Software Engineering
Software Requirements Specification (SRS) Document

GLookUp
https://github.com/usman-z/GLookUp.git

Dec 6, 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 REST APIs that connect with our Database to create or read and return JSON data.
MySQL	MySQL is an open-source relational database management system (RDBMS) that stores, manages, and retrieves data in structured tables. In this project, MySQL serves as the backend database to store and manage all data.
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. In this project, AngularJS is used to create an interactive and user-friendly front-end interface for our application GLookUp (GLU)
Spring Web	A framework within the Java Spring ecosystem that provides tools and libraries for building web applications. Spring Web is used to create the backend infrastructure, enabling the handling of HTTP requests, routing, and overall web functionality.
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.

REST API	REST API (Representational State Transfer Application Programming Interface) is a set of rules and conventions for building and interacting with web services using standard HTTP methods. These RESTful endpoints will be used make API calls over HTTP and get return JSON responses upon communicating with the database.
JSON	JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy to read and write. We will use this data-interchange format for data transmission between our server app and the client app.

1.4. Intended Audience

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

- Integration with GitHub API:
    - Challenge: Ensuring smooth integration with the GitHub API to fetch and display student projects and repositories.
    - Solution: Thoroughly understanding the GitHub API documentation, handling authentication, and addressing rate-limiting issues.
- Real-time Collaboration:
    - Challenge: Implementing real-time collaboration features, such as live updates on project changes or collaboration invitations.
    - Solution: updating the main database, instead of the individual

1.7. References

1. Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2018). The Java® Language Specification, Java SE 11 Edition. Addison-Wesley.
2. MySQL. (n.d.). MySQL Documentation. Retrieved from https://dev.mysql.com/doc/
3. W3C. (n.d.). HTML Living Standard. Retrieved from https://html.spec.whatwg.org/multipage/
4. W3C. (n.d.). Cascading Style Sheets (CSS) - The Official Definition. Retrieved from https://www.w3.org/Style/CSS/
5. AngularJS. (n.d.). AngularJS Documentation. Retrieved from https://docs.angularjs.org/

6. Spring Framework. (n.d.). Spring Framework Documentation. Retrieved from https://docs.spring.io/spring-framework/docs/current/reference/html/web.html
7. Spring Boot. (n.d.). Spring Boot Documentation. Retrieved from https://docs.spring.io/spring-boot/docs/current/reference/html/web.html
8. Visual Studio Code. (n.d.). Visual Studio Code Documentation. Retrieved from https://code.visualstudio.com/docs
9. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
10. JSON. (n.d.). JSON - JavaScript Object Notation. Retrieved from https://www.json.org/
11. UNC Greensboro. (n.d.). About UNCG. Retrieved from https://www.uncg.edu/about/
12. GitHub. (n.d.). GitHub REST API. Retrieved from https://docs.github.com/en/rest
13. Microsoft. (n.d.). GitHub API Authentication. Retrieved from https://docs.github.com/en/rest/overview/resources-in-the-rest-api#authentication
14. University of California, Berkeley. (n.d.). What is REST. Retrieved from https://restfulapi.net/
15. Martin, R. C. (2003). Agile Software Development, Principles, Patterns, and Practices. Pearson Education.
16. The Apache Software Foundation. (n.d.). Apache Maven. Retrieved from https://maven.apache.org/

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

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

2.5. Constraints

- **Technology Stack Compatibility:**
  - The project is constrained by the compatibility of technologies chosen, such as AngularJS, SpringBoot, MySQL, and the GitHub API. Any updates or changes to these technologies may require corresponding adjustments to the system.
- **GitHub API Limitations:**
  - The integration with the GitHub API may be subject to rate limiting or other restrictions imposed by GitHub.
- **Browser Compatibility:**
  - The frontend, developed using AngularJS, may face challenges related to browser compatibility. Ensuring consistent and reliable performance across different browsers is crucial for the user.
- **Dependence on External APIs:**
  - The functionality of fetching and displaying GitHub projects relies on the stability and availability of the GitHub API. Any disruptions or changes to this external API may impact the application's core features.
- **Development Timeframe:**
  - The project may be constrained by a fixed development timeframe. Delays in development may affect the ability to deliver the product within the specified schedule.
- **Budget Constraints:**
  - The project is subject to budget constraints that may limit the acquisition of certain tools, technologies, or additional resources. This may impact the scalability and extensibility of the system.
- **Skill Set of Development Team:**
  - The expertise and skill set of the development team impose constraints on the complexity and sophistication of certain features. essential to align the project's requirements with the team's capabilities.
- **Database Schema Changes:**
  - Changes in the database schema, especially related to MySQL, may have implications for the existing data and functionalities. Careful consideration and planning are required to manage such changes without compromising data integrity.

