

Software Requirements Specification

for

Web Based Personal Journal (JournalJay)

Version <1.1>

Prepared by

Group Name: Sizzle Snap

|  |  |  |
| --- | --- | --- |
| Laurel Anderson | 11638131 | laurel.o.anderson@wsu.edu |
| Irina Bejan | 11613022 | irina.bejan@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | 11/6/2020 |
|  |  |
|  |  |

Table of Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behavior Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| First Draft Version 1.1 | Laurel Anderson  Irina Bejan | The first draft has the team’s first attempt at each section. All parts are complete. Waiting for feedback. | 11/06/20 |

# *<In this template you will find text bounded by the “<>” symbols. This text appears in italics and is intended to provide explanations and guide you through the document. There are two types of comments in this document. The comments that are in black are intended specifically for the course. The comments that are in blue are more general and apply to any SRS. Please make sure to delete all the comments before submitting the document.>*

# Introduction

## Document Purpose

The product described in this document is the web based personal journal site JournalJay version <1.1>. The purpose of this document is to describe the scope of the JournalJay to users, system administrators and the development team. Aspects of the system will be described in this document including but not limited to, user interaction, system constraints, and interfaces.

## Product Scope

JournalJay is web-based journaling application using HTML/CSS and JavaScript. The web journal is a convenient alternative to traditional journaling for users that have electronic devices. The web journal has a unique sign in for each user. Each user can have journal entries up 365 entries that they can review at any time while they are signed in. The website also has a mood tracker. For each user entry, they can assign a mood for the day from a list of presets on the site. A user will also be able to change the journal entry background from a set of presets. Usernames, passwords, and journal entries are stored on a simple database. The software needs an internet connection to work.

## Intended Audience and Document Overview

This document is intended for Professor Brandon Hedden of CS 320, the developers, and JournalJay users. The Professor will look for the completeness of the entire document. The developers are Laurel Anderson and Irina Bejan; they will use this document to reference the project outline and requirements. The user can use this document to look up definitions used for the software (section 1.4) and the product functionality (section 2.2). This document is organized with the introduction first. The introduction includes the document specific requirements. The overall description section and the specific requirements section describes the software in its entirety. Section 4 describes the non-functional requirements.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Admin/  Administrator | Someone who is given specific permissions to manage and control the system. |
| App | Application |
| bcrypt | An open source hashing algorithm that is resistant to brute forcing. |
| CS | Computer science |
| CSS | (Cascading Style Sheets) A programming language used to style HTML. |
| HTML | (Hypertext Markup Language) The standard markup language for documents designed to be displayed in a web browser. |
| HTTP | (Hypertext Transfer Protocol) The data transfer protocol used on the World Wide Web. |
| IEEE | Institute of Electrical and Electronics Engineers. |
| JavaScript | High level programming language used with HTML and CSS on the Internet. |
| User | Someone who interacts with the web application. |

## Document Conventions

This document follows the IEEE formatting requirements. This includes size 11 Arial font for text, single spaces, and 1” margins. Italics are used only for comments. Sections are labeled by size 18 bolded text and centered on a dark background. Subtitles are labeled by size 14 bolded text, justified left.

## References and Acknowledgments

We would like to thank God for helping us not lose our minds during this semester.

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

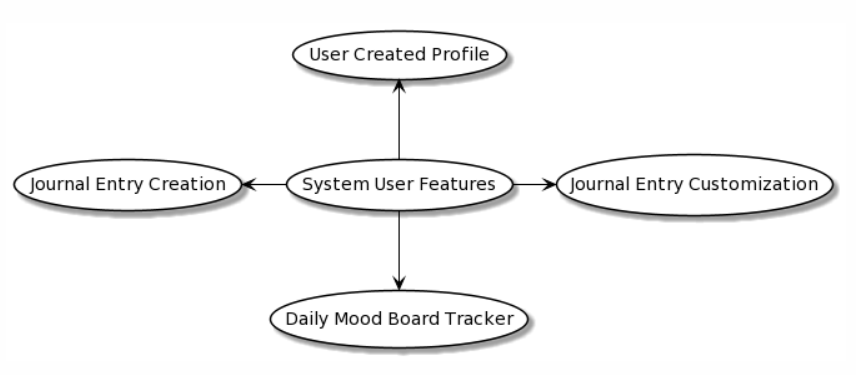
Practice for Software Requirements Specifications”, October 20, 1998.

