Software Requirements Specification

for

Code Management and Version Control System(Github)

Version 1.0 approved

Prepared by

A R Prashanth PES1201800410

R Sree Harsha Reddy PES1201800233

Nikhil N Gowda PES1201802023

PESU University

04-Feb-2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1

1.2 Purpose

Source code management (SCM) is critical to keep a record of the changes made to the source files. These systems will be up-to-date with the information of 'who' changed 'what' and 'when' did the change happen. Version control is the way to manage the source code effectively and keep track of the changes made to the source code. It is a principle employed to track, document and monitor every piece of code written i.e. it is used to manage changes that occur with every code commit. This method helps developers track the code, especially when multiple people are contributing at different points in time. This also helps roll back to the earlier version if there are any errors.

Version Control. These two terms are used interchangeably. However, source control is specific to source code**. Version control** also covers large binary files and digital assets.

1.3 Intended Audience

This document is intended for software developers,users , testers and documentation writers.This SRS helps the intended users to develop and improve the product further.The report includes the specific overview and diverse use cases of the product.This archive fills in as a guide for the designers on one hand and a product approval record for the planned client on the opposite.

1.4 Product Scope

Version control (also known as **revision control**, **source control**, or **source code management**) is a class of systems responsible for managing changes to computer, documents, large web sites, or other collections of information.It will perform basic functions that will permit the client record changes to a file or a series of files over time so that specific versions can be recalled later.The benefits and features of version control are mentioned here https://git-scm.com/about.

1.5 References

https://git-scm.com/about.
https://en.wikipedia.org/wiki/Version_control
https://www.slideshare.net/AdityaNarayanSwami/githubsource-code-management-system-srs
https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

2. Overall Description

2.1 Product Perspective

Our product is a distributed version-control system for tracking changes in any set of files, designed for coordinating work among programmers cooperating on source code during software development.Its goals include speed,data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems).It also provides cloud based hosting service for the repositories .

2.2 Product Functions

- **Frictionless Context Switching.** Create a branch to try out an idea, commit a few times, switch back to where you branched from, apply a patch, switch back to where you are experimenting, and merge it in

- **Role-Based Codelines.** Have a branch that always contains only what goes to production, another that you merge work into for testing, and several smaller ones for day to day work

- **Feature Based Workflow.** Create new branches for each new feature you're working on so you can seamlessly switch back and forth between them, then delete each branch when that feature gets merged into your main line.

- **Disposable Experimentation.** Create a branch to experiment in, realize it's not going to work, and just delete it - abandoning the work—with nobody else ever seeing it (even if you've pushed other branches in the meantime).

2.3 User Classes and Characteristics

Users of this code management and version control system will mainly be software developers. Since it is reasonable to assume that an average developer has knowledge about functionalities and usage of Version control,we assume that our users will already be informed about basic functionality of the product. Also clear documentation and tutorials about the product feature will be provided.

2.4 Operating Environment

The hardware platform for implementation may consist of modern electronic devices with internet which includes laptops, desktops,Remote Servers with Unix/Linux, windows and macOS.For accessing the hosting service Web browser is required.

2.5 Design and Implementation Constraints

Git is very fast, scales very well, and is very transparent about its concepts. The down side of this is that it has a relatively steep learning curve. Git exposes hashes as version numbers to users; this provides guarantees (in that a single hash always refers to the exact same content; an attacker cannot modify history without being detected), but can be cumbersome to the user. Git has a unique concept of tracking file contents, even as those contents move between files, and views files as first-level objects, but does not track directories.Uses CLI so might be difficult for users who use GUI frequently.

2.6 Assumptions and Dependencies

- The user must have basic knowledge about the product.
- Internet connection is required to access hosting service.
- Each User must have a User ID and password.
- Basic knowledge of CLI.

3. External Interface Requirements

3.1 User Interfaces

CLI for local version control. We basically aim to provide a Unix command line interface which most of the developers prefer.Our hosting website(similar to github) is GUI which helps in hosting the repositories and synchronising work done by various developers.

