Software Requirements Specification

for

Code Management and Version Control System

Code For All

(based on Github)

Version 1.0 approved

Prepared by

Darshan D (PES1201801456)

Ameya Bhamare (PES1201800351)

Drishti Hoskote (PES1201801283)

PES University, Bangalore

06/02/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Darshan D	06/02/21	First Version	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to capture the description and requirements of a Code Management and Version Control System named Code For All, based on GitHub. It includes a description of the functionalities that are required to be a part of the software. The description is accompanied by a synopsis of all the features that the software entails. Relevant details about the frameworks and interfaces are provided to elucidate each functionality. It also includes the non functional requirements that are intended to be a part of the software.

1.2 Intended Audience

This document is intended for the developers who will end up implementing this software and the professors at PES University who will act as the testers and the quality assurance persona.

The rest of the document contains the following information:

- **Description of the Product:**
  - Product Perspective explaining the need for such a product
  - The major functions that the product can perform
  - The types of user classes expected to use the product
  - The environment in which the product will operate
  - Design and Implementation constraints
  - Assumptions and dependencies

- **External Interface Requirements:**
  - Characteristics of the user interface
  - Details about the Software interfaces
  - Details about the communication interfaces

- **Analysis Models:**
  This includes the use case diagrams and the entity relationship diagrams

- **System features:**
  This lists out all the system features that should be a part of the software. Under each feature:

  - The feature is described in detail
  - The sequence of user actions that needs to be performed to bring the feature into action
  - Functional requirements needed to implement the feature

- **Non Functional Requirements:**
  - Performance Requirements
  - Safety Requirements
  - Security Requirements
  - Software Quality Attributes
  - Business Rules and Domain requirements

- **Appendix A: Glossary**
  This lists out all the terms necessary to properly interpret the SRS

- **Appendix B: Field Layouts**
  This contains the field layouts and the properties/attributes and the report requirements

- **Appendix C:** Requirement Traceability Matrix

1.3 Product Scope

This software is a source code repository hosting service. It provides a convenient way to manage source code and develop production level software. It allows for collaboration among developers across teams. Importantly, it allows for distributed version control. It provides a hassle free experience in the form of a web based GUI. It promotes open source culture by allowing developers to add patches. Patches are small files that indicate changes made in a repository. This promotes the availability of good code among developers.

1.4 References

In our case, references are not applicable.

2. Overall Description

2.1 Product Perspective

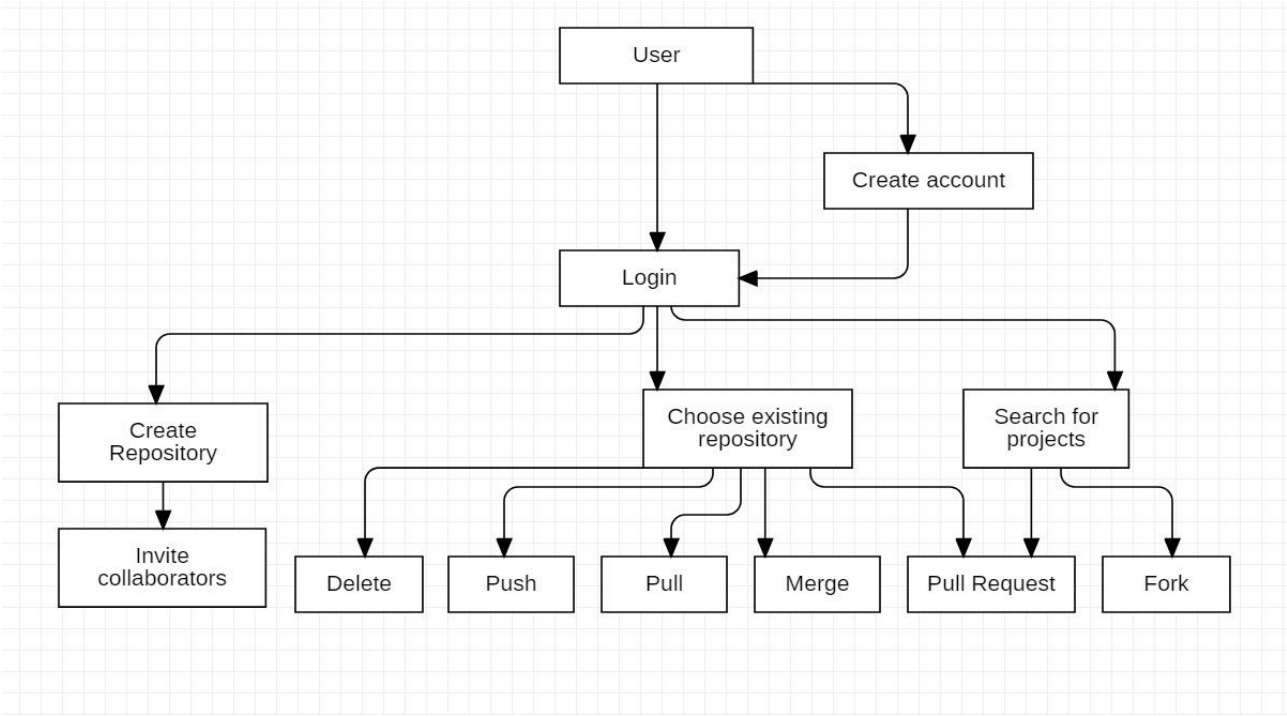
During the initial years of product and source code development, collaboration was not easy with teams across the world. Having large codebases is common in software development. Managing different versions of it was a tedious process. By means of our solution, we plan

