Software Engineering
Software Requirements Specification (SRS) Document

GLookUp
Sep 26, 2023

Version 1.0

By: Usman Zia, Danial Afzal

[I have followed the UNCG Academic Integrity policy on this assignment]

Table of Contents

1. Introduction

1.1. Purpose

The purpose of the GLookUp (GLU) project is to create a web application that connects UNC Greensboro Computer Science students, fostering collaboration and networking. Its primary goal is to enable students to have a profile that showcases their GitHub projects and highlight their rating from previous collaborations.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the GLookUp (GLU) application. Client-oriented requirements describe the system from the client's perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3. Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.
For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to create frontend connectivity with our Database.
MySQL	An Open-source relational database management system.
HTML	Hypertext Markup Language. This is the code that will be used to structure the web application and its content.
CSS	Cascading Style Sheets. This is the code that will be used to design the web application and its content.
AngularJS	A free and open-source JavaScript-based web framework for developing single-page applications.
Spring Web	Will be used to build our web application, one of the dependencies of our system.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create our backend application.
SpringToolsSuite	An Integrated Development Environment (IDE) based on Eclipse and specifically designed for developing Spring Framework-based applications. This will be used to code and run our backend application.
Visual Studio Code	Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft. This will be used to code our application's frontend.
API	Application Programming Interface. This will be used to implement many functions within the software ranging from getting a student's GitHub projects to communicating with our database.

1.4. Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

1.1. Developers:

Developers involved in the project, including front-end and back-end developers, will use this document to understand the system's functional and non-functional requirements. It will help them in the software design and implementation phases.

1.2. Project Managers:

Project managers responsible for overseeing the GLU project will utilize this document to gain insights into the project's scope, objectives, and specific requirements. It will aid in project planning, resource allocation, and monitoring progress.

1.3. Users:

The primary users of the GLU web application, which are UNC Greensboro Computer Science students, will reference this document to understand the features and functionality that the system will offer. It will serve as a reference guide for their interaction with the application.

1.4. Quality Assurance (QA) Team:

The QA team members will use this document to create test cases and ensure that the developed system aligns with the specified requirements. It will guide the testing process to validate the application's correctness and reliability.

1.5. Designers:

Graphic designers and user interface (UI) designers involved in the project will refer to this document to understand the overall structure and layout requirements of the user interface.

1.6. System Administrators:

System administrators responsible for deploying and maintaining the GLU web application will use this document to understand any specific server, database, or infrastructure requirements.

1.7. University Faculty and Staff:

Relevant university faculty and staff members may review this document to ensure compliance with university policies, security standards, and academic objectives.

1.8. Other Stakeholders:

Any other individuals or groups with a vested interest in the GLU project, such as potential sponsors, advisors, or external collaborators, may also refer to this document for an overview of the project's goals and functionality.

By specifying the intended audience for each section of the SRS document, you ensure that the information is presented in a manner that caters to the needs and interests of various stakeholders involved in the GLookUp (GLU) project.

1.5. Project Scope

The software goals of GLookUp align with the broader business goals of UNC Greensboro by promoting collaboration, networking, and the development of valuable skills among computer science students. Ultimately, the success of the app can contribute to the university's mission of providing a high-quality education and producing graduates who excel in networking and collaboration.

The benefits of the project to the university include:

- Enhanced Student Collaboration and Engagement: The primary business goal of the web app is to foster collaboration among UNC Greensboro Computer Science students. This increased collaboration can lead to improved learning outcomes, and a more engaged student community.
- Improved Networking Opportunities: The app's ability to showcase coding profiles and GitHub projects of students allows for building a strong professional network is crucial for students. This aligns with the university's goal of preparing students for successful careers in computer science.
- Reputation Building: By implementing a rating system for collaboration and contributions, the app helps students build a trustworthy reputation within the community. This aligns with the university's goal of producing high-quality graduates who are valued by potential employers.