3.2 Software Interfaces

- HTML5:HTML5 is the fifth revision of the HTML standard
- CSS: Cascading style sheets decide the styling rules of the website in separate file
- Javascript:JavaScript is a lightweight, object-oriented, cross-platform scripting language, mainly used within web pages.
- C: C is a programming language .It is used to build the local version control.
- Database management system(TBD)

3.3 Communications Interfaces

- We use HTTPS or SSH protocols to connect from local version control to the hosting service.

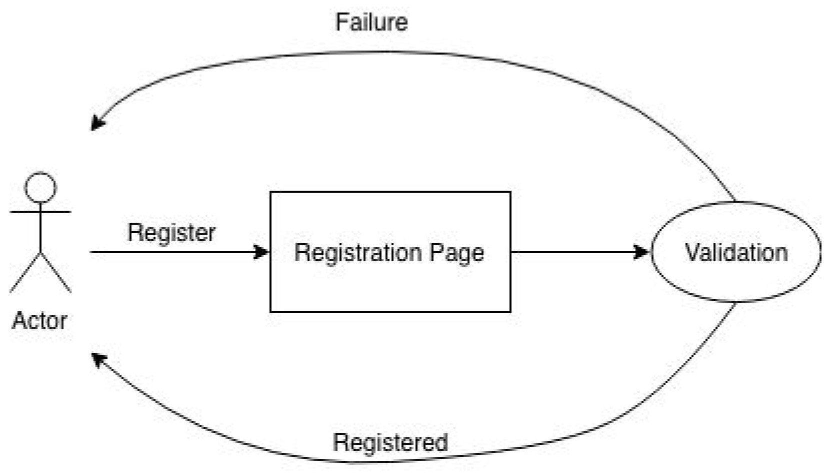
- The user information on the database contains the email address of the user to send various updates.
- We store the encrypted hash of the password in our database.

4. Analysis Models

<Include pertinent analysis models, such as use case diagrams and if applicable entity-relationship diagrams.>

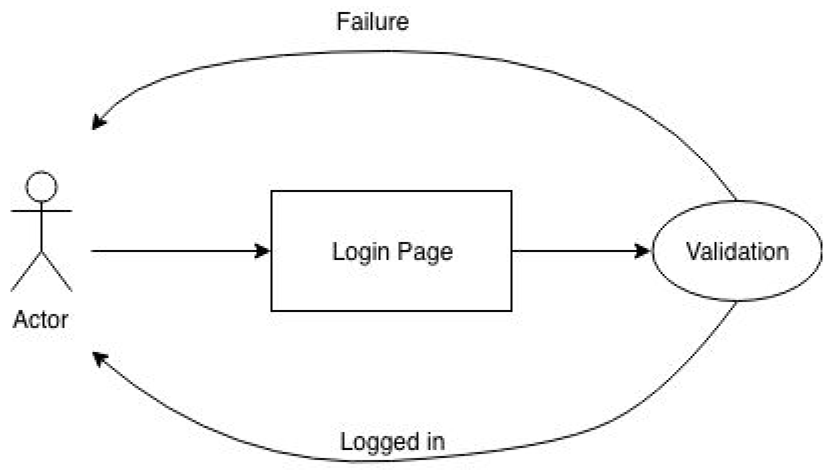
User authentication

registration of a user into github



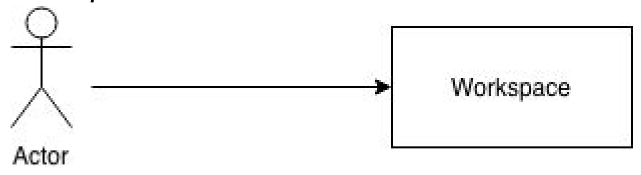
Users have to register in order to have his workspace in github cloud .

