**myJournal: Mental Health Tracker**  
  
System Architecture Specification (SAS)

*Team 5*

| *Matt Elpus* | Product Owner |
| --- | --- |
| *Daniel Guthart* | Scrum Master |
| *Jade Tustin* | Development Team |

Advisor: Vanessa Aguiar

Submitted in partial fulfillment

of the requirements of CSC 431

Software Engineering course project

April 5, 2024

# Version History

| Version | Date | Author(s) | Change Comments |
| --- | --- | --- | --- |
| 1.0 | 4/2/2024 | Matt, Daniel, and Jade | Updated document. |
| 1.1 | 4/26/2024 | Jade | Added the most recent sequence and class diagrams from the final presentation. Applied fixes detailed in feedback from professor/TAs, less the suggestions for implementing design patterns. |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[**1. System Analysis 4**](#_heading=h.3dy6vkm)

[1.1. System Overview 4](#_heading=h.1t3h5sf)

[1.2. System Diagram 5](#_heading=h.4d34og8)

[1.3. Actor Identification 5](#_heading=h.2s8eyo1)

[1.4. Design Rationale 5](#_heading=h.17dp8vu)

[1.4.1. Architectural Style 5](#_heading=h.3rdcrjn)

[1.4.2. Design Pattern(s) 6](#_heading=h.26in1rg)

[1.4.3. Framework 6](#_heading=h.lnxbz9)

[**2. Functional Design 7**](#_heading=h.35nkun2)

[2.1. myJournal Login 7](#_heading=h.1ksv4uv)

[2.2. myJournal Register 7](#_heading=h.5b4r0rg3aya)

[2.3. myJournal Add Entry 8](#_heading=h.uwkfrafrukjs)

[2.4. myJournal Edit Entry 8](#_heading=h.arulr2tk9fhk)

[2.5. myJournal Delete Entry 9](#_heading=h.nha685ettnpg)

[**3. Structural Design 10**](#_heading=h.44sinio)

### System Analysis

#### System Overview

The myJournal application will provide its users with an easy and convenient way to track their mental health on their own. In order to deliver on this aim, it is necessary to deploy the appropriate underlying system.

The system will primarily be responsible for managing user accounts and processing data pertaining to user requests. Per the current iteration of the System Requirements Specification, the capability to log in, register, view the dashboard, and create or modify journal entries must be present. This will be carried out using three classes, which each correspond to a layer of the architecture. The layering of this architecture is detailed herein (and in further detail in [section 1.4.1](#_heading=h.3rdcrjn)), and the details of the corresponding classes are laid out in [section 3](#_heading=h.44sinio).

User interactions with the myJournal system will generally adhere to the following format:

1. The user submits some request or call to the system by using the interface provided to them on the platform of their choice (mobile application on Android/iOS or web application via browser).
2. The appropriate module will employ the proper internal logic and mechanisms to handle the request, calling other modules or the API below it as necessary.
3. Any necessary changes to the database are performed.
4. For many calls, the necessary data will be returned and displayed to the user.

The system will adhere to a three-tier architecture style, which is specified in more detail in [section 1.4.1](#_heading=h.3rdcrjn).

#### System Diagram

#### Actor Identification

There are two main human actors who may interact with the myJournal system: non-registered users, and registered users. Non-registered users are essentially ‘locked out’ of the main system and are only given the capability to register an account. Registered users are permitted to log into the main system and access user-facing functionalities (i.e. create/modify journal entries, view dashboard) specified in more detail within the System Requirements Specification.

The system must also incorporate the ability for administrative actors to interact with the system in order to perform maintenance and continue to deliver the desired capabilities to the end-user.