2.6. Assumptions and Dependencies

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

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

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.
Logout 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 the 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

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

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

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

5.1. Performance 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

1. Strong Password Policies: Enforce the use of strong, unique passwords and implement password complexity requirements.
2. Data Encryption: Encrypt sensitive user data, such as personal information and credentials, both in transit and at rest using industry-standard encryption protocols.
3. Data Minimization: Collect and store only the minimum amount of data necessary for the application's functionality.
4. Secure API Endpoints: Implement secure API endpoints with proper authentication and validation of incoming requests.
5. Reporting Mechanism: Provide users with a straightforward way to report abusive or offensive content or users.
6. 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.
7. Vulnerability Scanning: Conduct regular vulnerability assessments and penetration testing to identify and mitigate potential security vulnerabilities.
8. Software Updates: Stay up-to-date with security patches and updates for all software components used in the application.
9. 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

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

5.4. Software Quality Attributes

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:

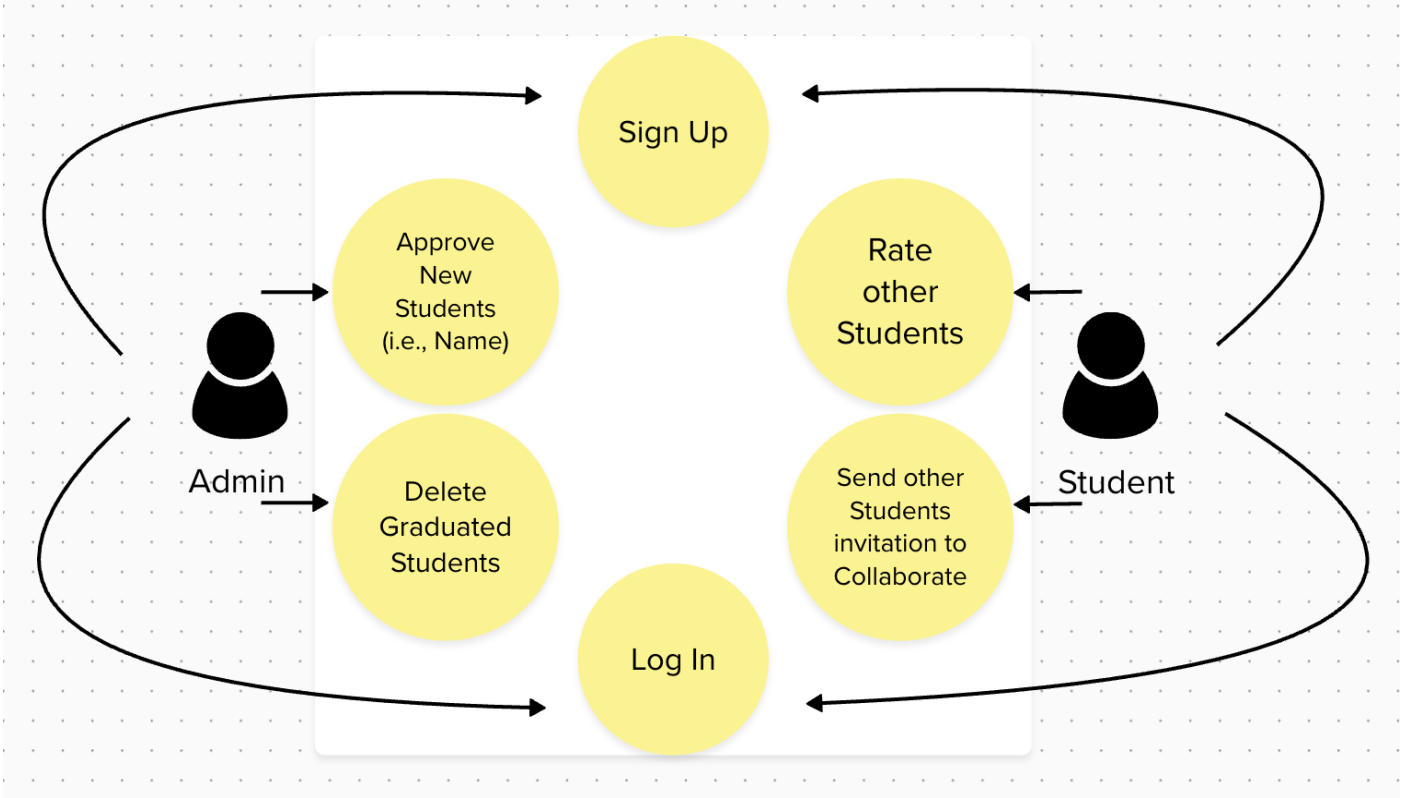
Personnel Costs: $100,000
Infrastructure and Hosting: $30,000
Software Tools and Licenses: $10,000
Contingency: $10,000
The planned delivery date for the MVP is December 5, 2023. The project team will regularly assess progress against this schedule and make necessary adjustments to ensure timely delivery. 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

- **Regulatory Compliance:**
  - Ensure compliance with relevant data protection and privacy regulations, such as GDPR, FERPA, or any other applicable laws. The system must adhere to legal requirements for handling user data.
- **Documentation:**
  - Maintain comprehensive documentation for both users and developers. This should include user guides, API documentation, and system architecture documentation to facilitate understanding and future development.
- **Scalability:**