to mitigate these issues by providing an easy to use platform for source code management. This will change the way software is developed henceforth.

2.2 Product Functions

- Create a user account
- Optionally, allow them to create a profile
- Create a new repository
- Push i.e. add files / folders to the repository
- Clone / Fork a repository
- Create a branch
- Checking out a branch / Commit
- Merge an existing branch
- Create a pull request to suggest changes to a repository
- Star an interesting repository
- Revert any unwanted changes

Top Level Data Flow Diagram



2.3 User Classes and Characteristics

We have primarily identified three class of users :

- **Developers from academia**
  This includes students, professors and lab instructors who will use the software for personal projects and research purposes. They are not likely to develop production level software.

- **Developers working in the corporate world**
  This includes software developers, testers, quality assurance engineers and project managers who are involved in the software development life cycle of production level software.

- **Developers working on open source projects**
  This includes freelancers or organizations that want to make good quality code available to everybody free of cost. They can develop a wide range of projects. From simple projects for representational purposes all the way up to large software solutions. Take for instance, Netflix's repository that has been maintained open source to benefit developers across the world.

2.4 Operating Environment

The said software will run as a web service hosted on a server. Any computing device connected to the Internet will be able to avail the services. It will support Linux, Windows, MAC, and Solaris among other operating systems. On the server side, we will need to avail the services of a cloud service provider to host our website, to store and manage all the repositories on the cloud.

2.5 Design and Implementation Constraints

Non-availability of a stable internet connection will hinder effective usage of this software. As academicians, we aren't bound by corporate or governmental policies. The web interface will be built using standard web technologies like HTML, CSS, and Javascript. The functionalities will be implemented using one or more object oriented programming languages like C++. As our website will be hosted by the cloud service provider, their infrastructure for storage will be used. The communication protocols that will be used include HTTPS, FTP and TCP to enable secure and reliable transfer of data over the network. Keeping in mind security issues, we have identified two primary classes. The first one being the corporate world where the intellectual property involved requires the highest

levels of security. The other one being the world of academia which mostly involves personal projects not requiring as high a level of security.

2.6 Assumptions and Dependencies

We assume that the cloud service provider we plan on using complies with the policies and security measures adopted while developing our project.
We assume that the cloud service provider does not sell the sensitive data of the repositories hosted on our website to any third party companies for their monetary benefits.
We assume that the user of our software has a stable internet connection to be able to access and use our web service.

3. External Interface Requirements

3.1 User Interfaces

- To start off with, we have a sign-up / sign-in interface that requests for username, email address, password and other necessary user details.
- On signing in, the user is led to their homepage where they can see a list of their public and private repositories.
- Besides that, they can navigate to edit profile, settings and other accounts.
- By other accounts, we are referring to other users who avail the services provided by the software. One can choose to collaborate with multiple users on a repository.
- On entering a repository, one can see the list of contributors, commit history, commit messages, issues, pull requests and files that make up the project.
- Each repository has its own readme to give an idea as to what it is about.
- There is a sidebar that indicates the percentage of each language used. For instance, 10% Python, 20% Java and so on.
- In clickables, we have buttons for starring, cloning, forking and downloading a repository.

3.2 Software Interfaces

On the back-end of our software, it will be connected to the software components like databases, development, debugging environments and other such tools that we have enabled as services from the chosen cloud service provider.
The data items or messages that are coming into our system will include the files being uploaded as part of the user's repositories. It also includes the user information entered to create a user profile.