#### Design Rationale

##### Architectural Style

As stated in section 1.1, this system will utilize a three-tier architecture. This structuring will employ elements of both database-centric and client-server architectures, with the addition of an intermediary layer to handle user requests and conform to privacy and security protocols while doing so. The three tiers of this architecture are as follows:

* *Presentation Tier* - This tier would encompass the user interface, whether on the web browser or mobile app. The specific characteristics of this tier will vary in implementation depending on the platform (iOS, Android, or web page).
* *Application Logic Tier* - This tier would include the services for managing user registration, login, journal entries, and data encryption. When a user performs some action within the presentation tier, execution of that action is handed over to this tier for processing. This tier will utilize calls to the API and database, as necessary, to process user actions and deliver information.
* *Data/API Tier* - This tier handles both calls to the Django API used to unify the above two tiers, in addition to any appropriate protocols or APIs for working with the databases where user data and journal entries are stored.

Employing an architecture with this structure should ensure consistency across user experiences, promote software and architecture stability, and enhance the development team’s ability to expand the functionality of and maintain the software in the long run.

##### Design Pattern(s)

In order to streamline the development process and ensure the greatest degree of compatibility between the various platforms and the application logic, some common software design patterns will be deployed. Some examples of these patterns include:

* Factory Pattern - This design pattern simplifies the creation of new objects. This will be used for creating new user accounts and journal entries within the system.
* Singleton Pattern - This methodology restricts an application to the creation of only one instance of a given class. This will be deployed for components that should only have a single instance, such as the database connection pool.

During development, it may become clear that other design patterns may be helpful in the development and maintenance of the myJournal system. Should this be the case, this document will be updated with the appropriate information pertaining to such design patterns.

##### Framework

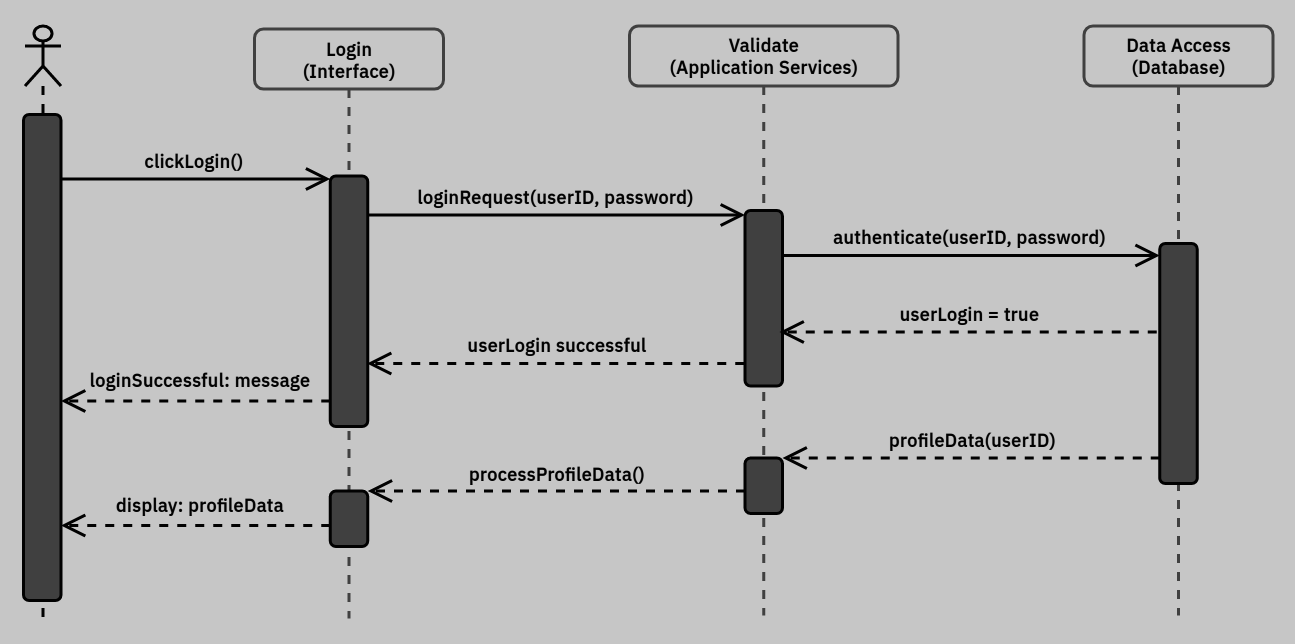
As stated throughout this document and within the System Requirements Specification, this system will deploy the Django framework for web development. This framework is based on the Python programming language and provides a straightforward API for developing both the web and mobile applications, helping to unify the user experience across platforms.