    - Design the system to be scalable to accommodate an increasing number of users and data. Implement strategies such as load balancing and optimization to ensure performance as user traffic grows.
- **User Training:**
    - Develop training materials and conduct training sessions for users to familiarize them with the features and functionalities of the GLookUp web application. Provide ongoing support and resources for users to maximize their utilization of the platform.
- **Accessibility:**
    - Ensure that the web application adheres to accessibility standards (e.g., WCAG) to make it usable by individuals with disabilities. This includes considerations for users with visual, auditory, or motor impairments.
- **Feedback Mechanism:**
    - Implement a feedback mechanism within the application to gather input from users regarding their experience, suggestions for improvement, and identification of any issues. Use this feedback to enhance the user experience and address concerns promptly.
- **Integration with University Systems:**
    - Integrate the GLookUp web application with existing university systems, if applicable. Ensure compatibility with university authentication mechanisms and databases to streamline user access and data management.
- **Mobile Application Consideration:**
    - Evaluate the feasibility of developing a mobile application version of GLookUp to enhance accessibility and user engagement. Consider user preferences and trends related to mobile usage.
- **Backup and Recovery:**
    - Establish regular backup procedures for the student account database and other critical data. Develop a robust recovery plan to minimize data loss in the event of system failures or unforeseen incidents.
- **Cross-Browser Compatibility Testing:**
    - Conduct thorough cross-browser compatibility testing to ensure that the GLookUp web application functions seamlessly across various web browsers and their different versions.
- **Usability Testing:**
    - Perform usability testing with representative users to evaluate the user interface, navigation, and overall user experience. Use the insights gained to make iterative improvements to the design.
- **Adherence to University Branding Guidelines:**
    - Ensure that the visual design, color schemes, and branding elements align with the university's branding guidelines. Maintain a consistent and professional look and feel throughout the application.

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.

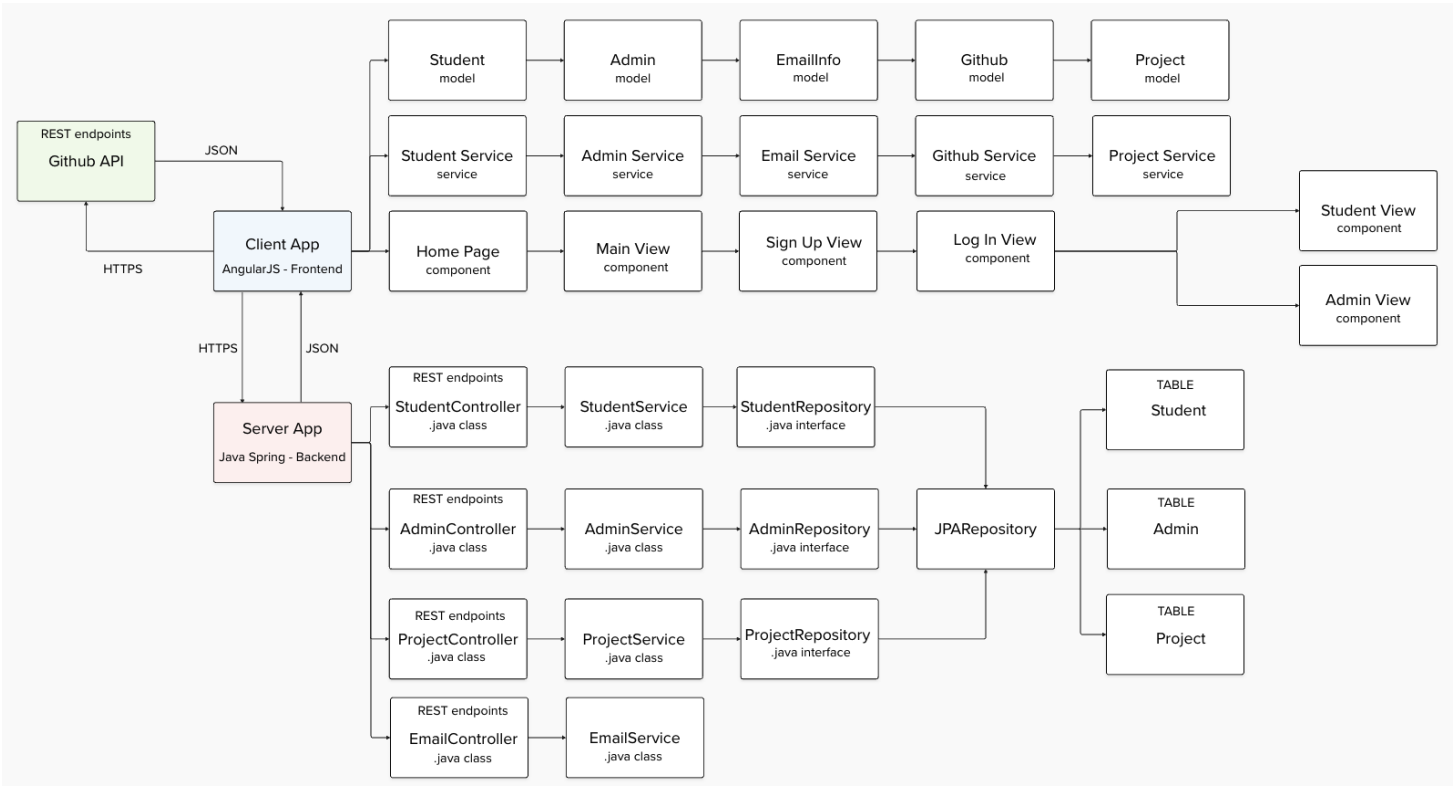
5.9.2. Actor: Student (Usman Zia)

- **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.

