Software Requirements Specification

for

Code Management and Version Control System

Version 1.0 approved

Prepared by Arjun Chengappa, Lavitra Kshitij Madan, Vishnu Charan Golugula

Team: A16

PES University

February 4, 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Suraj MS, R.Shrenik, Niranjan Bhaskar K, Adarsh Nair	06/02/21	Document Creation	1.0
Arjun, Lavitra, Vishnu	19/04/21	Lack of time and resources	1.1

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description and the requirements for the Code Management and Version Control System. Illustrating the purpose and complete declaration for the development of this system, elaborating on its system constraints, interface and interactions with other external applications, this document is primarily intended to be used to propose to a customer for its approval.

1.2 Intended Audience and Reading Suggestions

While the software requirement specification (SRS) document is written for a more general audience, this document is intended for software developers, project consultants, and team managers.
This document need not be read sequentially, users are encouraged to jump to any section they find relevant. Below is a brief overview of each part of the document.

1. **Introduction**
   This section offers a summary of the Code Management and Version Control System including relevant objectives and the product scope of the intended platform.

2. **Overall Description**
   This section includes the context and origin of the platform, a precise summary of the major functions that the product performs or lets the user perform. Additionally, a top-level data flow diagram of the Code Management and Version Control System is also included. Further, various user classes, its characteristics and its operating environment are elaborated. The section is concluded by describing any items or issues that will limit the options available to the developers and further elaborate upon assumptions and dependencies.

3. **External Interface Requirements**
   This module includes description of the logical characteristics of each of the user interfaces, software interfaces (including databases, operating systems, tools and libraries) of the system.

4. **Analysis Models**
   Includes a use case diagram to showcase the working of the system.

5. **System Features**
   This component illustrates the organization of the functional requirements of the product by highlighting system features and the major services provided.

6. **Other Non-Functional Requirements**
   This section elaborates on the performance, safety, and security requirements. Additionally, it also constitutes some quality characteristics and operating principles for the product.

7. **Other Requirements**
   Any other requirements apart from the ones specified above.

1.3 Product Scope

The Code Management and Version Control System is an internet code hosting platform for effective collaboration and distributed version control. The system manages and stores revisions of projects, while tracking contributions from the users. Although it is traditionally used for code, it could be used to manage any other type of file such as a word document or a final cut project. It can be thought of as a ledger containing all the changes done to the document by different collaborators.

Features:

- Effective collaboration between users on the platform.
- Commits to hosted code / other data on the platform
- Issue tracking with labels (Urgent, Minor, Major).
- Incremental Search (Search-as-you-type) feature for finding users, repositories, issues etc.
- Email notifications.
- Task lists, document viewer, code viewer

1.4 References for the SRS creation

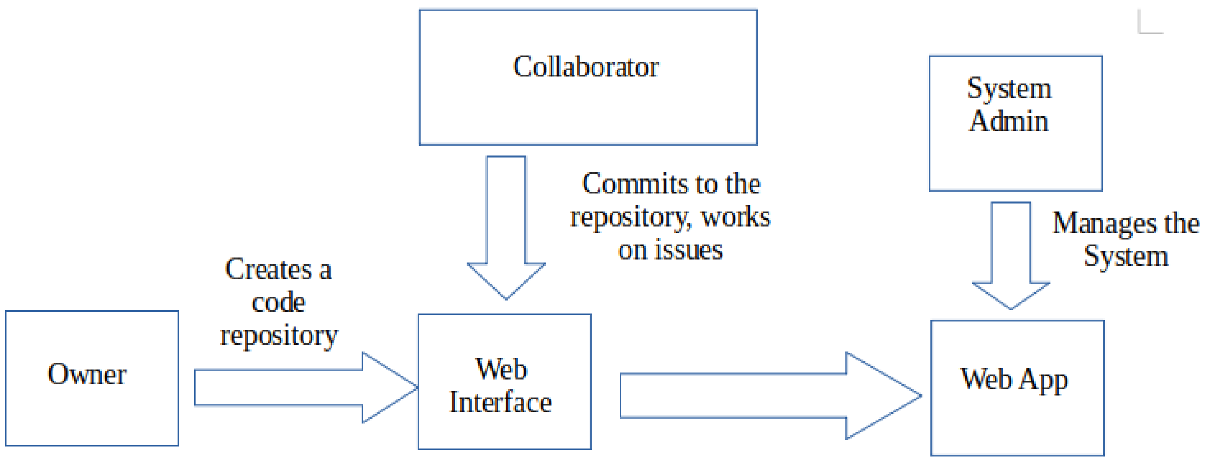
- IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.
- Wiegers, Karl, Software Requirements (3rd Edition), Microsoft Press 2013.
- Standardized Statement Templates – beginning to write SRS
  - Link: Coursera - Standardized Statement Templates

