WASHINGTON STATE UNIVERSITY
VANCOUVER

CS 320 Course Project - Software Design Document

for

Journaling Software

Prepared by

Group Name: Dream Team

Malcolm Anderson	**011707961**	**malcolm.i.anderson@wsu.edu**
Tony Maniscalco	**011467861**	**antonio.maniscalco@wsu.edu**
Chase Jamieson	**11732855**	**chase.jamieson@wsu.edu**

Date: 21 November 2020

1 INTRODUCTION 1

1.1 PROJECT OVERVIEW 1
1.2 DEFINITIONS, ACRONYMS AND ABBREVIATIONS 1
1.3 REFERENCES AND ACKNOWLEDGMENTS 1

2 ACTIVITY DIAGRAM(S) 2

2.1 CREATING AN ENTRY 2
2.2 EDITING AN ENTRY 3
2.3 DELETING AN ENTRY 4
2.4 VIEWING AN ENTRY 5
2.5 VIEWING A LIST OF ENTRIES 6

3 CLASS DIAGRAM(S) 7

3.1 STRUCTURAL VIEW 7

4 BEHAVIORAL DIAGRAM(S) 8

1 Introduction

1.1 Project Overview

The software being developed is a journaling application. Users may compose journal entries (referred to throughout the document simply as entries), share them with other users, modify their contents, and delete them. The software is event-driven; procedures should only be run when something of interest happens (e.g. a user creates a new entry, or edits or deletes an existing entry). Since the journaling software is data-driven, sequence diagrams will be used for describing the system behavior. **Note: The student who was assigned to create the behavioral diagrams did not submit their work to the rest of the group by the deadline.**

1.2 Definitions, Acronyms and Abbreviations

The following terms will be necessary for understanding the diagrams in this report:

- **Administrator (Admin)** – A special class of user which can perform actions such as manually creating or deleting users and modifying any data within the system/database.
- **Client** – The interface and underlying code that the user interacts with. For security purposes, the client must not have unlimited access to the server and database, and may only make requests to the server/database that can then be approved or denied.
- **Database** – The collection of information which the server keeps and distributes or modifies as necessary. Only the server has direct access to the information in the database. Information in the database may include user login information and entries.
- **Editor** – An interface displayed to the client which allows the user to modify the contents of an entry. When the user is finished modifying an entry, they may submit the new contents of the editor to the server, to set that entry's contents.
- **Entry** – Text written by a user or users, with an associated date of creation.
- **ID** – An identification number of string which is unique to a single object (such as a user or entry). Using IDs allows the system to keep track of different objects and provide the correct ones to the client.
- **Server** – Code running in a secure location, separated from the client or user. The server has unlimited access to information in the database, but acts on requests from the client to make modifications, ensuring that the user making the requests has the correct permissions first.