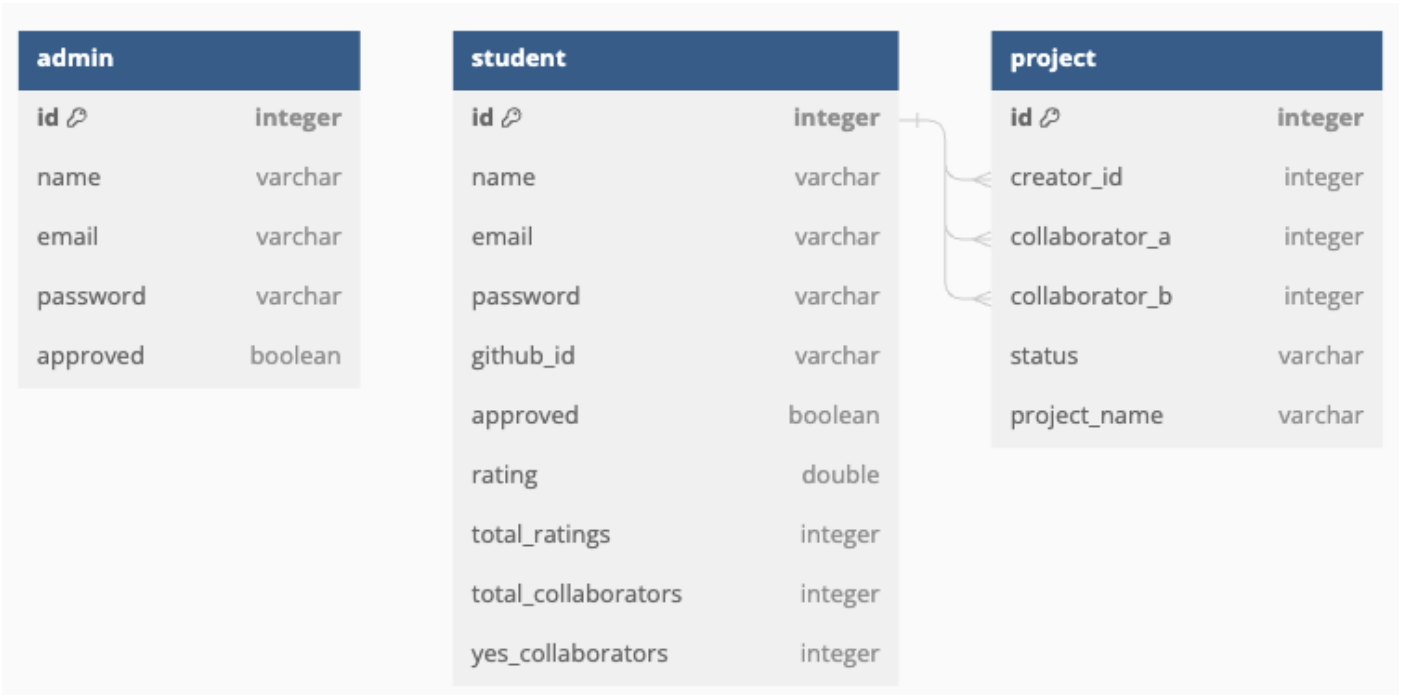
- **Use-Case Name**: Send Invitation to Collaborate
  - **Initial Assumption**: Usman Zia wants to invite another student to collaborate.
  - **Normal**: Usman visits the target student's profile, decides to collaborate, and sends an invitation.
  - **What Can Go Wrong**: The target student may decline 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



Student Table:

- id (integer): This is the primary key for the student entity and serves as a unique identifier for each student.
- name (char): This field stores the name of the student.
- email (char): The student's UNCG email address is stored in this field, serving as their username for authentication.
- github_id (char): The student's GitHub ID is stored in this field, and used to get their public repositories.
- password (char): The student's password is stored here, usually after being hashed and salted for security.
- approved(boolean): The student account’s approval status is stored here
- rating (double): This field represents the overall rating of the student, typically on a scale from 1 to 5.
- total_ratings (int): This field stores the total number of ratings a student has received, allowing for the calculation of their average rating.
- yes_collaborators (double): This field is used to calculate the percentage of positive responses when students are asked, "Would you collaborate with this student again?"
- total_collaborators (int): This field stores the total number of collaboration invitations received by the student.

admin Table:

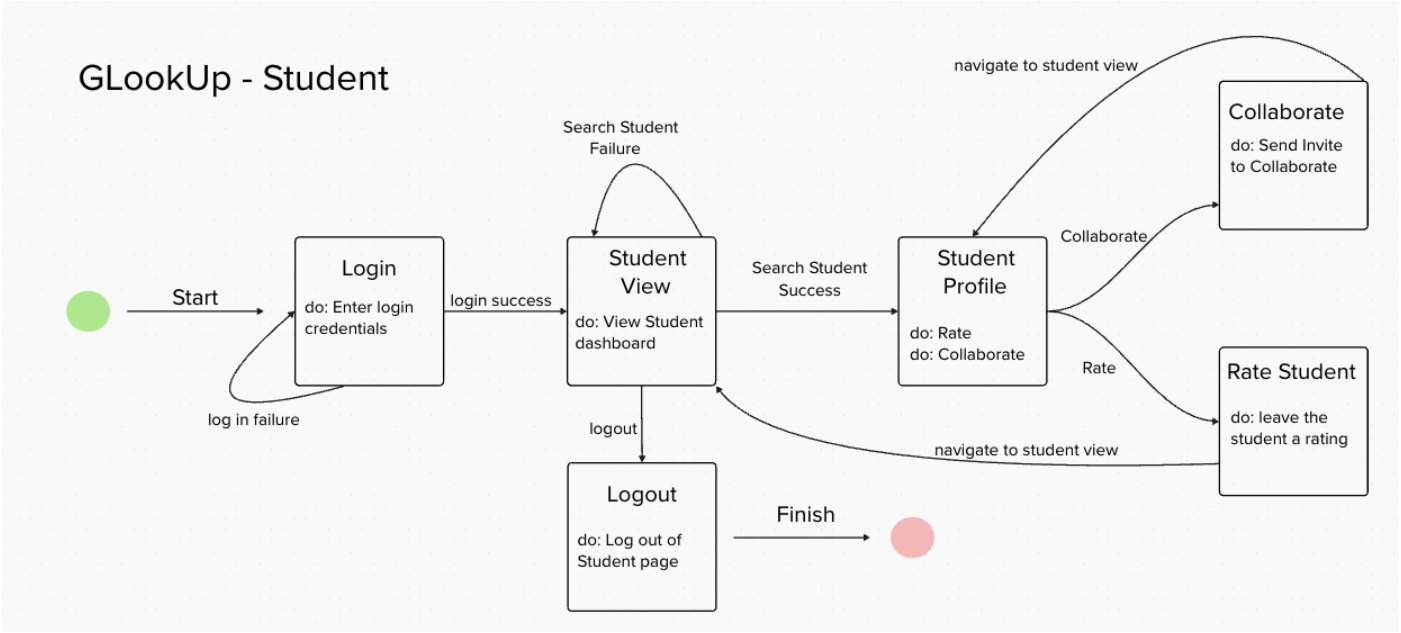
- id (integer): The primary key for the admin entity, ensuring a unique identifier for each admin.
- name (char): This field stores the name of the admin.
- email (char): The admin's email address is stored here, typically used for authentication.
- password (char): The admin's password, secured
- approved(boolean): The admin account’s approval status is stored here

project Table:

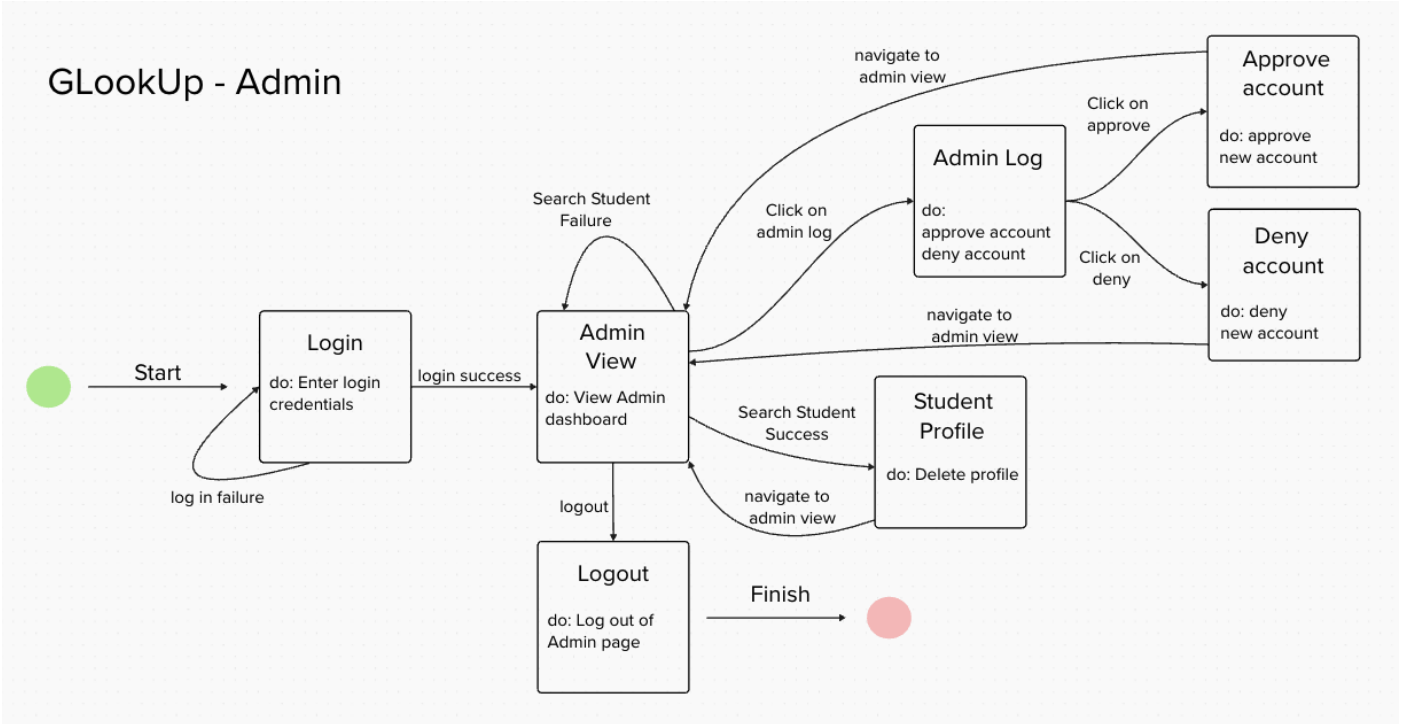
- id (integer): The primary key for the project entity, ensuring a unique identifier for each project.
- creator_id (integer): This field stores the studentId of the creator for the project.
- collaborator_a (integer): studentId of collaborator A.
- collaborator_b (integer): studentId of collaborator B.
- project_name (char): The name of the Project
- status (char): The status of the project (pending, active, or done).