2. Overall Description

2.1 Product Perspective

VCSs date back to the 70s. Version control systems records changes to a file or a set of files. A distributed VCS is one where individual users each have a copy of the history which lists all the changes done to the file or collection of files. A Code Management System works with the VCS, and provides additional features such as an intuitive interface allowing for easy collaboration, issue creation and tracking and controlling access to the set of files. The perspective of different sets of users are shown below in the diagram

The Product is a web application to help users (or different groups of them) maintain and manage the process of the development. It will also allow the users to assign levels of access for the files stored in the repository. There are mainly three user perspectives to the product. The Owner owns the repository and can modify the repository, allow collaborators, manage their access and has near unrestricted access to the files. The Collaborators can work on the issues filed on the repository. They can also commit to the repository subject to reviews to modify the origin repository. Normal users who aren’t collaborators can either request the owner/collaborator for access or work on an existing issue. The System admin manages and maintains the web application and ensures the continual operation of the services provided by it.


Collaborator
Commits to the repository, works on issues
System Admin
Manages the System
Creates a code repository
Owner
Web Interface
Web App

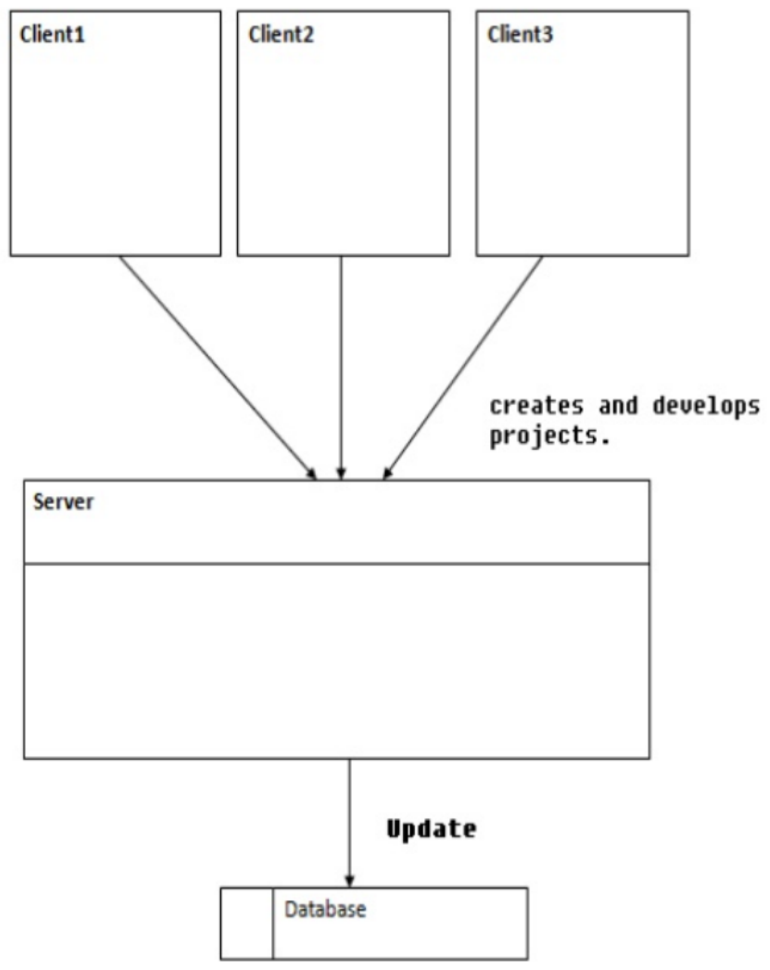
System Perspective Diagram

2.2 Product Functions

The product is meant to provide the below features to the users :-

- Hosting content (code, docs etc) on the web.
- User authentication and login.
- Search for different repositories which are designed to be public.
- Repository ownership and Access Control.
- Interface to post issues, review them with comments and tags (closed, pending, open etc)
- Maintain an insights section showcasing most active contributors and other repo activity

