

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: | 20 November 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 Activity Diagram(s) 2

2.1 D-1 2

2.2 D-2 2

2.2 D-3 2

3 Class Diagram(s) 3

3.1 D-1 3

3.2 D-2 3

3.2 D-3 3

4 Behavioral Diagram(s) 4

4.1 D-1 4

4.2 D-2 4

4.3 D-3 4

Appendix A - Group Log 5

< A table of contents

Edit this table as needed to suit the page numbers and titles of the diagrams you included. >

# Introduction

## Project Overview

< A brief description of the project and the diagrams.

TO DO: Write 1 paragraph explaining the project and anything of unique relevance for the diagrams included in this document. At minimum this should include which type of behavioral diagram your system will use.>

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).

**WHAT TYPE OF BEHAVIORAL DIAGRAM TO USE?**

## Definitions, Acronyms and Abbreviations

<Define all the terms necessary to properly interpret the report, including acronyms and abbreviations.

TO DO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order.>

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

* **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.

## 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].

# Activity Diagram(s)

## Diagram Description automatically generatedCreating an entry

<Provide a title, the activity diagram, and a description of it’s contents.

TO DO: Provide an activity diagram, including which use case it deals with, and give a short description of its contents. What is happening, what the user is doing, what the start and end state/goals are. Give the diagram an appropriate title instead of “D-1”>

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.

## Diagram Description automatically generatedEditing 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).

## Diagram Description automatically generatedDeleting 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.

## Diagram Description automatically generatedViewing 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.

## Diagram Description automatically generatedViewing 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.

# Class Diagram(s)

## Structural View

## Diagram, schematic Description automatically generated

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.

# Behavioral Diagram(s)

## <*This section will have either sequence diagrams or state diagrams. This will depend on if you system is data-driven (sequence) or event-driven (state).>*

## D-1

<Provide a title, the behavioral diagram, and a brief description about it.

TODO: Provide a Behavioral Diagram (sequence or state) and then give a description of what activity it is describing visually. For sequence diagrams, this would be each actor involved and the function calls between them. For state diagrams, this would be a table of each state with each state transition also labeled and described. A series of related complex states should be simplified as a “superstate” with a more in-depth view of it shown in a separate diagram. See the “operation” state in the microwave state diagram from Lecture 13 slides 27 and 28 for an example of this.>

## D-2

<Same as for D-1, repeat for as many diagrams as you have for this section. Remove if not needed.

TODO: Repeat same process as you did for D-1: Title, Diagram, Description.>

## D-3

<Same as for D-1, repeat for as many diagrams as you have for this section. Remove if not needed.

TODO: Repeat same process as you did for D-1: Title, Diagram, Description.>

Appendix A - Group Log

< Please include here all the minutes from your group meetings, your group activities, and any other relevant information. This should contain and continue the contents from the SRS. This is optional for one-person projects.>