Login



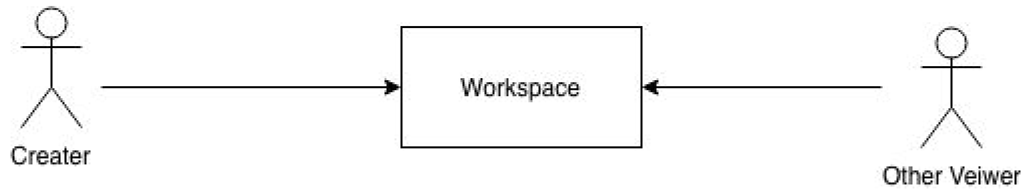
users can login via login page , users are not allowed to upload or edit source code until they validate themselves by login in with write username and password

workspace:



workspace is generated when an valid user creates a repository through graphical interface

Viewers:



anybody on the internet can view the workspace only if the user has made it public i.e. everyone can see .

5. System Features

In this section we can describe diverse gadget capabilities of the product. Which can be CLI support , workspace , code editor and authentication control . The primary purpose of this segment is to offer documentation on system features and the interfaces interplay with the software with external entities in detail

Workspace explorer and the code editor interface may be the main graphical person interface in which builders will interact with their workspace and its consisting of. It's the location where most of the interplay takes place . The user could be supplied with a graphical interface where they are able to create , edit and delete files . They are able to have a look at the distinction among the previous commits and review from the previous commit . Customers are provided with a choice to create branches or merge existing branches to the current version of source code .

As linux/unix is becoming more and more popular CLI support is very useful for automating things . in CLI support we basically aim to provide a UNIX like command line interface which is normally preferred by the software developers . With this CLI users will be able to run UNIX commands on their workspaces under the privileges given to them.

As linux/unix is turning into increasingly popular , cli aid is very useful for automating matters . In cli support we essentially aim to provide a unix like command line interface that is typically desired by using the software developers . With this cli users will be able to run unix instructions on their workspaces below the privileges given to them.

5.1 Workspace explorer

5.1.1 Description and Priority

The user will be provided with a UI interface where he can publish his content in his workspace . workspace explorer has the highest priority as most of the users prefer using UI then command line interface .Users will be provided with a page to register and login to their respective workspace and their workspace or source code will be stored in the cloud.

5.1.2 Stimulus/Response Sequences

Users should have an account on github . If not he/she/they can create a github account by going to the registration page . login to the account to access their workspace . users are allowed to upload any kind of document with a specific size limit TBD.

5.1.3 Functional Requirements

1. user authentication : users should have a valid account in github or should create a new account in order to use github services . Every user will be authenticated before they can use their workspace.
2. source code: every user will be allowed to keep multiple versions of the code by default. users will have edit access on their source file .

5.2 CLI support

5.1.1 Description and Priority

CLI support is important because most software developers prefer using command line interfaces rather than graphical user interfaces . it should support features like push , pull , commit , add TBD.

5.1.2 Stimulus/Response Sequences

- ❖ Authentication has to be done before pushing into a github from the command line tool .
- ❖ The purpose of the tool is to make the user's job easier . Users can automate it by creating a script etc .

5.1.3 Functional Requirements

- add : add all the provided files into the workspace if they meet the requirements TBD.
- push : upload the workspaces with changes made locally to the cloud .
- pull : get changes make in the cloud to local drive
- commit : finalise a version by committing it which will be recorded for further process like debugging or code reviewing .
- clone : download a complete workspace to the local drive .
- branch : create a parallel workspace .
- merge : merge the branched workspace with the parent workspace.

6. Other Nonfunctional Requirements

In this section,non-functional requirements are defined in elements.Nonfunctional requirements are vital to the success of software.These requirements include performance requirements,safety requirements,security requirements,software quality attributes and business rules.

6.1 Performance Requirements

- As this software program is internet based,it does require a powerful server device with high band net access.Server device needs to have an effective CPU and high speed internet so that it can handle multiple users simultaneously without any additional latency.

- There have to be sufficient no of servers to address excessive traffic.And good scheduling among them will enhance the performance.
- Performance requirement from the client side is that application to be developed as a lightweight web app so that it can work on all platforms.
- Capacity:
  - The software must be able to handle at least min of 100 users at a time.
- Response Time:
  - The load time for user interface screens shall take no longer than five seconds.
  - The log in information shall be verified within ten seconds.
- User-Interface:
  - Queries shall return results within ten seconds.