Data flow diagram



2.3 User Classes and Characteristics

User Classes	Frequency of Use	Functions Used	Privilege Level	Characteristics
Owner(s) of the repository.	Medium-High	Has unrestricted access to the repository ● managing access ● Request approval for contributor addition, commits ● Can comment. ● issue creation, deletion, tracking ● overall maintenance	High	Their day-to-day interaction with the system would include approving commits and taking important design decisions wrt the project

Developers and designers	High	● Committing changes ● Bug tracking, fixing Based on the access provided to them by the owner, they will be able to make changes to the repository which will be reviewed by a senior dev or repo owner.	Medium-High	Developers form the majority of the user base. Their day-to-day activities would include commiting changes to the repo and requesting approvals.
New collaborators	Low	● Request approval ● Bug fixing (if allowed)	Low	New users wishing to contribute to a repo have the option of fixing bugs and committing changes to a repository subject to approval.
System Administrator	Low-Medium	● Website Maintenance and Security ● Handling the application's server-side and client-side issues. ● Bug fixes (if any) ● Overseeing day to day operations	High (wrt the system itself)	Sys-admins handle the end-to-end maintenance and security of the web application.

The most important user classes include the developers and collaborators interacting with the system on a day-to-day basis from both a technical perspective and a business perspective. The system in place allowing for seamless usage requires constant maintenance without which the user base would decline.

2.4 Operating Environment

Environment: Cross-browser responsive web application. Will run on any desktop OS that comes with a web browser.
Hardware: Not Applicable. Use online hosting services.
Version: 1.0
Other dependencies: None

2.5 Design and Implementation Constraints

- **Regulatory policies:**
  - Keeping content on website civil (Should not host abusive / unlawful content on website)
  - Overloading servers with extremely high number of requests may lead to the user getting banned from the platform.

- **Potential Hardware and Software problems:**
  - Software requirements: A web browser
  - May not work on mobile operating systems
  - Multiple users pushing code into the same branch simultaneously can result in merge conflicts.
  - Upload of files through the web application is limited by the client browser.

- **Security considerations:**
  - Use HTTP for the website.
  - Two-factor Authentication (2FA). Not implemented

- **Others:**
  - The system might be limited by the number of active servers.

2.6 Assumptions and Dependencies

- Developers have experience creating basic user interfaces on the web and working with databases (relational or non-relational).
- Familiarity with Git VCS. delete this point
- Familiarity with implementing basic security protocols on a website.

3. External Interface Requirements

3.1 User Interface

- **Home Page**
  - First screen that the user is able to view on the website.

    - Has a Nav bar containing buttons depending on whether or not the user has signed in.

- **Login**
    - Contains a form with two inputs: Username and Password.
    - Shows the respective errors when an invalid login is attempted.
    - Redirects to homepage upon successful login

- **Signup**
    - Contains a form with 4 inputs:
        - Name
        - Username
        - Email
        - Password
        - Profile picture
    - Necessary validations are implemented for all inputs.
    - Shows the respective errors when an invalid signup is attempted.
    - Logs the user in and redirects to homepage upon successful signup

- **Repositories**
    - Displays the repositories that are owned by the user or those in which the user is a collaborator.
    - The repository name, owner name and a brief description of the repository is displayed

- **Create Repository**
    - Allow users to create repositories.
    - Displays a form with 3 inputs at first:
        - Repository name
        - Repository description
        - Number or collaborators
    - Does the necessary validation upon form submission.
    - Redirects to another form if no discrepancies are found. Users can add the collaborators in this form.
    - Repository is created on submission of the form and the user is redirected to the Mange Repository page of the repository.
    - The names of the collaborators that don't exist is displayed on top of the redirected page

- **Manage Repository**
    - Users can add collaborators, upload files and create folders in the repository using this page.
    - Only the owners are allowed to add collaborators to a repository.