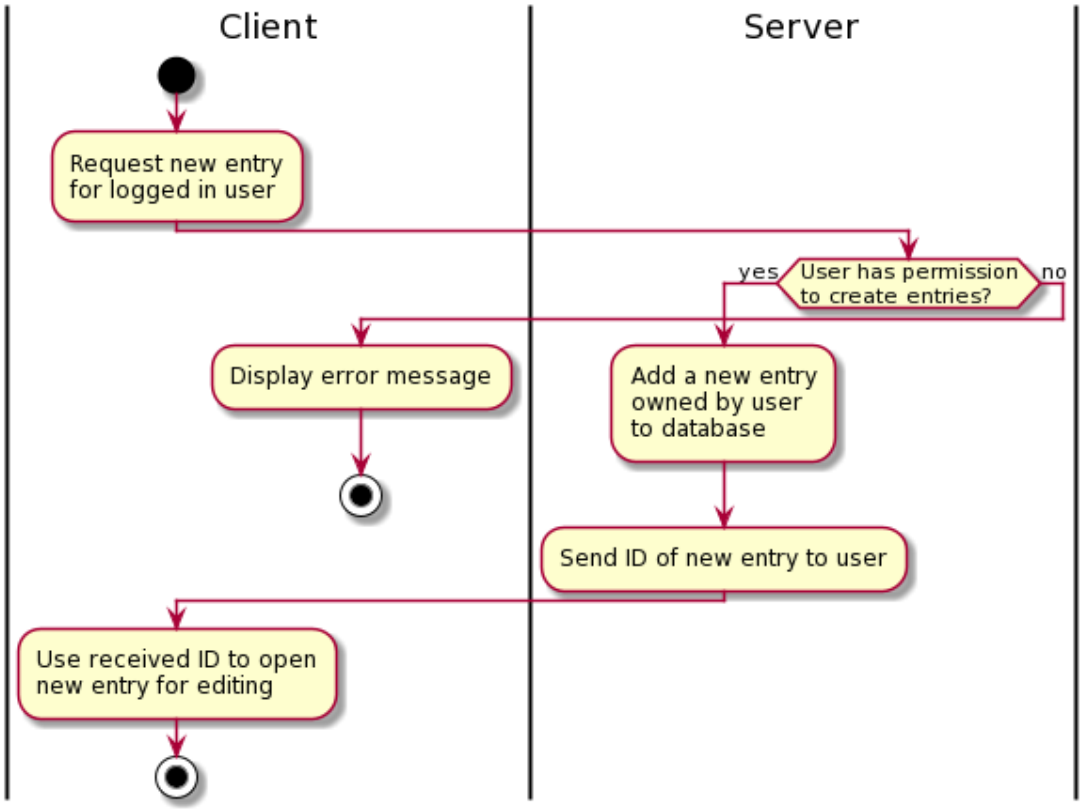
1.3 References and Acknowledgments

[1] "Accounts," Meteor API Docs. [Online]. Available: https://docs.meteor.com/api/accounts.html. [Accessed: 07-Nov-2020].