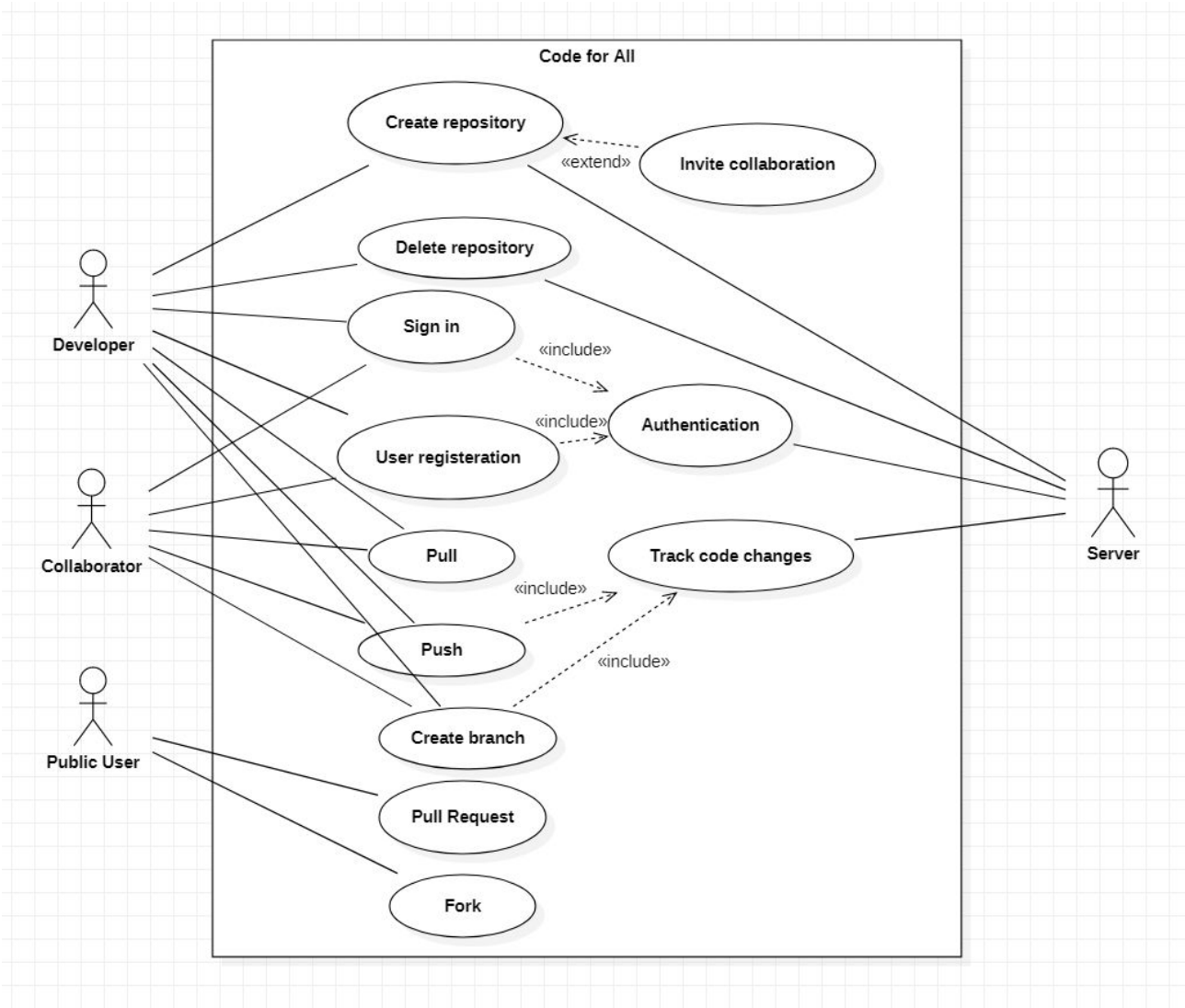
The messages going out will include the status messages. These signify the status of the operations that the user performs on his/her repositories. Outgoing messages also include the files of the hosted repositories itself. The users should be able to view the contents of their repositories.
The services needed will include the compute, network and storage services that we will be using from a cloud service provider.

