Software Requirements Specification

for

Cuse Alumni Groves

Version 1

Prepared by Group 19

Syracuse University CSE687

April 1, 2024

Table of Contents

Table of Figures

Table of Tables

Name	Date	Reason For Changes	Version
Team Wac	3/9/2023	Initial Revision	--
Team Wac	3/21/2023	Added acronym list Added use case diagram and activity diagram Update System Block Diagram	01

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) is intended to delineate software requirements for the customer's gas pump system. This SRS is intended to guide the system's developers to implement the required functionality, as well as the test team to develop appropriate Verification and Validation (V&V) plans and procedures required to demonstrate to the customer that the system was built to this specification.

1.2 Scope

This document specifies the requirements for the following capabilities:

1. User registration and authentication
2. Alumni-student networking features
3. Profile management and customization options
4. Feed page with updates, discussions, and job postings
5. Chat functionality (future scope)

1.3 Definitions

Table 1 Acronyms and Definition

BDD: Block Definition Diagram.
SRS: Software Requirements Specification.
UML: Unified Modeling Language.
V&V: Verification and Validation.

1.4 References
1. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications - Revision of IEEE Std 830-1993

1.5 Overview

This document follows the recommended format specified in IEEE Std 830-1998 IEEE Recommended Practice for Software Specifications. For Section 3, the specific template A.5 for organizing information by feature is followed.

2   Overall Description

2.1 Product Perspective

The Cuse Alumni Groves platform is designed to facilitate communication and collaboration within the Syracuse University community. It offers features for alumni and current students to connect, share insights, and explore opportunities. The platform can be accessed via web browsers or mobile applications, providing a seamless experience across devices. Figure 1 System Block Diagram illustrates the platform's overview, utilizing a Unified Modeling Language (UML) Block Definition Diagram (BDD).

This platform operates as a standalone system but can be integrated with existing university systems for enhanced functionality and data sharing. It is a centralized hub for Syracuse University members to engage, network, and support each other in their academic and professional endeavors.
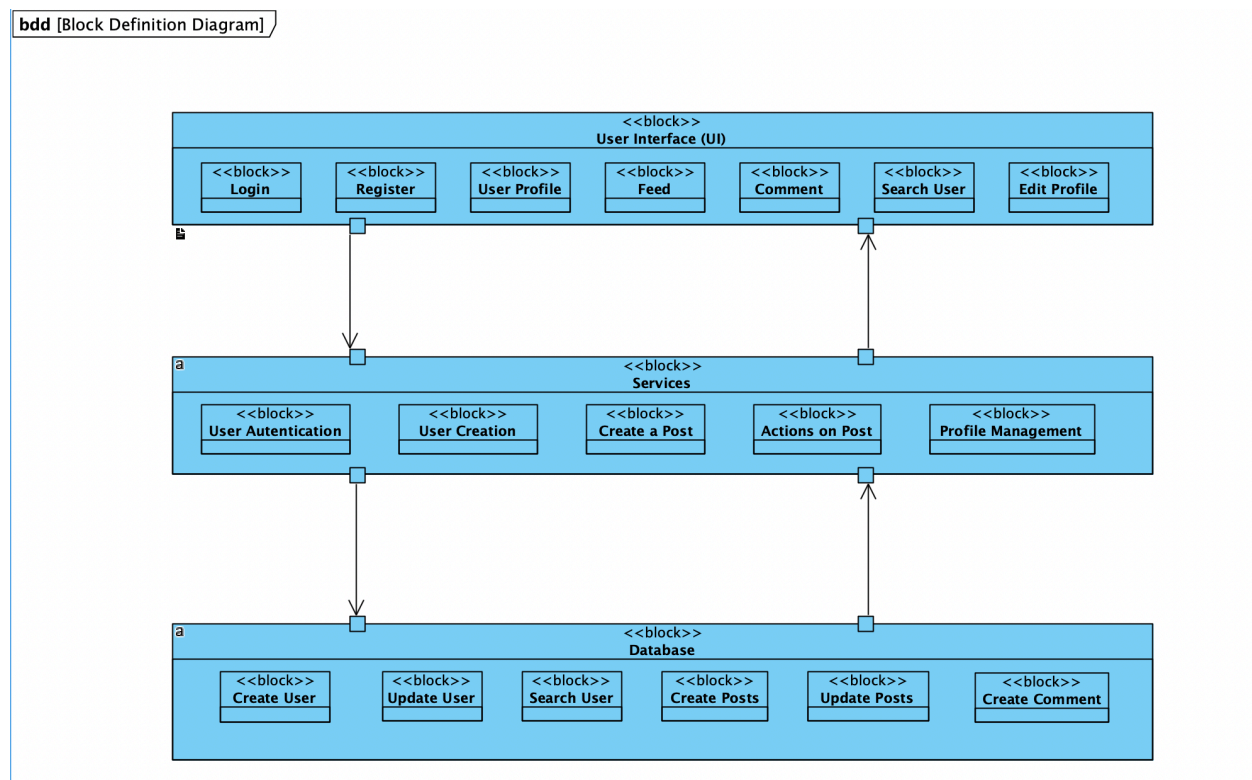
Figure 1 System Block Diagram