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

2 Activity Diagram(s)

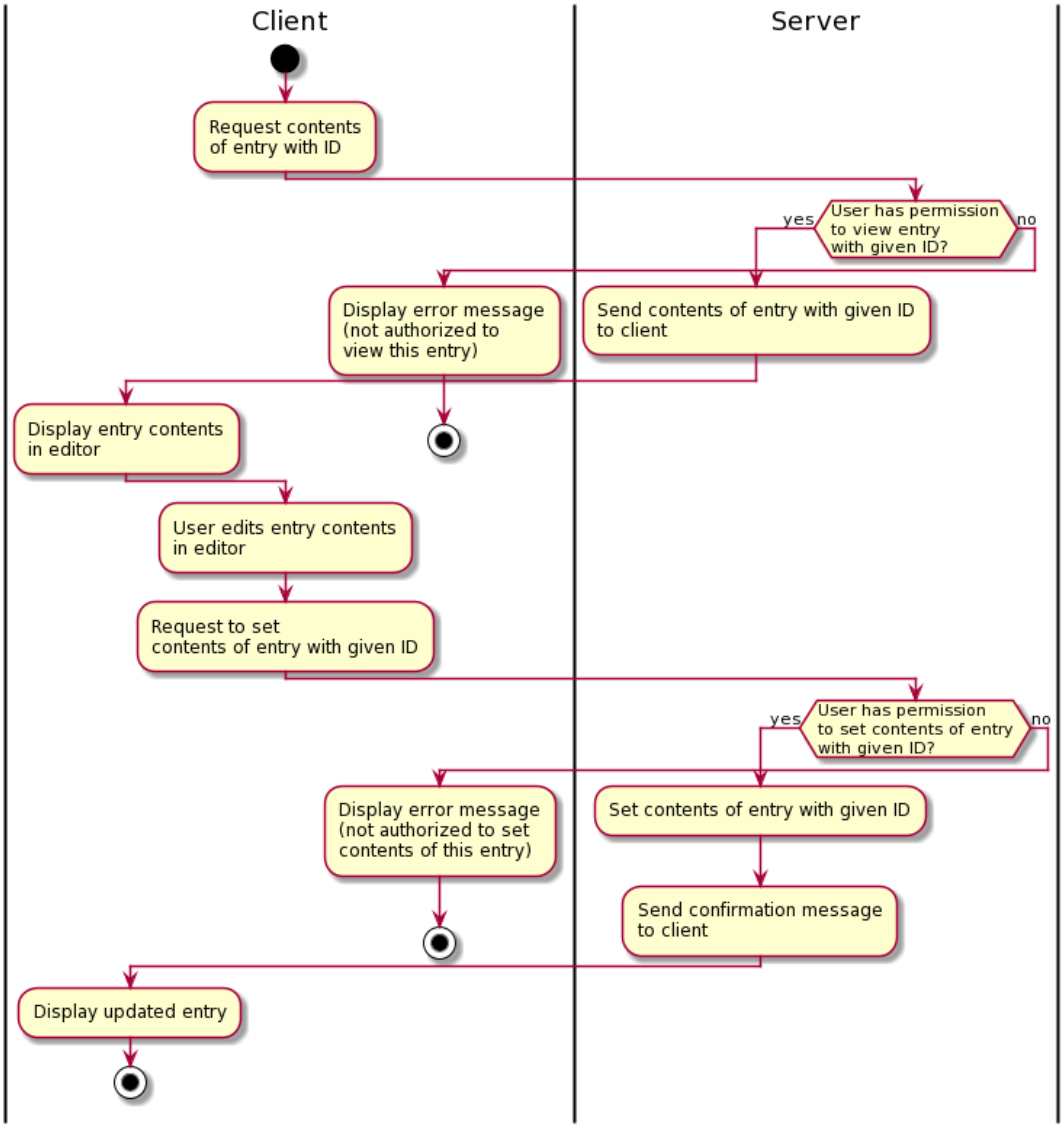
2.1 Creating an entry



The above diagram depicts the process for a user creating a new entry. Within a user's permissions, there may be a field describing whether or not that user is allowed to create and own entries. Before creating a new entry for the user, the server must check whether or not that user has this permission enabled. If not, the server can respond to the client's request with an error message, which the client will display; if the user does have the permission, the server can create the new entry, assign it to the user who requested its creation, and send information about it to the client so that the user can edit it.

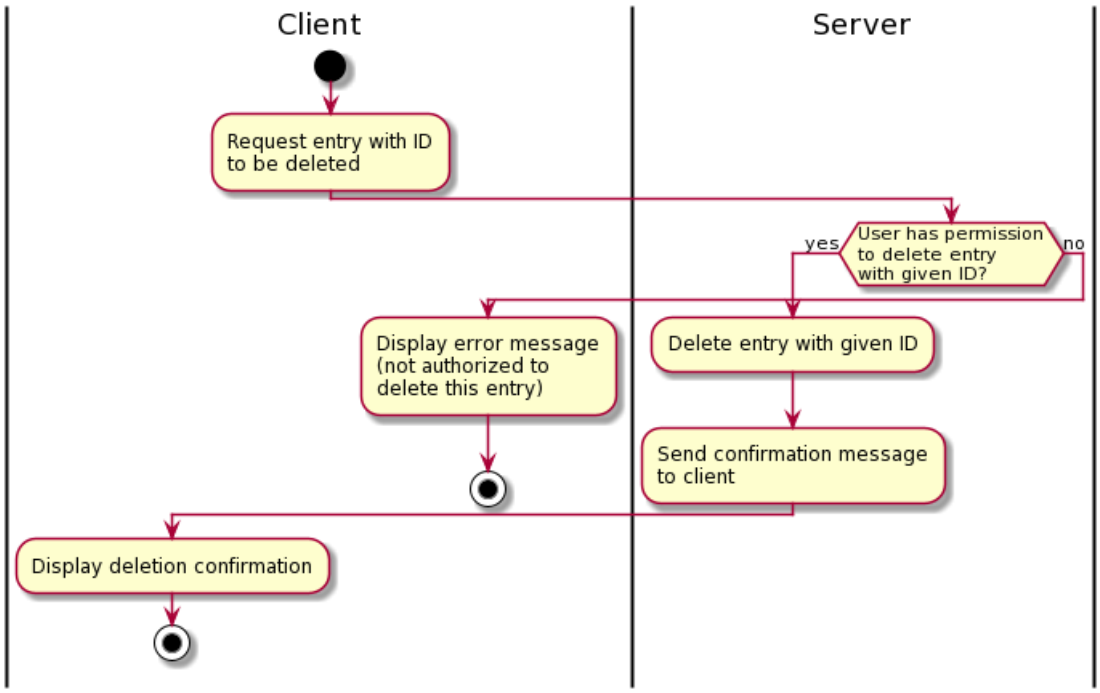
Note that it is very important that the permission verification process take place on the server, not the client. If the verification check took place on the client, the user could theoretically modify the code running on their client to bypass the check, allowing them to perform actions outside of what they have permissions for.