1.6. Technology Challenges (can be left blank for now)

[Any technological constraints that the project will be under. Any new technologies you may need to use]

1.7. References (can be left blank for now)

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

2. General Description

2.1. Product Perspective

The GLookUp (GLU) web application originates from the need for a centralized platform that connects all UNC Greensboro's Computer Science students, and thereby enhancing effective collaboration on projects.

2.2. Product Features

The product features include the ability for Computer Science students to create their coding profiles, and the ability for administrators to approve new accounts or remove alumni accounts. Students can add their GitHub projects to their profiles, a platform for them to showcase their work. For students, the functionality also includes the ability to rate other students based on previous collaboration experiences, and sending invitations to collaborate with other students. For administrators, the functionality includes the ability to approve new accounts and delete old accounts.

2.3. User Class and Characteristics

Our website application does not expect our users to create an account just to view another student's rating, it's a perfect place for newcomers who are in need of finding great collaborators and they can do so by searching up other students without creating a new account.

2.4. Operating Environment

[Specification of the environment the software is being designed to operate in.]

The application is designed to operate on the web across all devices, hence being fully responsive.

2.5. Constraints (can be left blank for now)

[Any limiting factors that would pose challenge to the development of the software. These include both design as well as implementation constraints.]

2.6. Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The software will be dependent on Spring Web in order to create API endpoints that communicate with our database and respond with the appropriate JSON, this will be developed within SpringBoot. The application will also use the GitHub API (https://api.github.com/) that will display the student's projects on their profiles for other students to see.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

3.1. Primary

- FR0: The system will allow the user to lookup a student's profile based on the student's name. This information will contain the user's name, their collaboration rating, their GitHub projects, and an invitation for the viewing student to collaborate with this student.
- FR1: The system will allow the user to enter their information to make a new account as a Student or as an Administrator in the database. Admin accounts will only have access rights, but Student accounts will have their profile created.
- FR2: The system will allow the student to send an invitation to the other student to collaborate. If a user is not logged in with a Student account, the login will be displayed and a login will be required for the collaboration invite to be sent.

3.2. Secondary

- A Github API call will be made to retrieve the projects of a student to show them on their profile.
- Authorization of a new student account is approved by an Administrator.
- An email from will be sent to the user to whom the collaboration invite needs to be sent along with the profile of the student who wishes to collaborate with them.

4. Technical Requirements

4.1. Operating System and Compatibility

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

1. General User Interface Guidelines

1.1. Consistency: Maintain a consistent user interface design across all screens to ensure a seamless user experience.

1.2. Accessibility: Ensure that the application adheres to accessibility standards to accommodate users with disabilities.

1.3. Responsive Design: Design the UI to be responsive, adapting to various screen sizes and devices.

1.4. User Feedback: Provide clear and informative feedback to users for actions taken within the application.

1.5. Navigation: Include intuitive navigation menus, breadcrumbs, and links to help users move through the application effortlessly.

1.6. Error Handling: Display informative error messages to guide users in resolving issues.

2. Sample Screen Layouts

2.1. Login Page
Input fields for username and password.
"Forgot Password" link.
"Login" button.
"Register" link for new users.

2.2. Registration Page
Fields for user details (name, email, student ID, etc.).
"Submit" button for registration.
"Login" link for existing users.

2.3. User Profile Page
User's profile picture.
Display of user's GitHub projects with ratings.
Edit profile button.
Log out button.

2.4. Project Listing Page
List of available projects.
Filter and search options.

"View Details" button for each project.
Pagination controls.

2.5. Project Details Page
Detailed information about the project.
List of collaborators.
"Collaborate" button.
Back to project listing button.

3. Buttons and Functions

3.1. Common Buttons and Functions (Present on most screens):
Home: Navigate to the user's profile page.
Notifications: Access notifications and alerts.
Search: Initiate a search for projects or users.
Messages: Access the messaging system.
Settings: Access user-specific settings and preferences.