# Overall Description

## Product Perspective

JournalJay is an online application that allows users to create personalized journal entries in an easy way. When the user goes to the website, they will be prompted to sign in or create username and password. Once signed in, the user will be able to create one journal entry a day. The user will also be able customize their journal entries and pick a mood for the day from a list of presets while they are creating their entry. The user will be able to revisit entries later.

The user will also be able to get help regarding the site from a help button on each screen. The help button will show the user a quick tutorial that describes the different functions of the site. JournalJay is a self-contained product, a digital conversion of a traditional medium. The usernames, passwords, and journal entries will be stored on a simple database client side.

A simple diagram showing the project perspective is as follows:

## Product Functionality

Here is some of the functionality of JournalJay:

* The user can create a username and password for their own account.
* The user can sign into their account.
* The user can create a journal entry every day.
* The user can customize the journal entry while they are creating it.
* The user can review any previously created journal entry while signed in.

## Users and Characteristics

There are two types of users that interact with the system, general users and system administrators. Each of these users will interact with the system in a different way so each of them has their own requirements. General users include first time users and returning users. New users will need to create a username and password when they first register. After the initial interaction with JournalJay, new users will not have to create a username and password again.

The general users will most likely have mid-level to advanced technical experience because they use an online journal instead of a traditional journal medium. Avid users can sign in once a day to put their thoughts down in an entry. They may come back in the same day to review entries and moods from other days. They also may decide to use the entry customization in its entirety. Secondary users may only use the site to make an entry 3 to 5 times a week. They would not use the mood tracker or customization options. They may, however, sign in to review their previous journal entries.

The system administrators can access the internal logic of the site and make any changes they see fit. Administrators can also see all the current users, passwords, and journal entries stored on the database.

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist. In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface

TO DO: As stated above, in at least one paragraph, describe the environment your system will have to operate in. Make sure to include the minimum platform requirements for your system. >

On the server side, the components of the application must function within an Ubuntu Linux operating system environment. On the client side, the components of the application must function within common web-browser environments. These browsers will minimally include:

* Apple Safari 11+
* Google Chrome 86+
* Microsoft Internet Explorer 11+
* Mozilla Firefox 84+

## Design and Implementation Constraints

Due to the lack of database experience of the developers of this project, the journal site application will use a basic database to manage the usernames, passwords, and journal entries. The database will rely on the user’s browser having JavaScript functionality. The database will also reside locally on the user’s computer. The application uses a model view control system to organize the code. The languages used for the application are JavaScript and HTML/CSS. Additional libraries will include underscore.js and bcrypt.

## User Documentation

Because the system offers basic functionality, the user documentation component is a Help menu in a designated spot on the web application. Upon clicking the button, the user will get the option of viewing a tutorial with Jay, the Journaling Blue Jay, or submitting an on-line help request. The tutorial will be a separate page with screenshots of each button and what it does, or a screen-recording of a simulated user interacting with the system with a voice-over narrating the functionality. To submit a ‘help’ request, the user will simply be instructed to email a specified help account.

## Assumptions and Dependencies

The assumptions and dependencies include:

* The browser the consumer uses has the necessary JavaScript libraries we are using.
* If we decide to encrypt the usernames and password, we will (probably) use the bcrypt libraries for JavaScript. The browser should support bcrypt.
* The browser should be able to handle the underscore.js library.
* The user can have up to 365 journal entries.

# Specific Requirements

## External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., Cancel) that will appear on every screen, error message display standards, and so on. Define the software components for which a user interface is needed.