To learn more about the Django framework, click [here](https://www.djangoproject.com/).

### Functional Design

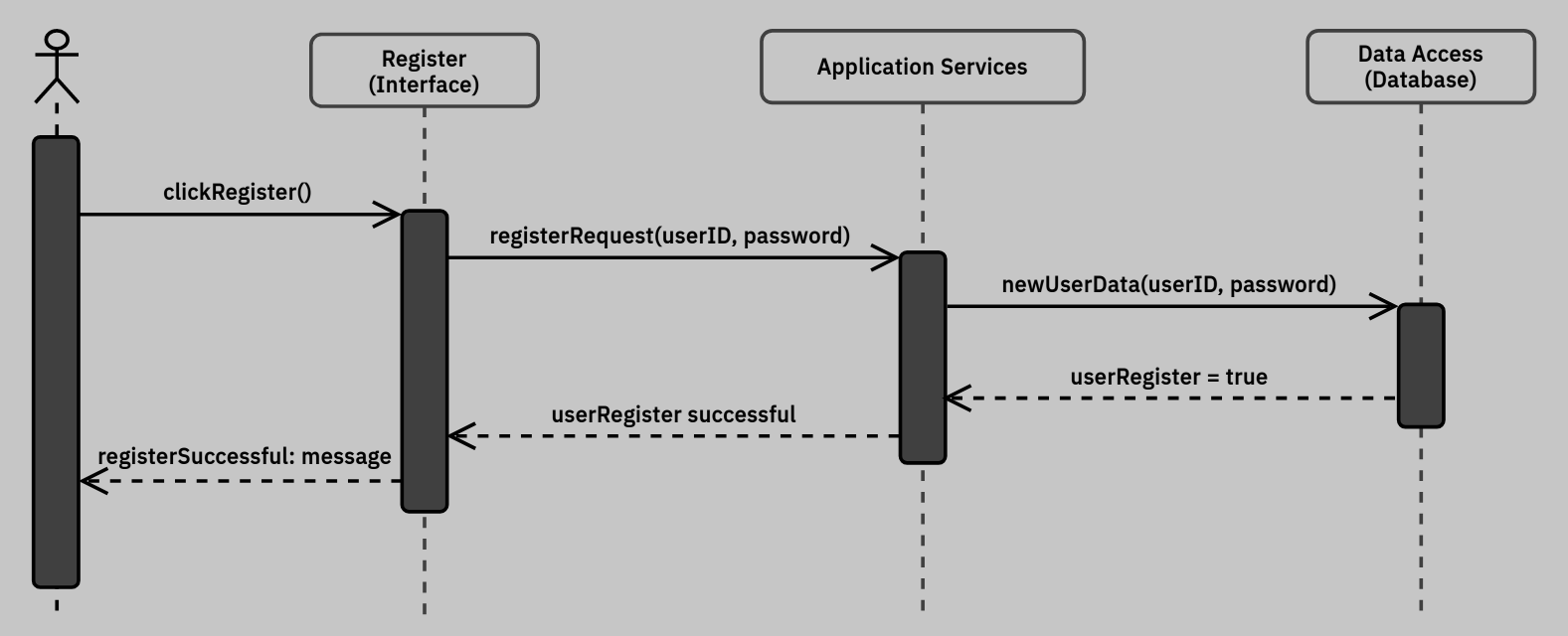
All diagrams presented in this section are UML sequence diagrams reflecting how users interact with the software and how the underlying classes will perform desired actions. All functions and classes detailed herein are provided in [section 3](#_heading=h.44sinio).