- **Code Viewer**
    - In-app code viewer to view source code
    - Change history of the file is displayed towards the right of the display screen.

- **User Settings**
  - Profile
    - User has the option of viewing and editing his profile details including his name, email, bio, connected accounts and visible elements of his/her account.
  - Account Settings
    - User has the option of changing his/her username, exporting account data and deleting and migrating the account.
  - Account security
    - This section covers the security aspects of the account on the system (passwords)
  - Notifications
    - Here, the user has the ability to choose the types of notifications (email, SMS etc) he/she receives.

3.2 Software Interfaces

- Apache or any other web server of choice can be used and information will be accepted through HTML forms and using various JS methods the basic login authentication will be conducted. This will be followed by a two-factor authorization after which the corresponding HTML page will be displayed.
- A relational or non-relational database will be used to store various information about the repository, files, commits, etc. from the user and the collaborator's through the GUI's input.
- The frontend of the website can be created using Vanilla JS and CSS or any other frontend JS framework (React / Vue / Angular).

3.3 Communications Interfaces

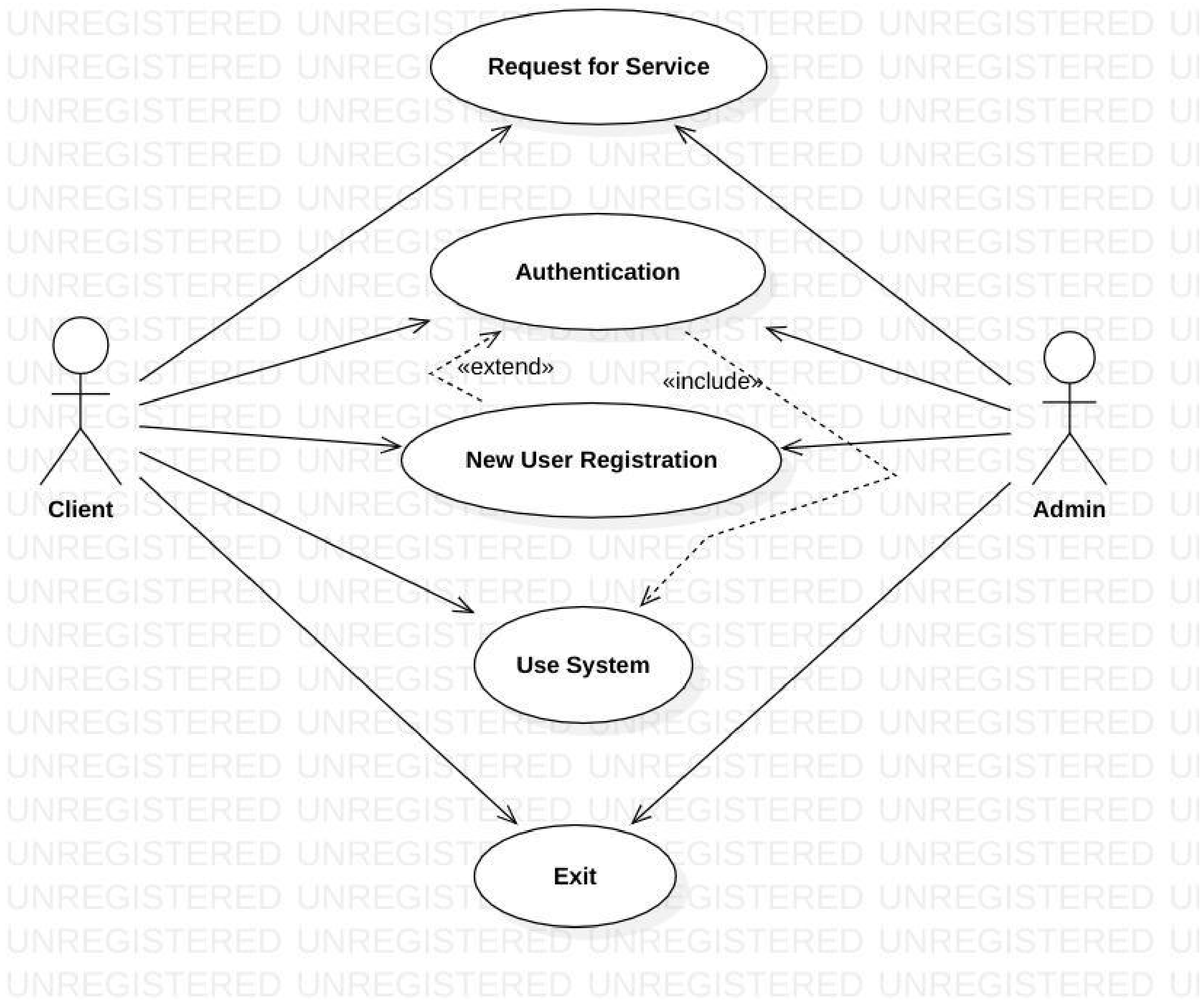
The product makes use of only one standard for communication:
- **HTTP protocol**: For web interface access and data transfer. It needs port 80 to facilitate communication at the server side.