3.3 Communications Interfaces

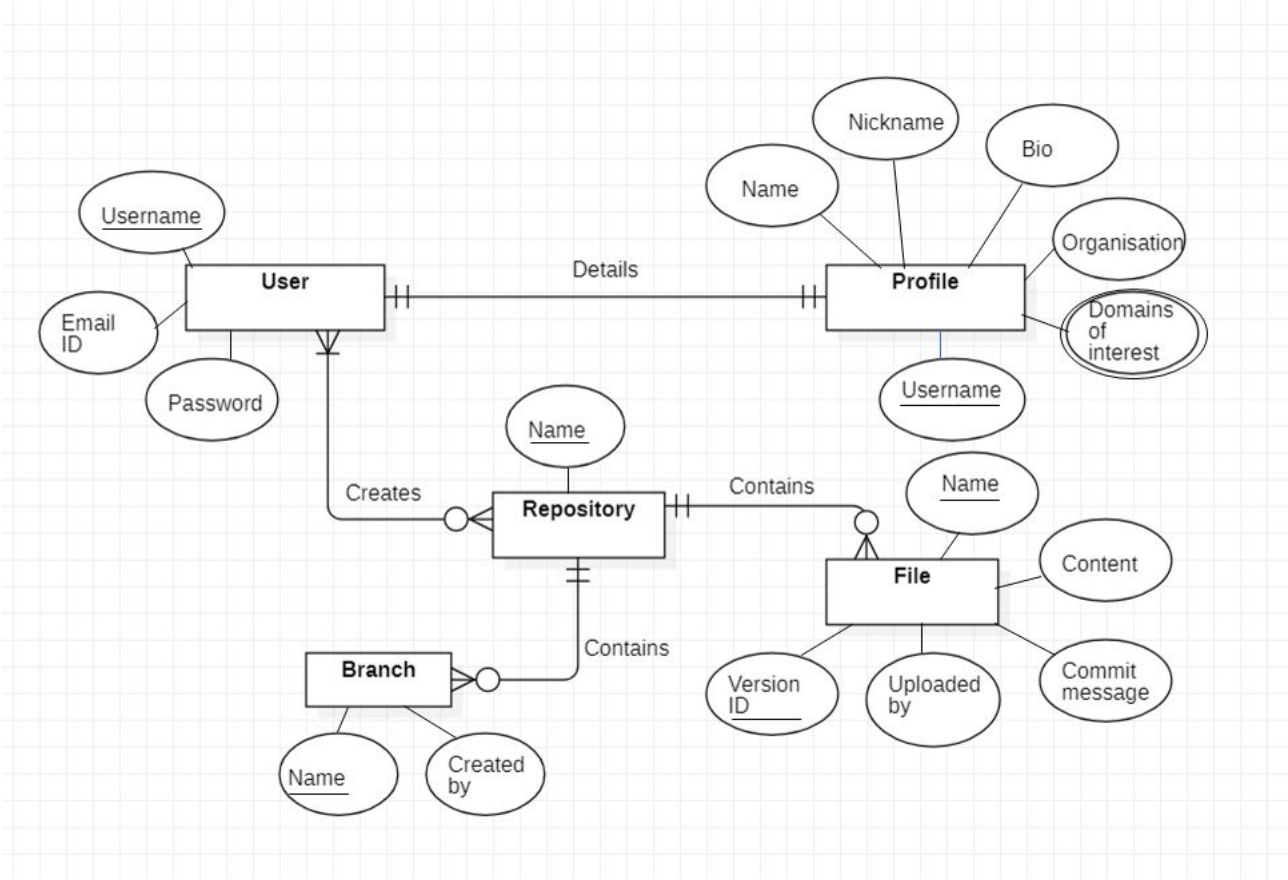
The communication requirements include a web browser on the client side to access the website, network server communication protocols such as HTTPS and FTP to transfer files safely and reliably, forms to collect user data to build his/her profile. There is a requirement to encrypt sensitive data that is being hosted privately on the website for collaboration and version control purposes but not for public view. The data transfer rates should be quite high considering the large amounts of data being transferred and to enable fast and easy access for the users to make it worthwhile for them to host their repositories on the website rather than housing it locally. There has to be well defined synchronization mechanisms in place to ensure proper collaboration among the developers.

4. Analysis Models

Use Case Diagram



ER Diagram



5. System Features

5.1 Creating a repository

5.1.1 Description and Priority

As the name suggests, this feature allows a user to create a new repository.
A repository encompasses the entire collection of files and folders associated with a project, along with each file's revision history.
It is a high priority feature as it is the placeholder for all the content that a user would like to store.

5.1.2 Stimulus / Response Sequences

- A GUI that will provide an interface to create a repository at the click of a button.
- It will lead to a page that has customization options like repository name, owner name, option to add collaborators, privacy of the repository to name a few.
- Option to enable a readme associated with the repository.
- An ignore file that allows the user to list all the file types that are not permitted in the
- repository.
- A license, if and when applicable.

5.1.3 Functional Requirements

1. A web interface with a clickable button to create a repository.
2. A form to collect all the customization inputs that the user desires.
3. The user requirements will have to be sent over a network using the appropriate communication protocols, to the data center where the database is housed.
4. Based on the user's requirements, entries into the corresponding tables of the database will be made.
5. Possible Error Scenarios :
    a. If a repository name is being re-used by a user, an error message will be displayed. They will be asked to rename it to something else.
    b. If there is loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    c. If the name entered for a repository doesn't adhere to the naming standards, the user is prompted to change the name.

5.2 Push

5.2.1 Description and Priority

This feature allows us to add content to a repository. One can upload existing files from their local storage. It allows the user to update the remote repository hosted on our website with the changes made to the local version of their repository.

It is a high priority feature because nothing can be done unless a repository has some content in it. This is the only way to add and update the content of a repository.

5.2.2 Stimulus / Response Sequences

The GUI will have a button to push any changes that the user wants to make to the repository. It can include changes to code, newly created files or folders, and so on.