#### myJournal Login



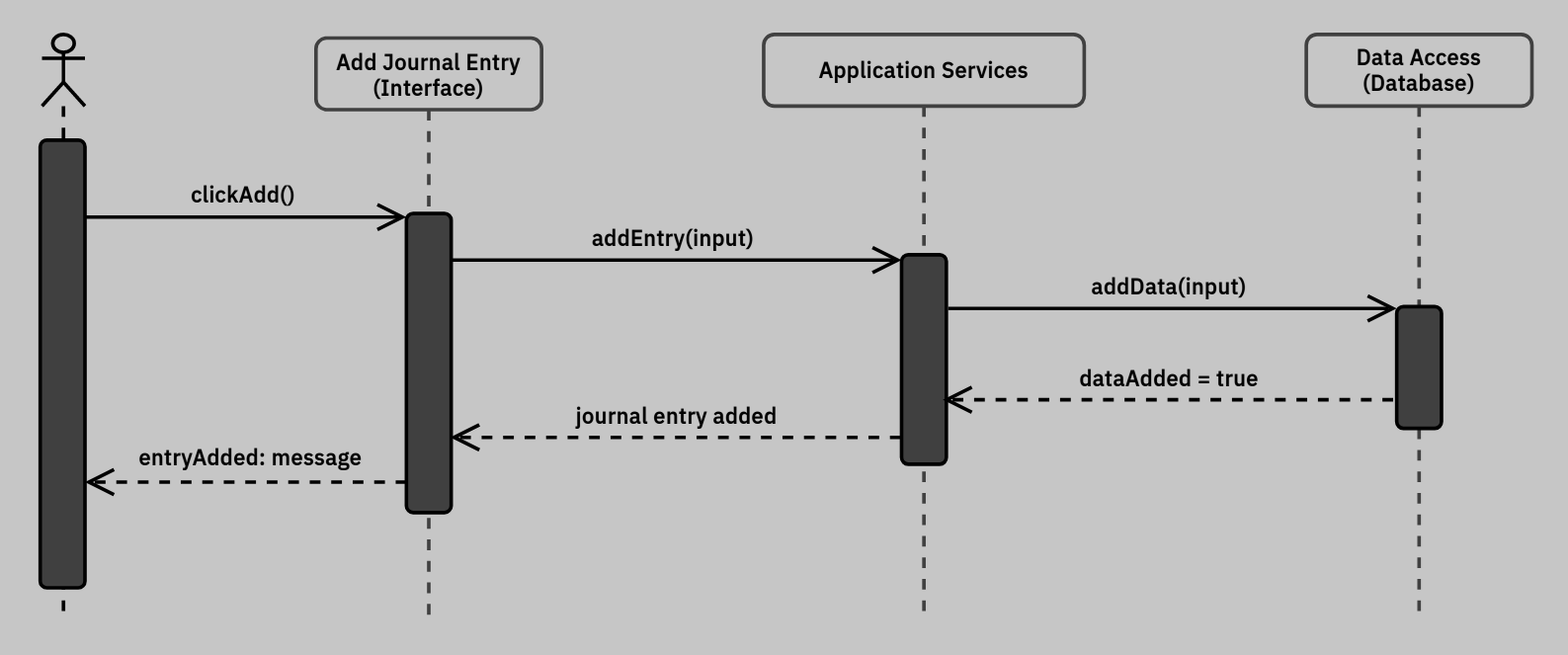
Upon entering the myJournal application, users will be prompted to log in or register. Users who are already registered may log in by clicking on the login button and entering their email and password into the appropriate fields. This submits a login request to the application services layer, which then passes an authentication request to the data access layer. When authentication is complete, the user will receive a message that their login was successful, and their profile data will be loaded.

