020 at 9:51 PM, Malcolm – Success
16 December 2020 at 12:02 PM, Antonio – Success

4.2.9 The user must *not* be able to view the contents of an entry if they are logged out.

This test relies on test 4.2.8 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).

2. On the "My Entries" page, click the "View" button underneath the entry with the contents "I'm feeling great today! Really great, actually!".
3. Copy the browser's current URL (should be in the format localhost:3000/view/[identifier]).
4. Click "Log out" in the top right corner of the screen.
5. Paste the URL copied in step 3 into the browser URL bar and press Enter/Return to navigate to that page. The software should redirect to an error page, with the text "Oops! This page couldn't be found.". If so, the test SUCCEEDED; if not, the test FAILED.

Result(s):
15 December 2020 at 9:56 PM, Malcolm – Success
16 December 2020 at 12:03 PM, Antonio – Success

4.2.10 The user must *not* be able to edit the contents of an entry if they are logged out.

This test relies on test 4.2.8 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, click the "View" button underneath the entry with the contents "I'm feeling great today! Really great, actually!".
3. Click "Edit".
4. Copy the browser's current URL (should be in the format localhost:3000/view/[identifier]).
5. Click "Log out" in the top right corner of the screen.
6. Paste the URL copied in step 4 into the browser URL bar and press Enter/Return to navigate to that page. The software should redirect to an error page, with the text "Oops! This page couldn't be found.". If so, the test SUCCEEDED; if not, the test FAILED.

Result(s):
15 December 2020 at 9:58 PM, Malcolm – Success
16 December 2020 at 12:04 PM, Antonio – Success

4.2.11 A second user account can be created.

This is important for subsequent tests.
Steps:

1. Ensure that no user is currently logged into the system.

2. From the homepage, click "Log in".
3. On the login screen, click "Create account".
4. In the username field, type "SecondUser". In the password field, type "pass".
5. Click "Register".
6. If the webpage redirects to the "My Entries" page, with "SecondUser" in the top right corner, the test SUCCEEDED. If an error message appears, the test FAILED.

Result(s):
15 December 2020 at 10:01 PM, Malcolm – Success
16 December 2020 at 12:05 PM, Antonio – Success

4.2.11 The user must *not* be able to view the contents of an entry they do not own.
This test relies on tests 4.2.8 and 4.2.11 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, click the "View" button underneath the entry with the contents "I'm feeling great today! Really great, actually!".
3. Copy the browser's current URL (should be in the format localhost:3000/view/[identifier]).
4. Click "Log out" in the top right corner of the screen.
5. Log into the account "SecondUser" using password "pass" (as created in test 4.2.11).
6. Paste the URL copied in step 3 into the browser URL bar and press Enter/Return to navigate to that page. The software should redirect to an error page, with the text "Oops! This page couldn't be found.". If so, the test SUCCEEDED; if not, the test FAILED.

Result(s):
15 December 2020 at 10:04 PM, Malcolm – Success
16 December 2020 at 12:08 PM, Antonio – Success

4.2.12 The user must *not* be able to edit the contents of an entry they do not own.
This test relies on test 4.2.8 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. On the "My Entries" page, click the "View" button underneath the entry with the contents "I'm feeling great today! Really great, actually!".

3. Click "Edit".
4. Copy the browser's current URL (should be in the format localhost:3000/view/[identifier]).
5. Click "Log out" in the top right corner of the screen.
6. Log into the account "SecondUser" using password "pass" (as created in test 4.2.11).
7. Paste the URL copied in step 4 into the browser URL bar and press Enter/Return to navigate to that page. The software should redirect to an error page, with the text "Oops! This page couldn't be found.". If so, the test SUCCEEDED; if not, the test FAILED.