3.4 Hardware Interfaces

a) **Server side**: A Cloud based web server with port 80 open for the web application.
b) **Client side**: A computer with a working NIC card and a connection to the Internet.

4. Analysis Models

Use case diagrams:



Services include repository creation, management, profile management, collaboration and commits etc.

5. System Features

5.1 User Authentication

5.1.1 Description and Priority

To gain access to the platform and its features, it is required that you must be authenticated. You supply or confirm credentials that are unique to you thereby proving that you are who you claim to be. The system supports the following method of authentication:

- Username and Password only

5.1.2 Stimulus/Response Sequences

- Registered User -> Redirect to login section -> Login with username, password -> Home page
- Unregistered User -> Redirect to registration section -> User is prompted to enter details -> Validate details -> Home page

5.1.3 Functional Requirements

REQ-1: Working 2FA system and login/registration page on the website
REQ-2: Email based verification for registration and login.
REQ-3: “Forgot Password” section to reset password.

5.2 Create Repository

5.2.1 Description and Priority

Repository creation is vital to a code management and version control system. Repositories are essentially containers with the concerned user’s project files and each file’s revision history. The ownership of a repository is confined to a single user. They can be one among the following two types:

- **Public Repositories:** Can be viewed by all users of the system. Restricted write access to the author of the repository.
- **Repositories with modified access:** Restricted read/write access to verified collaborators.

5.2.2 Stimulus/Response Sequences

- Upon successful authentication:
  - **Configure a new repository:** This allows the user to create an empty repository with the concerned user as the owner of the repository with access modification abilities.
- Upon choosing the required configurations for the new repository, the user is redirected into a page pertaining to the newly created repository.
- Commits to the repo are pushed to the database and the changes will be reflected in the user’s repository.

5.2.3 Functional Requirements

REQ-1: “Create repository” page should be functional
REQ-2: Repository should have a valid name (No special characters)

5.3 Search

5.3.1 Description and Priority

The search feature is essential for users to facilitate navigating the entire system with ease. The feature allows users to perform the following queries:

- Search for repositories and users.
- Search for matches in the files and the issues section within repositories.

5.3.2 Stimulus/Response Sequences

- The search field is universally accessible. The entered term(s) will be searched across the existing repositories and users to look for matches.
- Upon reception of keywords in the search field, the client sends requests to the search provider (HubSpot, Google Custom Search etc) to get corresponding responses as keyword matches (as mentioned above).
- The provider checks the database for matching keywords and further, returns the desired values to the client.
- Search results are displayed to the client.
- No results are returned if no matches are found.

5.3.3 Functional Requirements

REQ-1: Only english characters are allowed.
REQ-2: Scope of search can be specified if required. (Providers handle this automatically)

5.4 Repository Management

5.6.1 Description and Priority

The Repository Management feature allows users to configure the repo settings.

- **Access Control:** Can only be performed by the owner. It allows the owner to designate various privileges to contributors as well.
- **Basic operations & features:** Renaming, deleting or sharing repository details.
- Viewing files in the repo.

5.6.2 Stimulus/Response sequences

- The user can view repositories on his dashboard and via search.
- Clicking on one of the repository's hyperlinks redirects the user into the repository-specific overview page.
- Redirect web app to view and modify repo settings.

5.6.3 Functional Requirements:

- **REQ-1:** The User should have access to view or make changes to the repository.