#### myJournal Register



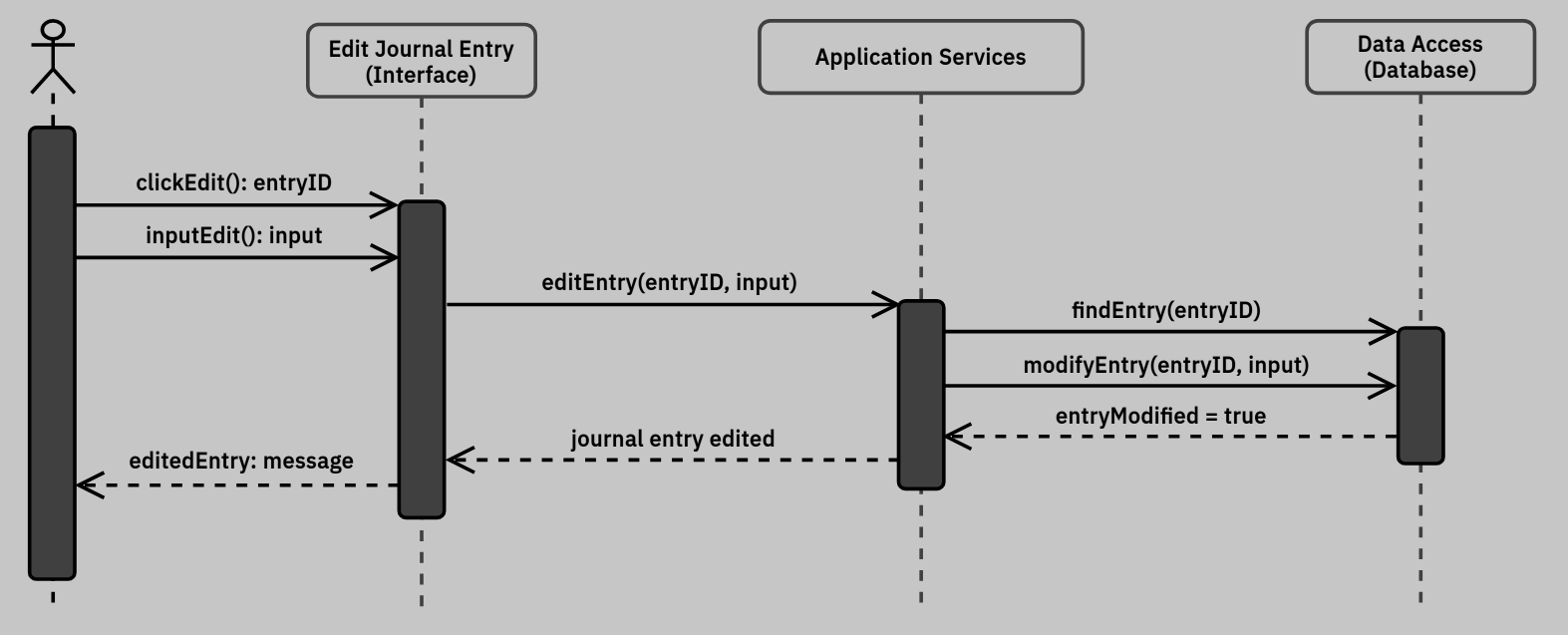
If a user enters the myJournal application for the first time or has not registered yet, they will need to register before accessing any of myJournal’s features. The user will click on the register button presented when the application is opened, and will enter a valid email and password into the displayed fields. When the user has done so, they will click a button that submits a registration request to the application services layer with the user’s entered information. Once this information is validated (that is, it is confirmed that the password meets non-functional requirements and that the email has not been used to register another account), the user will receive a message stating that their registration was successfully processed.