Result(s):
15 December 2020 at 10:06 PM, Malcolm – Success
16 December 2020 at 12:13 PM, Antonio – Success

4.3 Tests for Non-functional Requirements

4.3.1 The user must be able to create a new journal entry in less than 5 seconds.

This test relies on test 4.2.5 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).
2. Navigate to the "New Entry" page (at the URL localhost:3000/new or via the user interface from the "My Entries" page). Start a stopwatch as soon as the page is navigated to.
3. If the stopwatch is below 5 seconds (5000 ms) by the time the "Edit Entry" page appears, the test SUCCEEDED; otherwise, the test FAILED.

Result(s):
16 December 2020 at 10:52 AM, Malcolm – Success
16 December 2020 at 12:14 PM, Antonio – Success

4.3.2 The user must be able to access an existing journal entry (that they own) in less than 5 seconds.

This test relies on test 4.2.7 having succeeded.
Steps:

1. Log into the account "Test3" (see test 4.2.2).

2. On the "My Entries" page, click the "View" button underneath the entry with the contents "Hello world, this is a journal entry! Goodbye for now!". Start a stopwatch as soon as the button is clicked.
3. If the stopwatch is below 5 seconds (5000 ms) by the time the "View Entry" page appears, the test SUCCEEDED; otherwise, the test FAILED.

Result(s):
16 December 2020 at 10:58 AM, Malcolm – Success
16 December 2020 at 12:14 PM, Antonio – Success

4.4 Hardware and Software Requirements

The tests were performed both on macOS Big Sur and a Linux Mint distro, and compatibility was also tested on Windows. The Journaler software requires that a keyboard and mouse be connected to the computer. To run the software (and thus the tests described above), the Meteor framework had to be installed. Instructions for installing the Meteor framework can be found at https://www.meteor.com/install or in the document "Meteor_Install_Instructions.md" within the repository.

5 Analysis

5.1 Milestone 1

For Milestone 1, the precise workload or number of person-hours spent was difficult to quantify, since the SRS document was drafted while the team was also researching ways to have more specific descriptions, implementations, and requirements. Every member was able to complete their assigned tasks, due to the indirect time spent researching each member approximately spent 6-12 hours from initial discussion to final draft.

5.2 Milestone 2

For Milestone 2, the three diagram types were distributed between team members. Malcolm created the activity diagrams that depicted the process for creating, editing, and viewing entries. Antonio developed the Class diagram which covered the four classes used in the system. Chase was tasked with designing the behavioral diagrams for the system; however he did not submit his work by the deadline (it has since been completed - see section 2.1.3). This milestone took less time than Milestone 1 and 3. The time spent by each team member on their Milestone 2 diagrams was approximately 2-4 hours.

5.3 Milestone 3

Out of the three major milestones, completing Milestone 3 definitely took the most effort from the team members. On top of the Journaler software itself being developed, the Milestone 3 document describing the implementation of the contents of Milestones 1 and 2 was also written.

Development on the software went fairly smoothly. Two of the three team members did not have prior experience using the Meteor or React frameworks, but quickly learned how to use them effectively. Individual tasks for developing software components were split amongst the team members so that each member had specific goals to reach.

More challenges were faced while coordinating the sections of the Milestone 3 document. Sections were distributed amongst team members, but not as finely as for the software itself. As a result, there was some confusion as to how sections of the document were to be completed by the team members.

Between the development of the software itself and the authoring of the report document, it is estimated that each team member spent around 8-16 hours on this milestone.

6 Conclusion

There are three distinct areas in which the team learned a lot through working on the project: the importance of working with familiar frameworks, using similar development environments, and sharing similar coding conventions.

When the team began developing the project, two of the group members had very little to zero experience working with the Meteor and React frameworks which slowed the development process. While these members were still able to contribute, using more familiar frontend and backend frameworks may have improved development time.

During the development process, one member was working through VS Code on macOS with testing in the Safari web browser and the other two were using IntelliJ on a Windows PC. This led to some errors which were specific to some team members' machines, and added extra steps in the testing process where the team had to ensure the software worked on each machine. However, this also helped to ensure that the software was stable across operating systems and environments.