5.5 Issues

5.8.1 Description and Priority

Issues are a great way to keep track of the features, enhancements, and bugs pertaining to a repo. The users can collect user feedback, report software bugs, and organize tasks Moreover, the author/contributor can also link an issue to a commit. Issues can be tagged “closed” or “pending”.

5.8.2 Stimulus/Response sequence

- An “issues” tab is made available in the repository page where users can do either of the following operations:
    - **File a new issue:** This essentially allows the user to create a new issue concerned with the repository along with an issue description.
    - **Comments:** This feature allows other users to broadcast their views concerned with the issue.
    - **Issue assignment:** This allows the contributors (with read/write access) of the repository to assign the issue to a specific user, so that he/she can work on resolving the issue pertaining to the repository.
    - **Close resolved issue:** Say, the issue has been resolved by fellow developers and thereby we no longer need to keep the issue open. This option allows the member to close the issue.

5.8.3 Functional Requirements

REQ-1: The repository should exist.
REQ-2: The issue can be opened by any contributor, but can be closed only by the users with privileges and can be assigned only by the issue creator and the privileged users.

5.6 Profile Management

5.10.1 Description and Priority

Profile Management is a feature used to organize and describe the user’s profile. Profile management is a core feature necessary to provide users full access on what information they are providing on the website. The main features are specified as follows :-

- **Account Settings and Security :-** Provides a method for the user to change username, password, enable Two-Factor Authentication(2FA) and delete the account if the user chooses to do so.

- **Repositories :-** Lists the repositories where the user has either owner/collaborator privileges.
- **Profile Details :-** Provides a summary of the user's profile.

5.10.2 Stimulus/Response Sequences

- On the profile page, we have a summary of the profile being displayed.
- To use the other features, the user goes to the edit profile wizard, where the following tabs are present :-
  - **Account Settings and Security**:- This tab is used to rename, change password, enable Two-Factor Authentication and delete the account.
  - **Repositories** :- This tab shows a list of all the repositories where the user is listed as a collaborator or owner.

5.7 Commits

5.11.1 Description and Priority

Commits are snapshots of a repository at particular instances of time. Commits have the following attributes associated with them:

- **Timestamp of creation:** When the changes were made.
- **Commit author:** The user who created the commit.
- **Commit title:** Briefly describes the changes made.
- **Commit description:** Further describes the changes made (optional).

These records are made available in the repository and can be very useful to track changes. A commit records changes made to a particular repo in this case.

5.11.2 Stimulus/Response Sequences

- The commit wizard is made available to the user in a repository which allows them to upload files to the repository thereby associating it with a commit title and commit description.
- Upon requesting to push the initialized commit, the commit history is compared with that of the new commit.
- Assuming that there was no conflict with merging the commit into the repo, the commit is pushed.
- Say there was a conflict, the merge process is aborted and an error is shown.

5.11.3 Functional Requirements

REQ-1: The user trying to commit changes must have read & write access to the repository concerned.
REQ-2: Absence of merge conflicts essential for successful merge.

5.8 Other Nonfunctional Requirements

5.9 Performance Requirements

- Response times on the website should not take >3s ideally (ignoring client side constraints) for active CRUD operations.
- Higher speeds can be achieved with better compression, optimized images and CSS and better hosting services.

5.10 Safety Requirements

It is recommended that developers keep in mind the following before and while designing the system:

- **Strengthening Access Controls:**
  - Follow the least privilege model (subject should be given only those privileges needed for it to complete its task)
  - Restrict the creation of repositories to prevent users from exposing organization information in public repositories.
  - Revoke access for all inactive users who are no longer part of the contributors.
  - Ensure users do not share GitHub accounts or passwords.
  - Ensure every contributor uses two-factor authentication on their account.
- **Storing credentials in GitHub files:**
  - Leaking secrets to repositories, either via code, config files, or commit messages, provides a gateway for attacks.
  - Initiate removal of the repository once it is realized that sensitive info has been leaked.
- **Enabling security alerts (optional feature)**

5.11 Security Requirements

- **2-Factor Authentication**
  - An additional layer of security must be provided when logging into GitHub.
- **Required reviews**
  - It must be ensured that PRs have a specific number of approving reviews before collaborators can make changes to a protected branch.