On clicking the button, the user will be shown another interface where they will be asked to upload the necessary files and folders. The user will also be asked to enter any additional details like commit messages, version information and other such

things. Then the GUI will have a confirm button to finalize the changes the user is trying to make.

5.2.3 Functional Requirements

1. A web interface with a clickable button to push changes to the repository.
2. A form to collect the list of files and folders that the user wants to push to the repository.
3. Based on the files selected by the user, appropriate entries will be made into the database.
4. Ability to store files of different types should be handled.
5. Sufficient information about the version should be maintained to provide the distributed version control among the different collaborators of the repository.
6. The list of files will have to be sent over a network to the data center where the database is housed.
7. Possible Error Scenarios :
    a. If there is a difference in version between the user's local version of the repository and the most recent version of the remote repository, the user will be prompted to pull these changes. Following this, they can proceed to push their updates.
    b. An invalid file type based on the list of allowed file types for a particular repository as mentioned in the ignore file.
    c. If there is loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try uploading their files again.
    d. If regulatory compliance standards are being violated as in the case of a corporate setting, an error will be raised.

5.3 Pull

5.3.1 Description and Priority

Pull is used to update the local version of the user's repository based on the contents of the remote repository hosted on the website.

It is a high priority feature as it allows a user to update their local line of development based on the changes made by the collaborators on the remotely hosted repository.

5.3.2 Stimulus / Response Sequences

The GUI will have a button to pull the remote version of the repository onto the user's local storage. This will cause the current version of the remotely hosted repository that the user is viewing to be pulled and be updated on the local version of the user's repository.

5.3.3 Functional Requirements

1. A web interface with a button to pull changes made to the repository.
2. The required files will have to be retrieved from the database, both efficiently and accurately.
3. These files will have to be transferred over the network to the user's system so that the user can download and update these changes to the local version of the repository.
4. Possible Error Scenarios :
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    b. If the local version of the user's repository matches the remote version of the repository, the user will be prompted that their local version is already up to date and a redundant pull operation is unnecessary.

5.4 Create a Branch

5.4.1 Description and Priority

Branch feature enables the user to create a parallel line of development other than the main one. The user can make changes here without affecting the main line of product and software development. This is a high priority feature as this enables the user to test out new features, debug and make other prospective changes without breaking the existing code.

5.4.2 Stimulus / Response Sequences

The GUI will have a button to create a new branch. On doing so, a dialog box is shown where the user is asked to provide additional details. The user will have to mention the point in the directed acyclic graph of versions from where he/she would like to branch out and create a new line of development. Then the user will have to give an appropriate branch name and version info. There will be a confirm button to finalize the changes that the user is trying to do.

5.4.3 Functional Requirements

1. A web interface with a button to create a branch.
2. Appropriate information about the versions and point where the branch is made will have to be maintained at the back end to ensure that the user's requirements to create an alternate line of development are met.
3. A form to collect all the options and details that the user mentions in the prompted dialog box.

4. The entered details will have to be transferred across the network to the databases of the data centres of the chosen cloud service provider using the appropriate communication protocols.
5. These details will have to be stored in the appropriate tables of the database.
6. Possible Error Scenarios:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    b. If the naming of the branch does not adhere to the naming standards, the user will be asked to change the name of the branch
    c. If the branch name matches that of an already existing branch, the user will be asked to rename the branch.

5.5 Checking out a Branch / Commit

5.5.1 Description and Priority

This feature lets a user to navigate between branches / commits created by another user.
Checking out a branch / commit updates the files in the working directory to match the version of the files stored in that branch / commit.
It asks that all new commits on that particular branch / commit be updated. It is a way to select which line of development a user is working on.
Multiple developers can work on a single piece of software, each making their own changes in a protected way, without adding unstable code to working software.
It makes it easy to review and collaborate with others in a failsafe way.
It is a high priority feature as this enables the user to switch between different lines of development and also allows the user to revert to older, more stable versions of the code if such a situation arises.

5.5.2 Stimulus / Response Sequences

The GUI will allow a user to view and navigate between several existing branches or previously made commits by the same / different user(s).
Based on their requirements, they can choose a particular branch or commit.
A confirm button will be provided to ensure that the user is checking out the right branch.

5.5.3 Functional Requirements

1. A web interface to view the list of available branches and select the one to be checked out.

2. All details i.e. a brief description for each of the available branches need to be made easily visible for the user to make a choice.
3. A dialog box that will allow the user to give a name to the branch being checked out for the sake of convenience.
4. The user's choice along with the associated information needs to be transferred across a network for the checkout to take place successfully.
5. Possible Error Scenarios:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.