3.2. Profile Page:
Edit Profile: Allow users to update their profile information.
Add GitHub Project: Enable users to add their GitHub projects.
View GitHub Profile: Direct link to the user's GitHub profile.

4. Messages and Alerts

4.1. Success Messages:
Registration successful.
Project added successfully.
Profile updated successfully.

4.2. Error Messages:
Invalid username or password.
Email already exists.
GitHub project not found.
Network error, please try again later.

5. Style Guides

5.1. Color Scheme: Use a consistent color palette that aligns with the GLU branding.

5.2. Typography: Specify font styles and sizes for headings, paragraphs, and other text elements.

5.3. Layout: Define the overall layout structure, including the placement of headers, navigation menus, and content areas.

5.4. Icons: Specify the use of icons for common actions and navigation.

5.5. Images: Define guidelines for image sizes, resolutions, and formats.

5.6. Responsive Design: Ensure that the UI design adapts gracefully to different screen sizes and orientations.

4.2.2. Hardware Interfaces

[All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.]

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

[Determination of all the communication standards to be utilized by the software as a part of the project]

The communication protocol, HTTP, must be able to connect to the local Spring application backend in order to Create, Read, Update, Delete from the Database.

The communication protocol, HTTP, must be able to connect to the GitHub API and return the user's projects.

4.2.4. Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to the used, the database management system and libraries describing the need and the purpose behind each of them.]

We will use Angular to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

5. Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

5.1. Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

- NFR0(R): The local copy of the student account database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the student account database) will consume less than 50MB of memory

5.2. Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

1. Strong Password Policies: Enforce the use of strong, unique passwords and implement password complexity requirements.

2. Multi-Factor Authentication (MFA): Provide the option for users to enable MFA for an added layer of security.
3. Data Encryption: Encrypt sensitive user data, such as personal information and credentials, both in transit and at rest using industry-standard encryption protocols.
4. Data Minimization: Collect and store only the minimum amount of data necessary for the application's functionality.
5. Secure API Endpoints: Implement secure API endpoints with proper authentication and validation of incoming requests.
6. Reporting Mechanism: Provide users with a straightforward way to report abusive or offensive content or users.
7. User Support Channel: Establish a user support channel (e.g., email, chat, or a help center) for users to report security concerns and seek assistance.
8. Vulnerability Scanning: Conduct regular vulnerability assessments and penetration testing to identify and mitigate potential security vulnerabilities.
9. Software Updates: Stay up-to-date with security patches and updates for all software components used in the application.
10. Terms of Service and Acceptable Use Policies: Clearly define and communicate terms of service and acceptable use policies to users, outlining prohibited activities and consequences for violations.

5.3. Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

- NFR4(R): The system will only be usable by authorized users.

5.4. Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

5.4.1. Availability

- The system should be available 24/7 to accommodate different time zones and users' schedules.
- Implement redundancy and failover mechanisms to minimize downtime.
- Regularly perform maintenance and updates during off-peak hours.

5.4.2. Correctness

- Ensure that all information displayed in user profiles, project details, and ratings is accurate and up-to-date.
- Implement input validation to prevent incorrect or malicious data from being entered.
- Conduct thorough testing, including unit testing and system testing, to verify correctness.

5.4.3. Maintainability

- Code should be well-documented, following coding standards and best practices.
- Use version control systems (e.g., Git) to track changes and collaborate on code development.

5.4.4. Reusability

- Design the system with components and modules that can be reused in different parts of the application.
- Implement a clear API or interface for external integration or extension.
- Angular

5.4.5. Portability

- Ensure the application is compatible with various web browsers (e.g., Chrome, Firefox, Edge, Safari) and their different versions.
- Make sure the system can run on multiple operating systems (e.g., Windows, macOS, Linux).
- Consider mobile responsiveness for different screen sizes and devices.

5.5. Process Requirements

5.5.1. Development Process Used

Agile: This approach promotes collaboration, adaptability, and incremental development. It involves regular feedback from stakeholders and developers, which can be valuable for a project like GLookUp.