Coding conventions involve components such as commenting styles, naming rules, and indentation. There was not a unified naming scheme for the software's methods and objects, which led to some inconsistencies between the class/behavioral diagrams and the code base. Commenting styles were also not unified, which led to some code being better-documented than other code. These small differences accumulated and led to a longer development time. Spending more time defining and adhering to a set of coding conventions for future projects may drastically improve time spent on development.

Appendix A - Group Log

Date	Description
28 September 2020	● Setting up group ● Discussing project ideas ○ Trivia game ○ Survey software ○ Web scraper ○ "Tinder for DnD players" ○ Project idea finder
2 October 2020	● Exchanging information ● Deciding how to structure project group ● Deciding when to hold biweekly meeting times ○ Decided on Tues/Thurs at 3 PM, when there are matters to discuss
5 October 2020	● Finalizing team hierarchy ● Discussing project ideas ○ Journaling software ● Finalizing Team Formation Agreement
8 October 2020	● Setting up Google Docs document for collaborating on SRS
11 October 2020	● SRS template sections and hints transferred to Google Docs document
13 October 2020	● Brief meeting ● Work on section 1 of SRS started
18 October 2020	● First work on section 2 of SRS completed
24 October 2020	● Quick check-in
27 October 2020	● Quick check-in
30 October 2020	● Quick check-in ● Functional requirements added ● Non-functional requirements added ○ Performance

	○ Safety and security
4 November 2020	● Started transferring writing from Google Docs document to SRS Word document template ● Requested student ID numbers for document
5 November 2020	● Jamieson's student ID number added to document
6 November 2020	● GitHub repository created with standard Meteor project and SRS Word document
9 November 2020	● Quick check-in to get development environment set up
10 November 2020	● Quick check-in to get development environment set up
11 November 2020	● Quick check-in to get development environment set up
12 November 2020	● Quick check-in to get development environment set up
13 November 2020	● Quick check-in to schedule formal meeting and get development environment set up
15 November 2020, 16 November 2020	● Workload for Milestone 2 distributed among team members
17 November 2020	● Activity diagrams created and pushed to repository, began filling out Milestone 2 template
18 November 2020	● Finished writing activity diagram descriptions and some parts of introduction
19 November 2020	● Quick check-in to discuss project and Milestone 2
21 November 2020	● Class diagram with description added to document
25 November 2020	● Remaining work on project split between Antonio and Malcolm
26 November 2020	● UI layouts for Edit Entry and Unauthorized pages created, pushed to repository ● Account system mostly implemented, pushed to repository

27 November 2020	● Account system with login/signup screen implemented, pushed to repository
1 December 2020	● Scheduled meeting for 3 December 2020
3 December 2020	● Discussed modifying group progress deadlines ● Journal entry data structure and database implemented, pushed to repository and merged with master branch
6 December 2020	● Check-in to gauge progress ● Made decision to forgo "view/edit" permissions on individual entries, and instead only allow entry owner to view/edit an entry
7 December 2020	● Decided on a time to do the class presentation (16 Dec at 1 PM)
9 December 2020	● Gave Chase access to documents and repository ● Assigned Chase to work on user documentation for software ● Began working on Milestone 3 document
11 December 2020	● Check-in to gauge progress ● Entry list and entry view placeholder prototypes implemented for testing purposes
12 December 2020	● Check-in with other members to gauge progress ● Antonio offers to work on testing part of Milestone 3 document
13 December 2020	● Communication to get code base re-synchronized
14 December 2020	● Communication to get code base re-synchronized ● Chase added behavioral diagram to Milestone 3 document
15 December 2020	● Bug involving creation of new entries fixed ● Functional requirements testing performed ● Class diagram updated
16 December 2020	● Final versions of Entry List and View Entry pages created ● Non-functional requirements testing performed ● Software presented

	● Repository restructured to fit project requirements ● Submitted