5.6 Merge

5.6.1 Description and Priority

The Merge feature enables the user to merge different branches. This is a high priority feature as it allows the user to merge the code developed in parallel lines of development. So once a particular feature has been tried and tested out in a particular branch, it can be merged with the main branch.

5.6.2 Stimulus / Response Sequences

The GUI will have a button to merge existing branches of a repository. On clicking the button, a dialog box appears where the user has to choose the branches that he/she wishes to merge. Then there is a confirm button to finalize the changes that the user is willing to do.

5.6.3 Functional Requirements

1. A web interface to view the available branches and choose the ones to be merged.
2. The changes to be made on merging the branches has to be communicated via the network using the appropriate communication protocols to the back end database that is housed at the data centre of the chosen cloud service provider.
3. Appropriate changes will have to be made in the backend database based on the branches being merged. Care has to be taken to ensure that the right versions of the file are being merged.
4. Possible Error Scenarios:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    b. If the files being merged have segments of code that have been removed or changed as compared to the version of the files in the other branch, merge conflict error will occur as there is no defined way to

merge these files. In such a case, the user will be asked to manually handle the conflicts to enable merging of the branches.

5.7 Cloning a repository

5.7.1 Description and Priority

This feature allows one to copy a repository as-is from the remote server to their local computer. It makes it easier to add or remove files, and push larger commits. Cloning a repository pulls down a full copy of all the repository data that the remote repository has at that point in time, including all versions of every file and folder for the project. This is a medium priority feature as this enables a user to clone and use public repositories of other developers.

5.7.2 Stimulus / Response Sequences

The GUI will have a button to clone an existing repository. On clicking the button, a confirm button will pop-up. On clicking confirm, the repository will start downloading onto the local machine.

5.7.3 Functional Requirements

1. The general interface of the software should feature a clone button for every existing repository.
2. On clicking the clone button, all the files and folders associated with that repository must be transferred from the cloud storage, over the network and to the local machine of the user.
3. Possible Error Scenarios:
    a. If there is loss of connectivity due to an unstable internet connection, the clone will fail and the user will be prompted to click the button again.
    b. If the user tries to clone a private repository, the button will be blurred indicating that it is a private repository and thus cannot be cloned.

5.8 Pull Request

5.8.1 Description and Priority

This feature enables the user to make a request to a public repository owned by other developer(s), to update the changes the user is proposing in the remote version of the repository hosted on the website. This a medium priority feature.

5.8.2 Stimulus / Response Sequences

The GUI will have a button to make a pull request to the owners of the public repository. On doing so, a dialog box pops up asking for further details. Here the user is asked to enter the issue that he/she is trying to solve, the solution the user is proposing and the changes that he/she is requesting to be committed. Then there is a confirm button to finalize the changes that the user is willing to do.

5.8.3 Functional Requirements

1. A web interface with a pull request button that leads to a dialog box asking for further details.
2. A form to collect the data that the user enters in the dialog box.
3. The collected data has to be communicated via network to the database housed in the data centres of the chosen cloud service provider using the appropriate communication protocols.
4. If the owner of the repository accepts to enforce the changes proposed by the pull request, updates will have to be made to the databases.
5. Possible Error Scenarios:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    b. If the owner of the repository does not feel that the changes proposed in the pull request solve the issue being addressed, he can choose to disregard the pull request. In such a case, the user will be prompted with the appropriate message along with any comment that the owner mentions.

5.9 Fork a repository

5.9.1 Description and Priority

A fork is a copy of a repository. This feature allows one to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for one’s own idea.

5.9.2 Stimulus / Response Sequences

The GUI will have a button to fork an existing repository. On clicking the button, a confirm button will pop-up. On clicking confirm, the repository will start downloading into their account and not the local machine as in the case of cloning a repository.

5.9.3 Functional Requirements

1. The general interface of the software should feature a fork button for every existing repository.
2. On clicking the fork button, all the files and folders associated with that repository must be made available in the user's account.
3. Possible Error Scenarios :
    a. If there is loss of connectivity due to an unstable internet connection, the fork will fail and the user will be prompted to fork the repository again.
    b. If the user tries to fork a private repository, the button will be blurred indicating that it is a private repository and thus cannot be forked.

5.10 Difference between versions

5.10.1 Description and Priority

This feature enables the user to check out the differences between the current version of the file the user is viewing as compared to the version of the file in a particular commit (snapshot) at a previous point in the line of development. This is a medium priority feature as this enables the user to analyse the updates that have been made across commits and could help to figure out the reason why a particular error started occurring or why an already implemented feature stopped working as desired.

5.10.2 Stimulus / Response Sequences

While viewing the contents of a particular file in the repository, the GUI will provide a button to check out the difference between the current version of the file and the version of the file at a previous point in time. On clicking the difference button, a dialog box will pop up that shows the history of commits for that particular file. Here the user can choose the commit version that he would like to compare the current version of the file with. On doing so, the interface will show the older version of the file, the newly added segments will be shown in green while the newly removed segments will be shown in red.