5.5.2. Time Constraints

Sprint/Iteration Duration (for Agile): Sprints are utilized for a duration of one Week until project completion

The GLookUp (GLU) project aims to deliver a Minimum Viable Product within 4 months from the project initiation date. This MVP will include core features such as user registration, GitHub project integration, and basic profile creation and viewing. Subsequent features and enhancements will be delivered incrementally in subsequent releases based on user feedback and project priorities.

5.5.3. Cost and Delivery Date

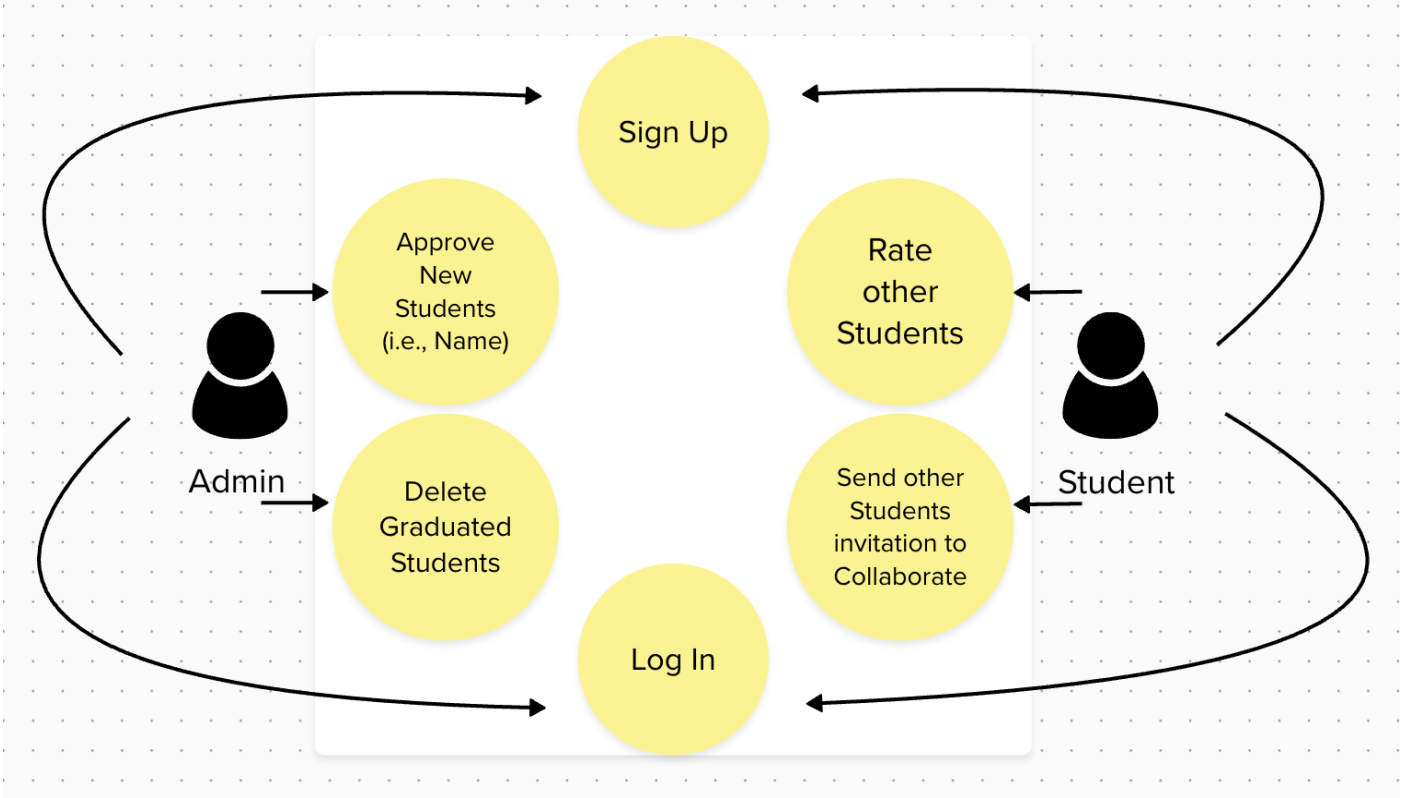
The estimated cost for the development of the GLookUp (GLU) project is $150,000, which includes personnel costs, infrastructure, and software tools. This budget will be allocated as follows:

Personnel Costs: $100,000
Infrastructure and Hosting: $30,000
Software Tools and Licenses: $10,000
Contingency: $10,000
The planned delivery date for the MVP is December 5, 2023. The project team will regularly assess progress against this schedule and make necessary adjustments to ensure timely delivery. The budget and schedule will be subject to change as the project evolves, and stakeholders will be informed of any deviations from the initial estimates.

5.6. Other Requirements

TBD

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Admin (Danial Afzal)

- **Approve new accounts**: When a new account is created, the admin verifies the account's UNCG email and name, and then approves of the account
- **Remove old accounts**: The admin keeps up with the inactive or old accounts of students who have become alumni of UNCG or administrators who are no longer active, then those accounts are removed.

5.8.2. Actor: Student (Usman Zia)

- **Rate students**: Any student can visit a student's profile and give them a rating from 1 - 5 stars, based on their previous collaboration together. As part of the rating process the student also answers a Yes or No question that asks, Would you collaborate with this Student again?
- **Send invitation to collaborate**: Any student can visit a student's profile and if they like a student's ratings and projects, they can send an invitation to collaborate which directly sends an email to that student with the information of the user who wishes to collaborate with them

5.9. Use-Case Model Scenarios

5.9.1. Actor: Admin (Danial Afzal)

- **Use-Case Name**: Approve New Accounts
  - **Initial Assumption**: A new user has requested account approval.
  - **Normal**: Danial Afzal verifies the UNCG email and name, approves the account, and sends a confirmation email to the user.
  - **What Can Go Wrong**: If the provided information is incorrect or invalid, Danial Afzal rejects the account request and notifies the user.
  - **Other Activities**: Danial Afzal logs the approval/rejection and account details for auditing.

- **System State on Completion:** The user account is either approved and active or rejected.

- **Use-Case Name**: Remove Old Accounts
  - **Initial Assumption**: There are inactive or old student accounts that need to be identified and removed.
  - **Normal**: Danial Afzal identifies and removes accounts of alumni or inactive students.
  - **What Can Go Wrong**: None in the normal flow.
  - **Other Activities**: Danial Afzal keeps a record of removed accounts for auditing purposes.
  - **System State on Completion**: Inactive or old accounts are removed from the system.

5.9.2. Actor: Student (Usman Zia)

- **Use-Case Name**: Rate Students
  - **Initial Assumption**: Usman Zia wants to rate another student with whom he collaborated.
  - **Normal**: Usman Zia visits the student's profile, provides a rating (1-5 stars), and submits it.
  - **What Can Go Wrong**: None in the normal flow.
  - **Other Activities**: The system records the rating for future reference.
  - **System State on Completion**: The student receives the rating, which is reflected on their profile.

- **Use-Case Name**: Send Invitation to Collaborate
  - **Initial Assumption**: Usman Zia wants to invite another student to collaborate.
  - **Normal**: Usman visits the target student's profile, decides to collaborate, and sends an invitation.
  - **What Can Go Wrong**: The target student may decline the invitation.
  - **Other Activities**: The system sends an email to the target student with collaboration details.
  - **System State on Completion**: The invitation is sent, and both students are notified

6. Design Documents

6.1. Software Architecture

6.2. High-Level Database Schema

6.3. Software Design

6.3.1. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.2. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.3. State Machine Diagram: Actor Name (Responsible Team Member)

6.4. UML Class Diagram

7. Scenario

7.1. Brief Written Scenario with Screenshots