6.3. Software Design

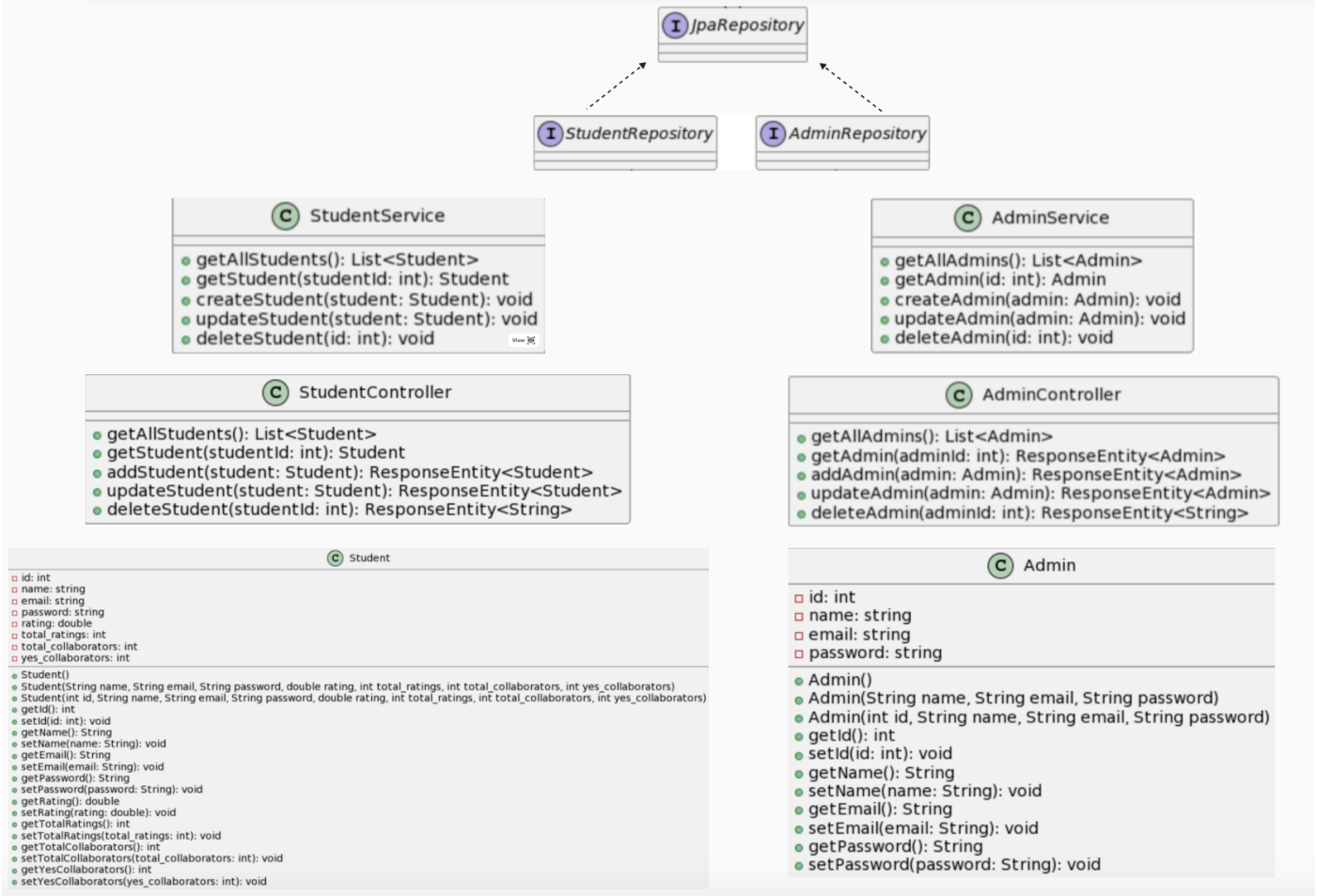
6.3.1. State Machine Diagram: Student (Usman Zia)