5.10.3 Functional Requirements

1. A web interface with a clickable button for showing the difference between versions of the file.
2. On clicking the button, a dialog box will have to be shown where the user can choose the version of the file that he would like to compare the current version of the file with.

3. The collected details about the versions will have to be transferred via the network to the back-end using the appropriate communication protocols.
4. The required versions of the file will have to be fetched from the back-end database.
5. The different versions of the file will have to be compared and the differences should be shown in a convenient manner on the front end interface.
6. Possible Error Scenario:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.

5.11 Star

5.11.1 Description and Priority

This feature allows a user to "star" i.e. mark repositories and topics to keep track of projects they find interesting and discover related content in their feed.
Starring makes it easy to find a repository or topic again later.
One can see all the repositories and topics they have starred on their star page.
This is a low priority feature since one can always search for relevant repositories in the search bar. However, it is a convenient feature to possess as it allows for better user experience.

5.11.2 Stimulus / Response Sequences

On each repository's page, there will be a button to "star" that repository. On clicking it, the repository will be added to their star page.

5.11.3 Functional Requirements

1. The general interface of the software should feature a "star" button for every repository.
2. On clicking the "star" button for a particular repository, the entire repository must be made available in the user's star page.
3. Possible Error Scenario :
    a. If there is loss of connectivity due to an unstable internet connection, the star will fail and the user will be prompted to star the repository again.

5.12 Revert changes

5.12.1 Description and Priority

This feature enables the user to revert the updates made by the user in a particular commit. The user can revert to the version of the file before these changes were made. This is a low priority feature as this can also be done by checking out a previous commit in the history of development and the user can start developing from that point again.

5.12.2 Stimulus / Response Sequences

The GUI will provide a button to revert the changes made in a particular commit. On clicking the button, the user will have to give additional details such as the commit whose changes he/she is willing to revert, the specific updates that should be reverted. Once the necessary details are given, the interface provides a confirm button asking the user to finalize the changes that he/she is willing to make.

5.12.3 Functional Requirements

1. A web interface with a clickable button that reverts the changes.
2. On clicking the button, a dialog box should pop up where the additional details are collected.
3. A form to collect these additional details.
4. These details about the changes that should be reverted and the commit to which it should be applied are transferred via the network using the appropriate communication protocols to the back end database.
5. The appropriate changes will have to be made to the back end database.
6. Possible Error Scenarios:
    a. If there is a loss of connectivity due to an unstable internet connection, the user will be asked to refresh and try again.
    b. If the user enters invalid commit info, he/she will be asked to reconsider and re-enter the details.

5.13 Create a user profile

5.13.1 Description and Priority

A profile page tells people the story of a user’s work through the repositories they're interested in, the contributions they've made, and the conversations they've had. This feature allows a user to create a profile by providing all the necessary information.

One can add personal information in their bio. For instance, previous workplaces, projects contributed to, or any interests they may have.
This is a low priority feature since it does not hinder project building capabilities in any way.

5.13.2 Stimulus / Response Sequences

Once a user's account is created, a button which says “create user profile” will be displayed.
On clicking the button, the user will be directed to the create profile page which is essentially a form. On supplying all the information, they can click the “submit” button at the end of the form. All their details will be updated and reflected on their account.

5.13.3 Functional Requirements

1. A web interface with a “create user profile” button that leads to a form asking for further details.
2. A form to collect all details that the user wants to feature replace the blank fields in the corresponding tables of the database.
3. Possible Error Scenarios :
    a. If there is loss of connectivity due to an unstable internet connection, the form submission will fail and the user will have to refresh the page and re-fill the form.
    b. A user might enter invalid data into fields. For instance, digits in the name field.
    In the event that something like this happens, a dialog box must pop-up indicating to the user that they have entered an invalid data type and must enter information in the appropriate format.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

- The back end should perforce be able to manage large volumes of data in the order of terabytes (TB). This is in terms of it’s storage capabilities.
- The data transfer rate across the network should be sufficiently large. In the order of several megabytes per second (MBPS).
- The back end should be able to handle sufficiently high network traffic. There should be an ample number of servers to be able to handle multiple requests being sent by a

large number of clients at the same time. In the order of a few thousand requests per second.

6.2 Safety Requirements

- There must be a cap on the volume of data that can be uploaded by a user in one go. This will reduce the load on the back end servers enabling it to safely store the data. This will prevent both loss and manipulation of data.
- The user alone is responsible for the data uploaded on our website. He is held responsible if he uploads any sensitive data that does not comply with the policies of our software.
- There is a possibility that there could be a natural disaster that affects the data centres storing and managing the data hosted on our website. To prevent damage and loss of data in such a scenario, the data is replicated for fault tolerance in multiple data centres situated in different geographical locations.