2.2 Editing an entry



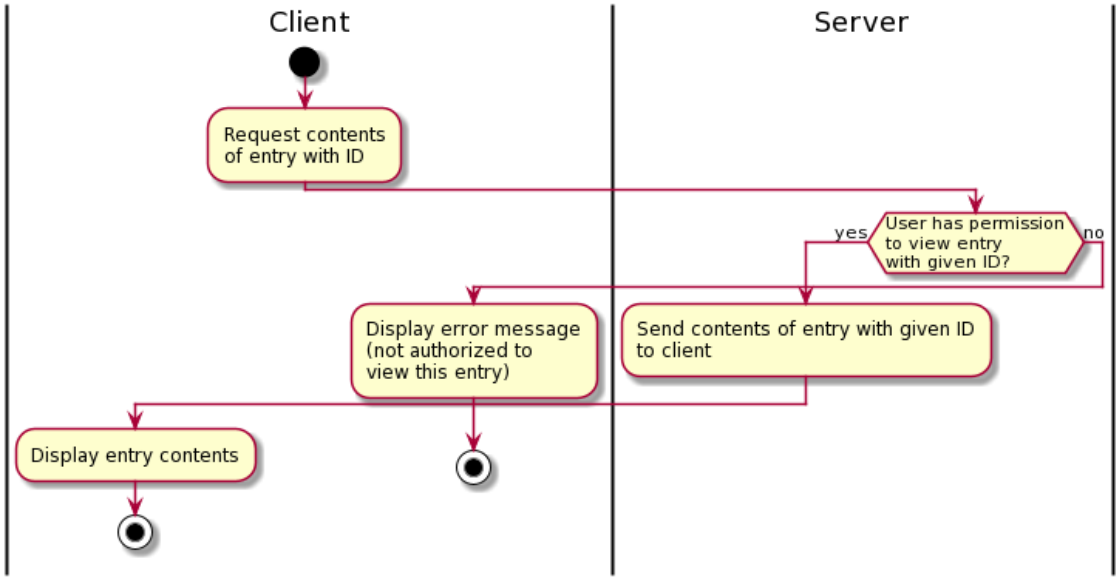
The above diagram depicts the process for a user modifying an existing entry. The general flow follows the same pattern as the other processes, but is more involved, since the server must perform two permission verification checks. Before the editor interface is displayed to the user, the server must verify that the user has permission to view the entry. Once the user submits their changes to the entry, the server must verify that the user still has permission to edit the entry (in case the owner of the entry removed their permissions while they were making changes).