6.3.2. State Machine Diagram: Admin (Danial Afzal)



6.4. UML Class Diagram

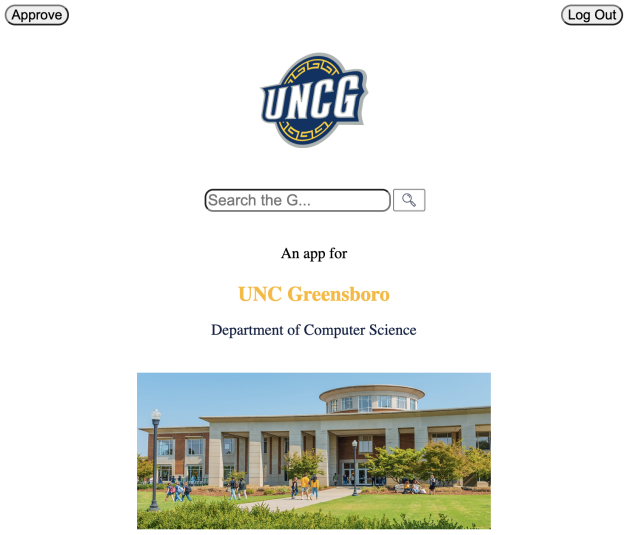


7. Scenario

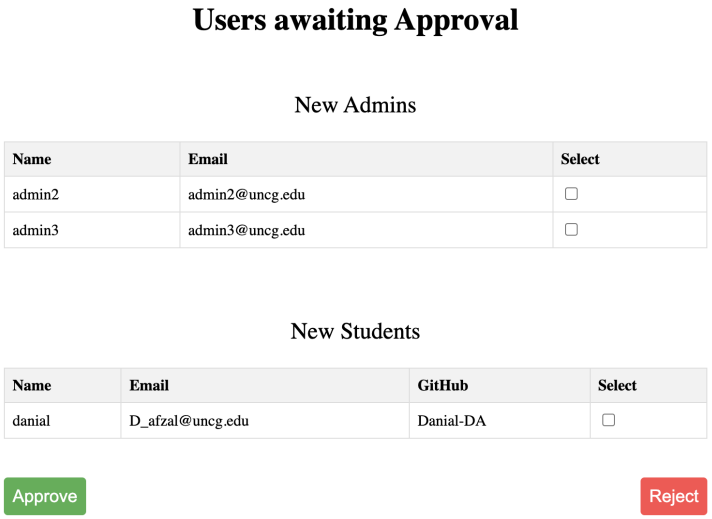
7.1. Brief Written Scenario with Screenshots

Scenario for Testing Glookup:

1. **Account Approval by Admin use case:**
    - Actor: Admin
        - An admin, Admin_A, logs into the platform



        - Student_X registers for an account for the first time using name, github id, email.
        - Admin_A verifies Student_X's UNCG email , name, github id.

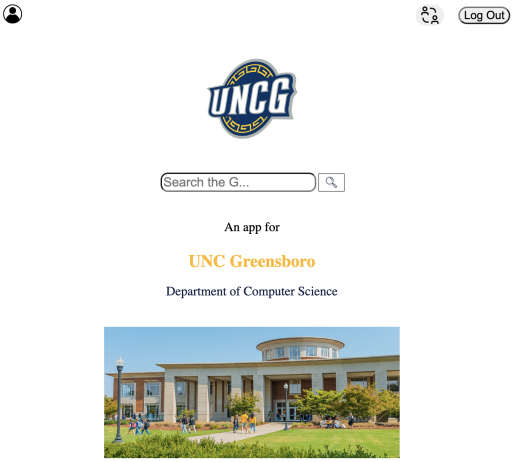


        - Admin_A approves Student_X's account.

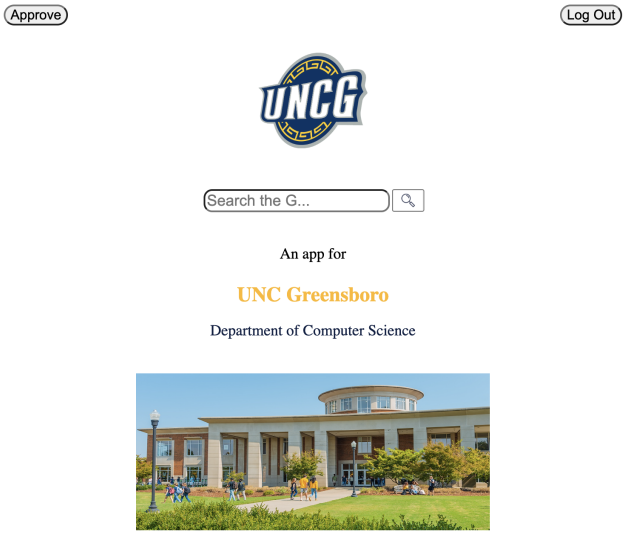
Success! **Approved successfully**

Go Home

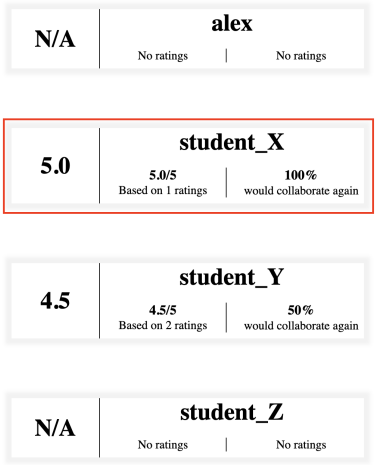
- Admin_A logs out.
- Student_x now logs in and can use Glookup



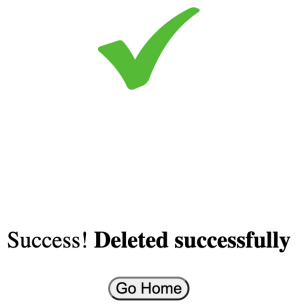
2. **Removal of Inactive Accounts use case:**
    - Actor: Admin
        - Admin_B logs into the platform.



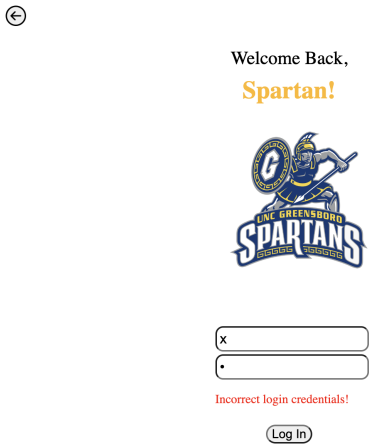
- Admin_B identifies the account of inactive student (student_x).