#### myJournal Add Entry



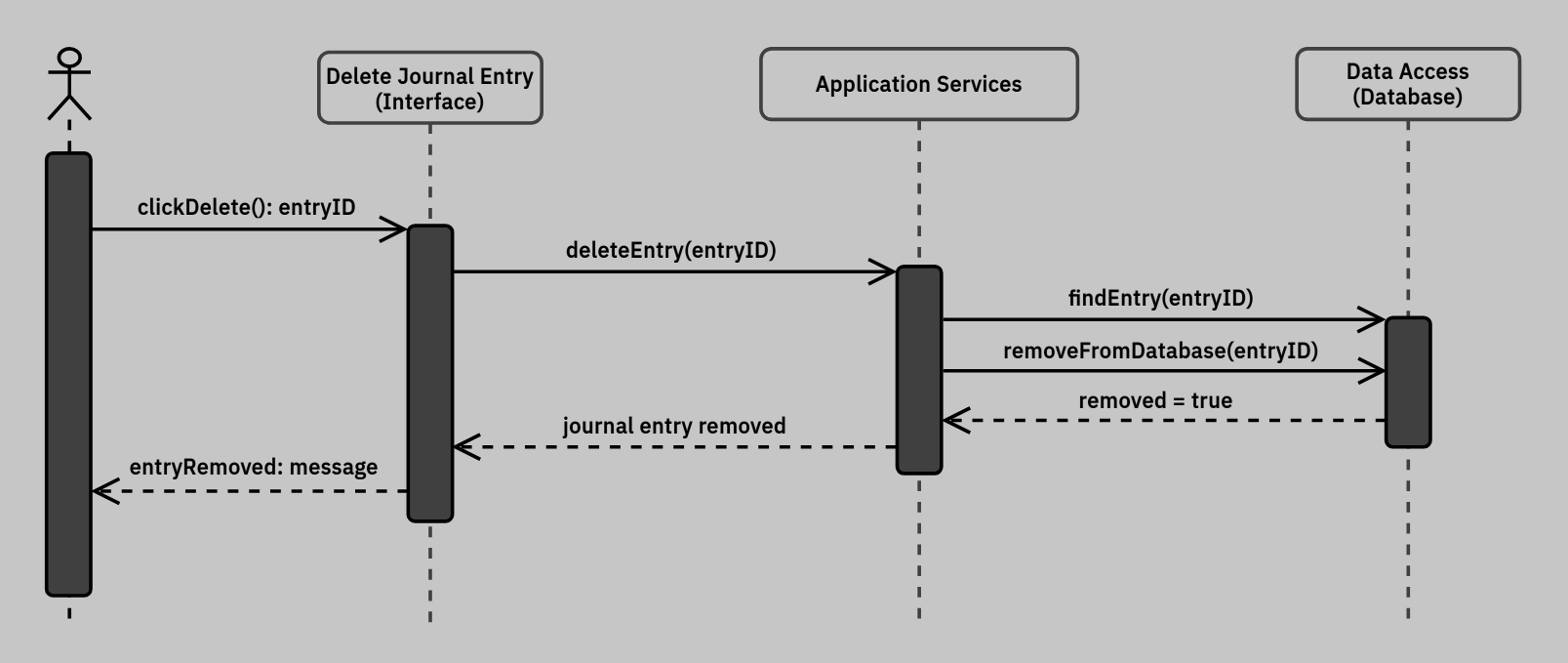
Once the user has logged into the myJournal system, they can add, edit, or delete journal entries from their dashboard. If a user wants to add a journal entry, they can do so by clicking on the relevant icon on the dashboard or by clicking on the appropriate option in the user menu (shown in the mockup). Once the option has been selected, users will enter whatever information they desire into their journal entry and click the appropriate button to add the entry. This submits an entry request to the application services layer, which in turn creates the relevant data entry into the database. When the system processes the user’s request, the user will receive some indication that their attempt to add the journal entry was successful.