2.3 Deleting an entry



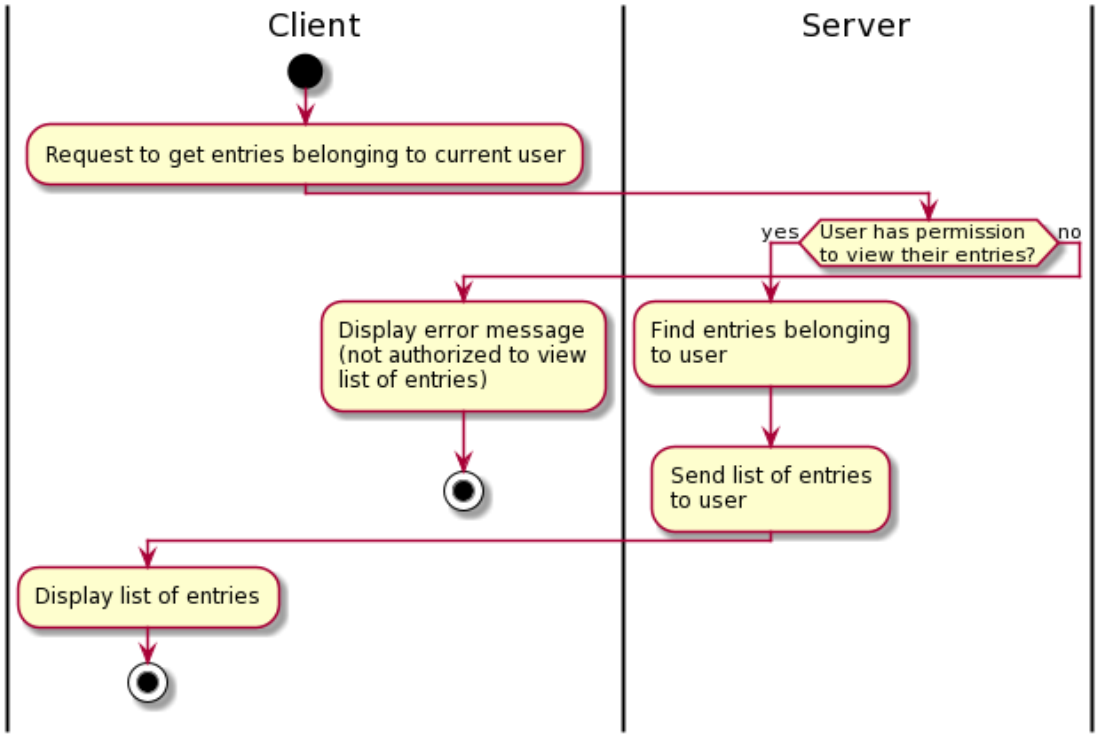
The procedure for deleting an entry, depicted in the diagram above, follows a very similar process to that for creating an entry in diagram 2.1. When the client makes a request to delete an entry with a given ID, the server checks whether the client's user is authorized to delete that entry. If they are authorized, the server deletes the entry and sends a confirmation message to the client and user. If they are not authorized, the server sends an error message to the client, which the client then displays to the user.

2.4 Viewing an entry



The procedure for viewing an entry follows that for creating entries (2.1). In this case, the server will check whether or not the requesting user has permission to view the requested entry before providing its contents to the client.