TO DO: The least you can do for this section is to describe in words the different User Interfaces and the different screens that will be available to the user. Optional: You may also provide an initial Graphical User Interface design (does not have to be final).>

The user interface of JournalJay is designed to be minimalistic with light color schemes and basic styling to encourage a clean and de-stressed state of mind. A user new to the web application will first see the Log-in Page. To register, they will click a marked button to go to the Registration page. When signed in, a user will see the Landing Page, where they can choose to create a new journal entry or view an old one. The “A” icon is the button the user presses to access their account page, and the “?” icon is the button the user presses to access the help menu.

Every user’s account page will display their username, and password, and they will have the option to change these fields for their own account.

The New Entry page is the space for creating a new journal entry. It includes title, date, body text, and mood-tracker sections. This allows the user to name their journal entry and begin typing the content of the entry. The date will be generated by the web app; it will not be edit-able. The mood-tracker will be a gallery of emoticons portraying different moods. The user can select one for each journal entry. There will be a “Save” button for the user to save their work.

The Previous Entries page will display each past entry as an icon. The icon will include the title given by the user, a picture of their mood if they selected one, and the date created. Upon pressing a journal entry, the user will be taken to the view-only journal entry page.

### Hardware Interfaces

The user will have a computer with a keyboard, mouse, and monitor. The keyboard will be used to input journal entries, usernames, and passwords. The mouse will be used to validate any fields such as accepting a username and password. The mouse will also be used to choose from any options given to the user such as choosing a mood for the day. The monitor will show the site, any messages from the site to the user, and the journal entries. JournalJay has no effect on the RAM or other hardware of the user system.

### Software Interfaces

The software interface integrates smoothly with the local Linux operating system. The interface will be able to manage the local input and output devices as needed. The application communicates with the database in order to get necessary information about each user. The communication between the database and the application consists of both reading and modifying the data.

### Communications Interfaces

JournalJay will be web based so the browser will handle encryption (HTTPS). JournalJay will interact with the user though an internet browser, but the database used will be client side on their computer. We may decide to encrypt the passwords when they are stored on the client computer. If we encrypt the passwords, we will consider using bcrypt.

## Functional Requirements

Security Requirements:

* Each user must have a corresponding username and password.
* The username and password may be stored locally on the user’s computer.
* The usernames and passwords should be encrypted using bcrypt.

Interface Requirements:

* The background of the site must be a light color, either white or light blue.
* The typeface of group text must be Karla.
* Any group text must be black.

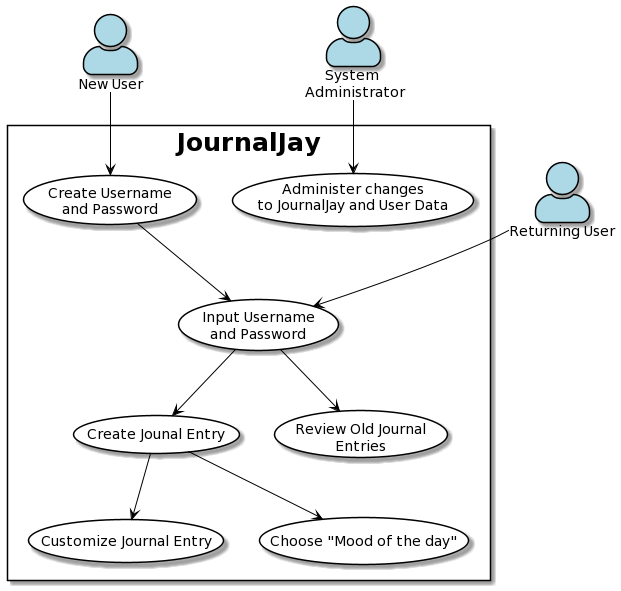
Journal Entry Requirements:

* The user must be able to create one journal entry for each day.
* The date must be auto-time stamped on the journal entry.
* Each journal entry must display in an enclosed square within the main window.
* The user may only input data within the enclosed square.
* While the user is creating the journal entry, they should be able to choose the background and typeface of the entry window from a group of presets.
* The user may a mood from a provided mood selection. The chosen mood must display on the corresponding journal entry icon.
* Once created, the user may only go back to review journal entries.