6.2 Safety Requirements

- The application must ensure that it leaves files untouched.No modification is allowed to the files uploaded by the user.
- The user should keep his private keys(password) safe and secure and the same should not be shared with anyone.
- The maximum size of the file is set to lessen the load on the server.A user cannot be able to upload a file whose size exceeds this threshold.

6.3 Security Requirements

- Information given by the users at the registration page is kept at the system database.The product must ensure the privacy of users data.This information can not be reached by other users or external threats.
- The personal information of users must be stored in encrypted form.
- Users will have to mandatory login in the system with a unique userID and a password.
- The system must automatically log out users after a period of inactivity.
- Workspace of the user should only be accessed through the user's own credentials and any other should not be able to access the user's private data.
- All exchanges from client to server involving private data shall occur using the highest available level of secure connection (e.g., https)

6.4 Software Quality Attributes

- ***Usability***
  - The application provides a pleasant and user friendly graphical interface with relatively simple functions.
- ***Maintainability***

- Maintenance is one form of change that typically is done after the software development has been completed.As the time changes,so do the needs.The system must be able to adapt accordingly without compromising performance.

- ***Reliability***
  - Servers might be damaged to various motives that may consist of natural disasters,power failures etc.The system designed have to include fault tolerance systems via replication of data.
- ***Robustness***
  - The system design shall include recovery scenarios allowing the ability to restore a state in minimum time possible.
- ***Availability***
  - The application will run 24x7 .

6.4.1 Business Rules

- ❏ Only owners of the repository have the rights to review and manage pull requests.
- ❏ Access control to the repository is maintained and managed by the owner of the repository.
- ❏ Creation and deletion of repositories can only be done by the owner of the repository.
- ❏ A user has a right to decide whether the repository has to be made public.
- ❏ All authenticated users can access public repositories.
- ❏ Domain requirements
  - ❏ All the changes made by each contributor should be tracked.
  - ❏ The file(source code) is of utmost concern and must be protected against every possible failure.

7. Other Requirements

Appendix A: Glossary

Term	Definition
VCS	Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

SCM	Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors.
HTTPS	Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet.
Database	A structured set of data held in a computer, especially one that is accessible in various ways.
Repository	A directory or storage space where your projects can live.
Clone	To clone a repository means to duplicate and download everything in the repository.
Commit	A commit, or "revision", is an individual change to a file (or set of files)
Interface	Interfaces are the languages, codes and messages that programs use to communicate with each other and to the hardware
GUI	A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user.
CLI	A command line interface (CLI) is a text-based user interface (UI) used to view and manage computer files.
SSH	Secure Shell(SSH) is a network communication protocol that enables two computers to communicate (c.f http or hypertext transfer protocol, which is the protocol used to transfer hypertext such as web pages) and share data.

Appendix B: Field Layouts

User Registration

Field	**Description**	**Length**	**Data Type**
Email ID	To store the mail id of user	254	Alphanumeric Special char @ is allowed
Username	Stores the username of user.Helps in uniquely identifying a user	40	Alphanumeric

Password	To store password required for authentication	min 8 max 16	Alphanumeric and special chars allowed

Repository Info

Field	Description	Length	Data Type
Name	Stores the name of repository	40	Alphanumeric
Branch name	stores the name of branch	40	Alphanumeric
Filename	Stores the name of file	40	Alphanumeric
File Description	Describes the content of file in brief.	72	Alphanumeric

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID
1	RQ-01	User Registration					
2	RQ-02	User Authentication					
3	RQ-03	Profile Page					
4	RQ-04	Search bar					
5	RQ-05	Create/Delete Repository					
6	RQ-06	Pull					
7	RQ-07	Pushing changes					
8	RQ-08	Clone					
9	RQ-09	Revert Changes					
10	RQ-10	Max File size					