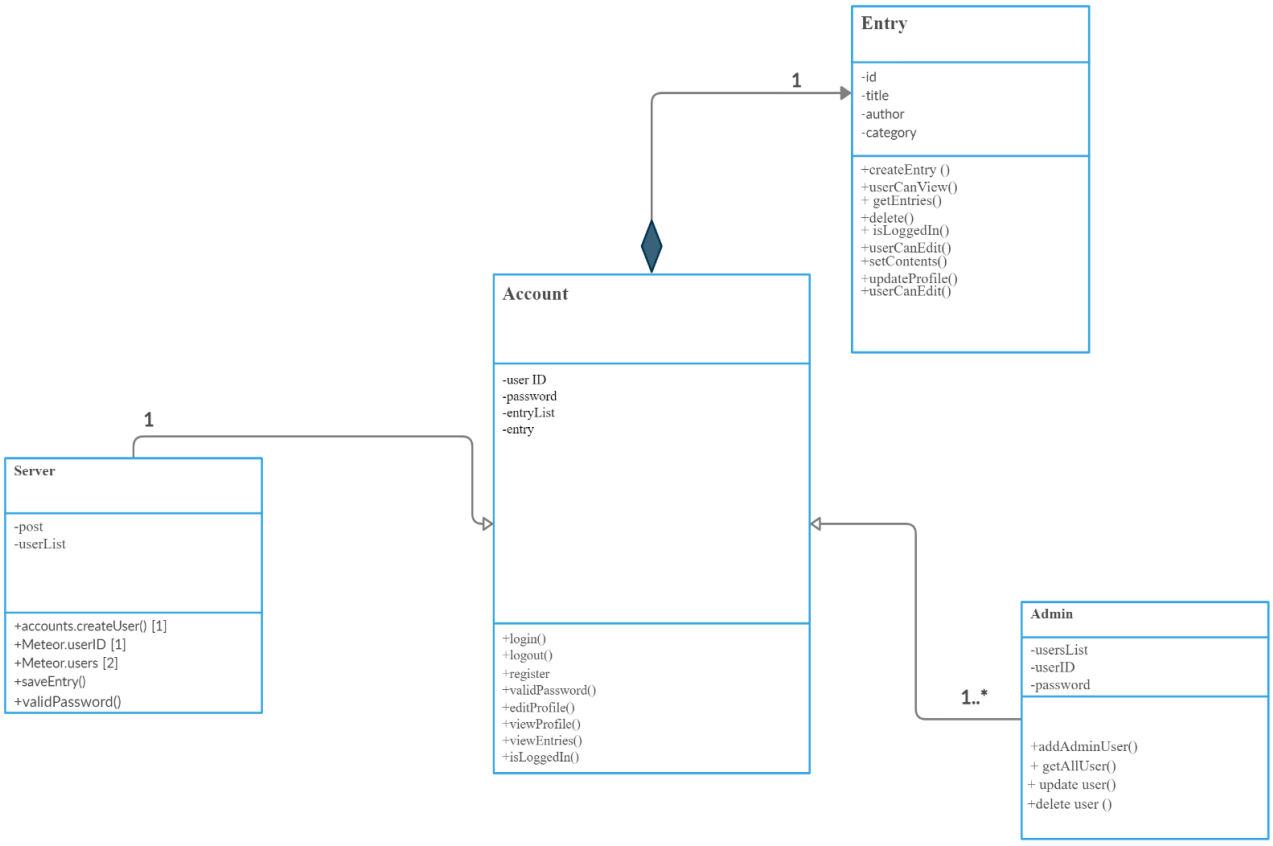
2.5 Viewing a list of entries



The procedure for viewing a list of entries again closely resembles that for creating entries (2.1). If users must have permission to view their entries, the server must first confirm that the given user has this permission before retrieving the information. Once it has confirmed that the given user may view their entries, it must request a list of the entries from the database, then send that list to the client so that the user can view it. If the user does not have this permission, it must send an error message for the client to display instead.

3 Class Diagram(s)

3.1 Structural View



There are four classes: Account, Entry, Admin, and Server. Account will be used for the user to login, logout, signup, and to make changes to their profile. Entry will be used to create new entries, delete entries, and edit entries. Admin class will be for creating admin accounts and managing users. Server will be used for logging in, retrieving user info, and saving entries.

4 Behavioral Diagram(s)

The student who was assigned to create the behavioral diagrams and write this section did not submit their work to the rest of the group by the deadline.

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 with other members
27 October 2020	• Quick check-in with other members

30 October 2020	• Quick check-in with other members
	• 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 with other members to get development environment set up
10 November 2020	• Quick check-in with other members to get development environment set up
11 November 2020	• Quick check-in with other members to get development environment set up
12 November 2020	• Quick check-in with other members to get development environment set up
13 November 2020	• Quick check-in with other members to schedule formal meeting and get environment set up
15 November 2020, 16 November 2020	• Workload for Milestone 2 distributed amongst team members
17 November 2020	• Activity diagrams created and pushed to repository, began filling out template
18 November 2020	• Finished writing activity diagram descriptions and some parts of introduction
19 November 2020	• Quick check-in with other members to discuss project and Milestone 2
21 November 2020	• Class diagram with description added to document