## Behavior Requirements

### Use Case View

The actors in this use case diagram are the new user, the returning user, and the system administrators. When a new user navigates to JournalJay, they will be prompted to input a username and password or instructed to create a new password. Once a new user creates their username and password, they are considered a returning user. Returning users can create a journal entry, customize their journal entry, choose a “mood of the day”, submit and review old journal entries. System administrators will be able to change the look of the site or the internal data of the site. Admins can also observe, remove, or change user imputed data.



# Other Non-functional Requirements

## Performance Requirements

1. Each page should be visible to the user within 5 seconds.
2. Each page should have buttons clearly marked with options for the user. Selecting an option should only take one click.

All other performance related to storage, memory, and processing should follow standard practices to ensure requirements are minimized.

## Safety and Security Requirements

No personal information including email address or phone numbers will be stored on the site. Each user will have a unique identifier. Due to the limited scope of this project, the username and password will not be heavily protected. The journal entries, username, and password will be stored locally. Optional security requirements include having a way to reset the password and lightly encrypting the usernames and passwords using bcrypt.

The project does not require for the site to be secure. There will be no personal information about the client on the site except the journal entries themselves. The security requirements will be the following:

* The site will be protected by a user created password and username.
* The usernames and passwords will be stored locally on the users’ machine.
* JournalJay will use HTTPS, that will take care of the general web security.
* Optional: encrypt the usernames and passwords with bcrypt.

## Software Quality Attributes

### Reliability Requirements

The application shall be reliable by fetching the associated account information and journal entries upon correct login credentials. This will be achieved using if-and-only-if logic when communicating with the database.

### Availability Requirements

The average system availability should be at least 98% of the time. This allows for approximately 3.4 hours of downtime per week. With basic functionality, JournalJay should not require more downtime, especially in this limited scope.

### Security Requirements

100% of the communication messages in the communication of a log-in session should be encrypted, so data cannot be intercepted from these messages.

An incorrect set of login credentials should elicit a failure message 100% of the time, not allowing a log-in to go through. After three log-in attempts and fails, the application should go into “lock-down” mode for 15 minutes, disabling the log-in function for that particular username.

When a taken username is attempted for a new account, the user should be asked to select a different username 100% of the time. The software shall not accept duplicate usernames.

### Maintainability Requirements

For application extendibility, the JavaScript, CSS, and HTML files shall be written to seamlessly implement new functions in the future. This includes following standard code conventions and writing generally organized and well-annotated code fragments. The system should also have the capability to host up to 100 users simultaneously.

For application testability, test environments should be built to thoroughly test each function. Developers should reference requirements frequently to test against different scenarios.

Appendix A – Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Full Term** | **Definition** |
| HTTPS | Hypertext Transfer Protocol Secure | Common method of secure data transfer on the internet. |
| CS | Computer Science |  |
| HTML | Hypertext Markup Language | Standardized markup language used to tag files to create graphic effects on websites. |
| CSS | Cascading Style Sheets | Style sheet language used to describe the presentation of a file written in a markup language. |
| IEEE | Institute of Electrical and Electronic Engineers | Professional association for electronic and electrical engineering. |
| N/A | bcrypt | A password hashing function used to encrypt data. |
| Admin | Administrator | Someone who is given specific permissions to manage and control the system. |
| App | Application | Software or program developed for users. |
| N/A | User | Someone who interacts with an application in a meaningful way. |

Appendix B - Group Log

10-2-20 10:00am–10:20am Filled out the team agreement form.

10-9-20 9:00am-9:25am Drafted a schedule and rough timeline to complete the SRS document. Discussed concerns and initial ideas in our project requirements.

10-16-20 9:00am-9:45am Sectioned out the SRS document. Set initial deadline of first draft for 10-25-20).

10-30-20 9:45am-10:30am Talked about project specifications and worked on the SRS document.

11-04-20 4:00pm-5:30pm Discussed final needed changes and set up GitHub repo for teamwork.

11-06-20 12:00pm-