#### myJournal Edit Entry



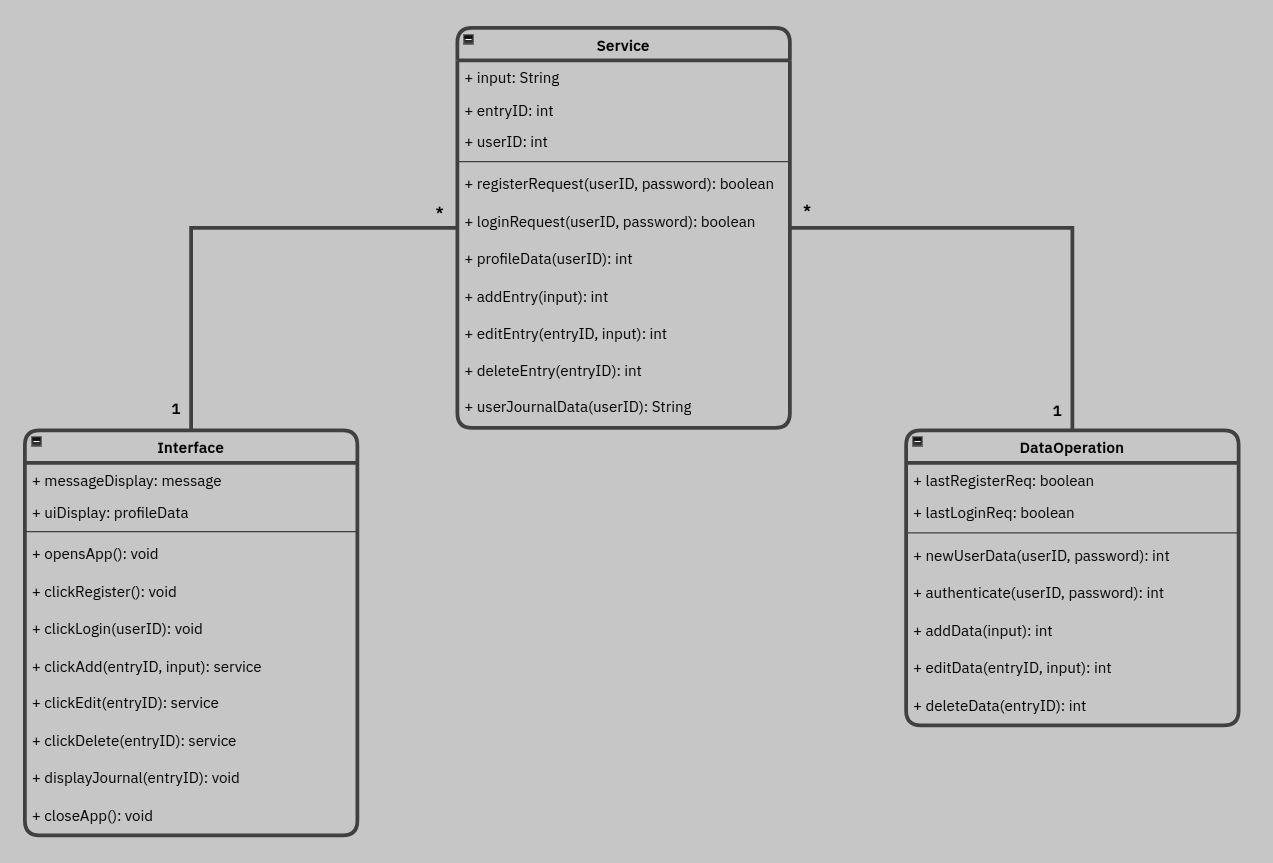
If a user wants to edit a journal entry, they can do so by clicking on the relevant icon on the dashboard or by clicking on the appropriate option in the user menu (shown in the mockup). Once the option has been selected, users will select the entry they’d like to edit and perform any desired changes. When done, the user will click the necessary button to update the entry. This submits an edit request to the application services layer, which in turn modifies the appropriate data entry into the database. Once the system processes the user’s request, the user will receive some indication that their attempt to edit the journal entry was successful.

#### myJournal Delete Entry



If a user wants to delete a journal entry, they can do so using the same prompt as is presented for editing journal entries (shown in the mockup). Once the option has been selected, users will select the entry they’d like to delete and press a button to confirm the action. This submits a delete request to the application services layer, which in turn removes the appropriate data entry from the database. Once the system processes the user’s request, the user will receive some indication that their attempt to delete the journal entry was successful.

### Structural Design

**