.

5.12 Software Quality Attributes

- Databases should remain consistent in case of an error.
- Optional:
  - Test-driven development approach with unit test execution and approval cycles.
  - Avoid inline and embedded styles during the creation of the web app to avoid context switching between HTML and CSS parsers.
  - Use browser-native JSON methods. Avoid XML for better application performance.
  - Style guide to be followed : JS Style Guide
- A web-based platform offers medium-high portability considering widespread internet access.

5.13 Business Rules

- **Users**
  - They retain ultimate administrative control over their User Accounts and the Content within it.
  - Functions include account creation, repository creation and modification, user access control, profile management and 2FA setup.

- **Owner of the repository**
  - The owner of a repository has ultimate administrative control over it.

- **System Administrator**
  - On the organizational side, the final authority on matters concerning feature approval and deployment lies with the system admin subject to technical and legal constraints.
  - Functions include improving the website, its performance, UI changes and changes to the database.

Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects.

- **Security Audits:** Since the code management system is offered as a SaaS, it is an essential requirement that the providers (us) undergo security audits
- **Integrated Applications and functions within the code management system**
- **For a real-world project with real-world use cases (Not required in this case):**
  - **Physical device security for devices on the backend network.**
  - **High availability:** Availability records act as a credit score for service vendors (us).
  - **Multiple, secure, disaster-tolerant data centers**

6. Other Requirements

The following requirements are relevant only for applications with a large user-base. For the purposes of the OOADSE Lab project, satisfying the minimum requirements to keep the website running should be enough.

- **Database requirements:**
  - MySQL database. No dedicated machines required.

- **Reuse policies:**
  - Information scraped, obtained via the API or otherwise can be used for the following:
    - Make sure you understand what constitutes open source software and what doesn't.
    - Scraping refers to extracting information from the Service via an automated process, such as a bot or webcrawler. Scraping does not refer to the collection of information through our API.
    - Information obtained must not be used for spamming purposes, including for the purposes of sending unsolicited emails to users or selling User Personal Information to recruiters, headhunters, and job boards.

- **Privacy:**
  - Misuse of User Personal Information is prohibited.
  - Any person, entity, or service collecting any user personal Information from the system will only use that info provided it is secured, authorized and has active feedback to complaints and removal requests.
- **Excessive Bandwidth Usage:**
  - The system's bandwidth limitations vary based on the features used. If it is determined that the bandwidth usage of a user is significantly excessive in relation to other users of similar features, the system administrator has the right to suspend the account, throttle file hosting and limit activity.

Appendix A: Glossary

- **VCS: Version Control System**
  - Version control is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.
- **SaaS**
  - **Software as a service** is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software"
- **SSH**
  - SSH or **Secure Shell** is a cryptographic network protocol for operating network services securely over an unsecured network.

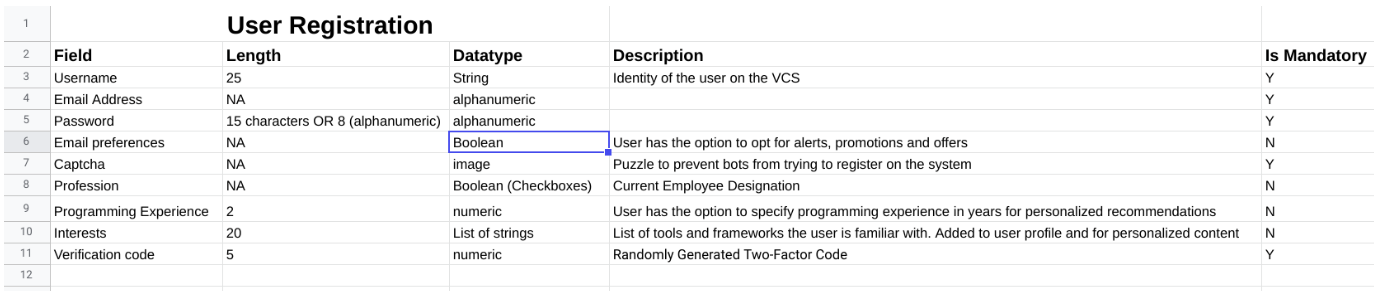
- **NIC**

    - A **network interface controller** is a computer hardware component that connects a computer to a computer network
- **2FA**
    - 2-factor authentication is an extra layer of security used to make sure that people trying to gain access to an online account are who they say they are. First, a user will enter their username and a password. Then, instead of immediately gaining access, they will be required to provide another piece of information/
- **REQ:**
    - Requirement