2.2 Product Functions

The following use case diagram depicts the users of the system and the intended way in which they will interact with the system.
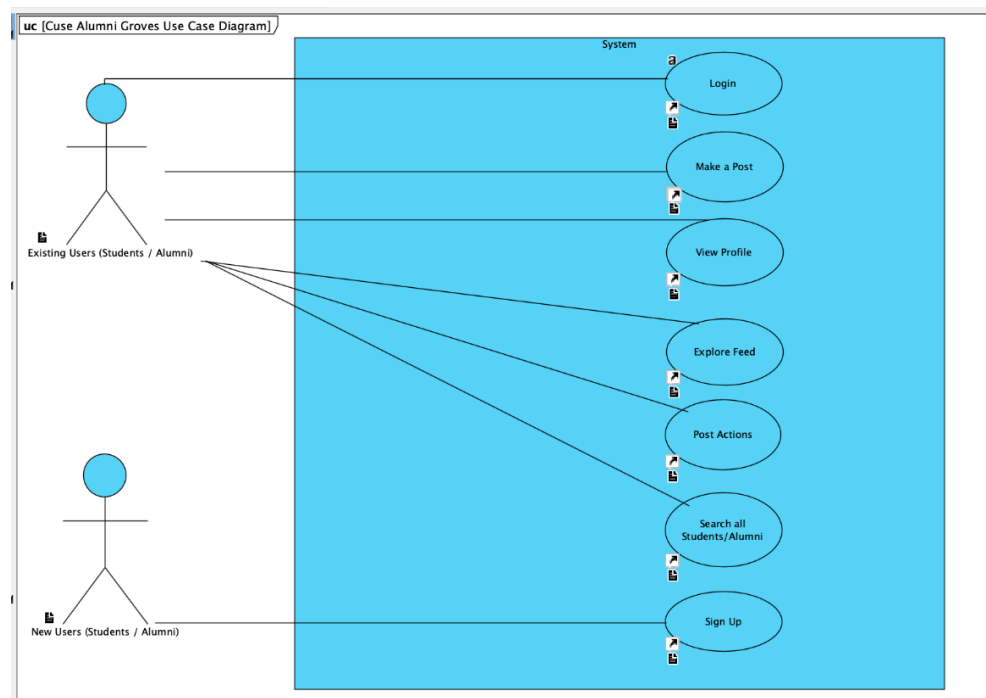
Figure 2 Use Case Diagram for all functionalities
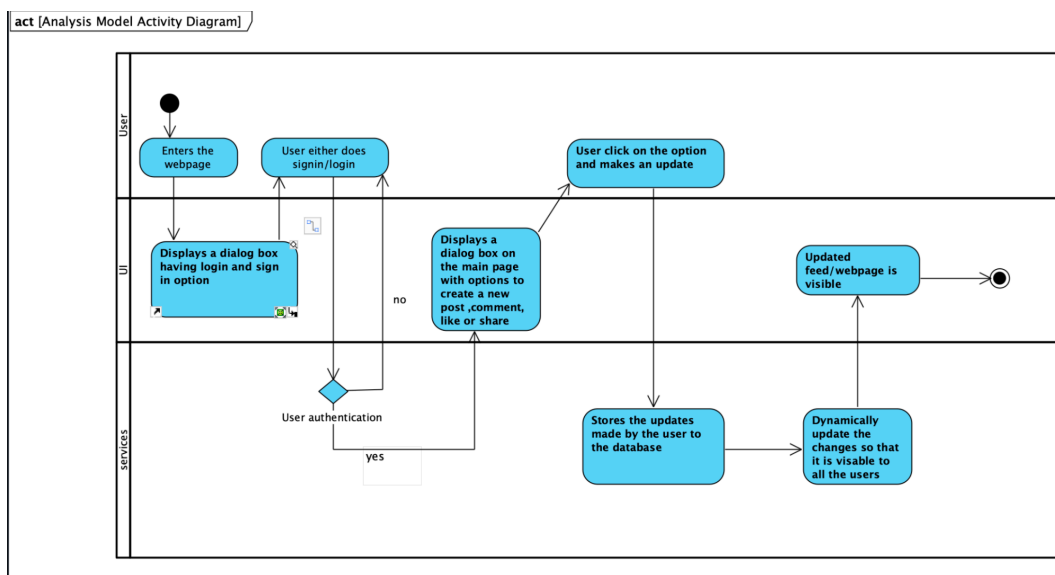
2.3 Product Behaviors

Figure 3 Activity Diagram for stimulus and response sequence

2.4 Product Requirements

Functional Requirements:

1. **User Registration and Authentication:** Users should be able to register with their Syracuse University credentials. Registered users should be able to securely log in to the platform.

2. **Profile Creation and Management:** Upon registration, users should be prompted to create and personalize their profiles with information such as their name, graduation year, major, etc. Users should be able to update and manage their profile information at any time.

3. **Feed Page with Comment Option:** The platform should have a feed page where users can view posts from other users within the Syracuse University community. Users should be able to comment on posts, facilitating engagement and interaction.

4. **Individual User Profile Page:** Each user should have a dedicated profile page showcasing their information, posts, comments, and any other relevant activity within the platform. Users should be able to customize the visibility of their profile information based on privacy settings.

Non-Functional Requirements:

1. **Resilience**: The system should be resilient to failures in individual microservices, with built-in fault tolerance mechanisms to maintain overall system stability and availability.

2. **Flexibility**: Microservices should be loosely coupled, allowing for independent development, deployment, and updates without impacting other services, fostering agility and innovation.

3. **Security:** Each microservice should enforce security measures such as authentication, authorization, and data encryption to protect sensitive information and prevent unauthorized access, ensuring data integrity and confidentiality.