- Admin_B clicks the profile and deletes the account of inactive student (student_x).

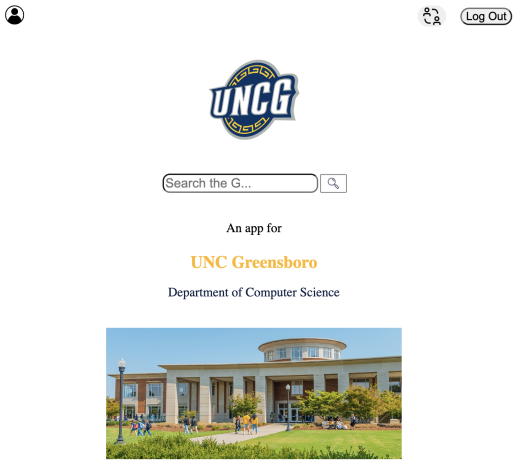


- Admin_B logs out.
- Student_x tries to login

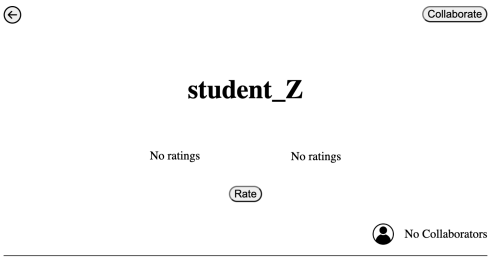


- Student_x cannot login , his account is no longer present

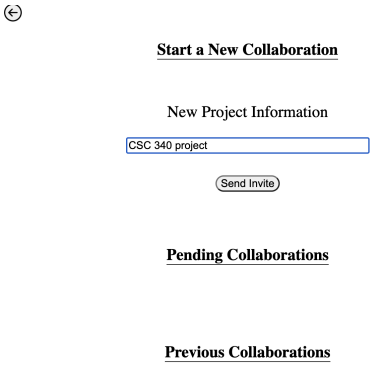
3. **Student collaboration with others use case:**
    - Actor: Student
        - Student_Y logs into the platform



        - Student_Y visits another student's profile, Student_Z.



        - Student_Y decides to collaborate and sends an invitation to Student_Z.

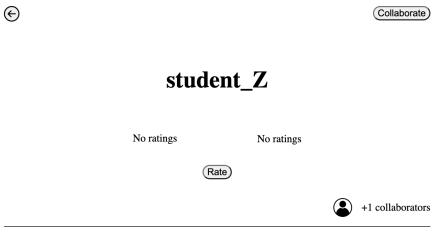


        - Student_Y logs out
        - Student_Z logs in
        - Student_Z receives an email notification about the collaboration invitation

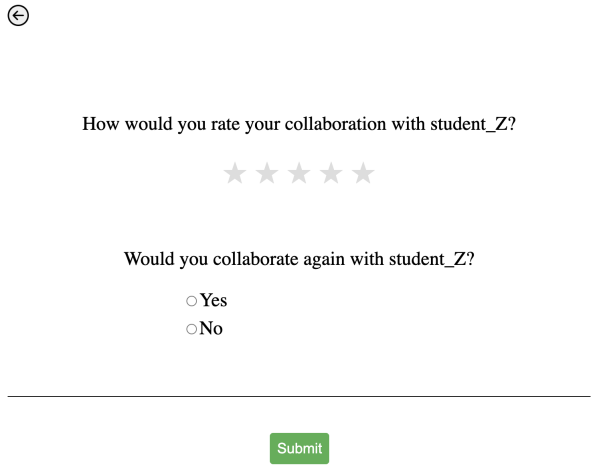
- Student_Y accepts the invitation and the project is moved to the active collaboration

- Student_Z logs out

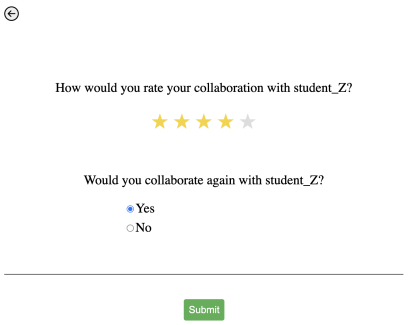
4. **Student Rating other student use case:**
   - Actor: Student
     - Student_Y logs into the platform.
     - Student_Y visits the student's profile he collaborated with , Student_Z.



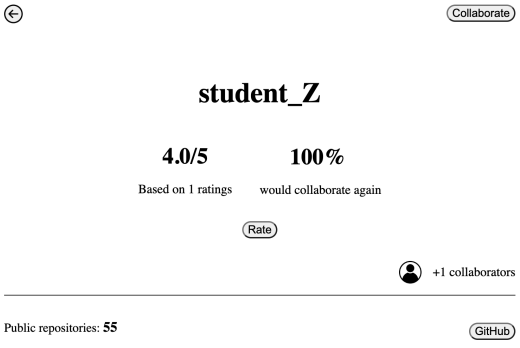
- Student_Y clicks rate



- Student_Y rates Student_Z with 4 stars and answers "Yes" to the collaboration question.



- Student_Y logs submits and logs out
- Student_Z's rating and collaboration rate has been updated.



- Student_Y logs out.