Appendix B: Field Layouts

Link to spreadsheet

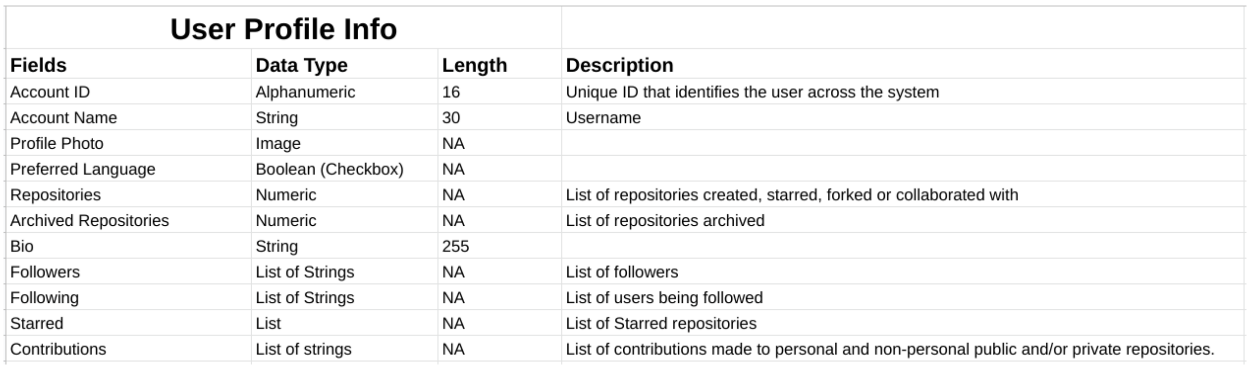
- **User registration:**



User Registration

Field	Length	Datatype	Description	Is Mandatory
Username	25	String	Identity of the user on the VCS	Y
Email Address	NA	alphanumeric		Y
Password	15 characters OR 8 (alphanumeric)	alphanumeric		Y
Email preferences	NA	Boolean	User has the option to opt for alerts, promotions and offers	N
Captcha	NA	image	Puzzle to prevent bots from trying to register on the system	Y
Profession	NA	Boolean (Checkboxes)	Current Employee Designation	N
Programming Experience	2	numeric	User has the option to specify programming experience in years for personalized recommendations	N
Interests	20	List of strings	List of tools and frameworks the user is familiar with. Added to user profile and for personalized content	N
Verification code	5	numeric	Randomly Generated Two-Factor Code	Y

- **User profile information:**



User Profile Info

Fields	Data Type	Length	Description
Account ID	Alphanumeric	16	Unique ID that identifies the user across the system
Account Name	String	30	Username
Profile Photo	Image	NA	
Preferred Language	Boolean (Checkbox)	NA	
Repositories	Numeric	NA	List of repositories created, starred, forked or collaborated with
Archived Repositories	Numeric	NA	List of repositories archived
Bio	String	255	
Followers	List of Strings	NA	List of followers
Following	List of Strings	NA	List of users being followed
Starred	List	NA	List of Starred repositories
Contributions	List of strings	NA	List of contributions made to personal and non-personal public and/or private repositories.

Appendix C: Requirement Traceability Matrix

Sl. No	**Requirement ID**	**Brief Description of Requirement**	**Architecture Reference**	**Design Reference**	**Code File Reference**	**System Test Case ID**	**Test Case Status**
1	R01	Correct Login			app.py	UT_2	Pass
2	R02	Incorrect Login			app.py	UT_3	Pass
3	R03	Register into the system			app.py	UT_1	Pass
4	R04	Create Repository			app.py	UT_5	Pass
5	R05	Search through your repository and other public repositories			app.py	UT_9	Pass
6	R06	Managing accesses to the repository(Owner)			app.py	UT_20	Pass
7	R07	Commit; without conflict			app.py	UT_11	Pass