6.3 Security Requirements

Any developer working on this software must comply with the following security requirements at all times:

- India's regulatory mechanism for data protection and privacy is the Information Technology Act, 2000 and its corresponding Information Technology Rules, 2011 i.e. Reasonable Security Practices and Procedures and Sensitive Personal Data or Information.
- Personal data is also protected under Article 21 of the Indian Constitution which guarantees to every citizen, the Right to Privacy as a fundamental right.
- Registration Information is used to create an account, and to provide the services of our software. This is how the users are uniquely identified.
- Profile Information is shared with other users based on prior sought permission.
- User Personal Information, specifically username is used to identify a user on our platform.
- User Personal Information may be used if it is necessary for security purposes or to investigate possible fraud or attempts to harm our users.
- The user is forbidden from uploading sensitive information on the website like security keys.

6.4 Software Quality Attributes

- Availability
  The aim of the software is to provide our users with an experience similar to what they would experience if their repositories had been present locally. In this sense, our software should be highly available.

- Correctness
  We need to ensure that there is no corruption or loss of data when it is transferred over a network. Data transfer needs to be accurate, safe and fast. Additionally, the quality of data must be ensured at all times.

- Reliability
  It is quite possible that there could be several instances like servers being damaged due to natural disasters, power failures, human intervention etc. As developers, we need to make sure that we provide appropriate fault tolerance mechanisms by replicating data across data centers housed globally. Additionally, we will provide checkpointing as a means to restore normalcy in the event of a failure.

- Scalability
  Our software should be able to handle increasing loads seamlessly. This is in terms of the number of increasing users, volume of data being stored and network traffic among others.

- Usability
  As the name suggests, we need to design a convenient interface that developers across all levels should feel comfortable with. No additional knowledge should be required to operate on our platform.

- Maintainability
  Our software should be developed such that it allows for easy maintenance. Any new developer should find it easy to tend to any errors / maintenance issues.

- Flexibility
  A user is required to have the bare minimum, a personal computer and a stable internet connection. While some softwares require that the computer possess certain configurations and hardware requirements, our software demands no such thing.

6.5 Business Rules

- This project is the intellectual property of the following individuals affiliated to the Department of Computer Science & Engineering at PES University, Bangalore:
  1. Ameya Bhamare (ameyarb1804@gmail.com)

2. Darshan D (darshand2000@gmail.com)
3. Drishti Hoskote (drish2000@gmail.com)

- For any changes to be made in the software, it is imperative that one of our written permission is sought. Failing to do so will be treated as a violation of IP rights. Violators will be duly prosecuted.
- Any illegal activity or malicious intent on the platform will be treated seriously. The matter will be reported to the cyber police. It is upto them to take action as they see fit.

Domain Requirements

- As this is essentially a file system on the cloud, all the operations that can be done on a file system should be allowed on our software.
- The user should be able to create repositories, folders, files of different types, rename the files / folders, delete them, access and read them, write and make modifications to them and other such basic operations.
- The domain requires that our software provides features to manage code in the most effective way possible.
- The software should allow the developers to collaborate among themselves irrespective of their physical location.
- The software should provide features to have a distributed version control mechanism where the developers can maintain different versions of the file.

Appendix A: Glossary

- **Distributed Version Control**
  It is a concept where the complete codebase is mirrored on every developer's computer. This includes it’s full version history.

- **Repository**
  A repository contains all of your project's files and each file's revision history. One can discuss and manage their project's work within the repository.
  It can be either public or private. A public repository can be accessed by anyone whereas a private repository is accessible only to the owner and the collaborators on that project.

- **GUI**
  It expands to Graphical User Interface. As the name suggests, it provides the user with an interaction oriented interface allowing for a seamless experience on a platform.

- **Cloud Service Provider**
  A cloud service provider is a third-party company offering a cloud-based platform, infrastructure, application or storage services. Much like a homeowner would pay for a utility such as electricity or gas, companies typically have to pay only for the amount of cloud services they use, as business demands require.

- **Web Service**
  A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

- **Collaboration**
  Most projects involve a lot of developers. This means the project owner needs to add their team to the repository so they can contribute to the repository. Once the team is added as collaborators they'll be able to edit the repository.

- **Source-code management**
  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. SCM is also synonymous with Version control.
  As software projects grow in lines of code and contributor head count, the costs of communication overhead and management complexity also grow. SCM is a critical tool to alleviate the organizational strain of growing development costs.

- **HTML**
  Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. HTML describes the structure of a Web page. HTML consists of a series of elements. HTML elements tell the browser how to display the content.

- **CSS**
  Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language.

- **JavaScript**
  JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

- **Object Oriented Programming Language**
  Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields, and code, in the form of procedures. A feature of objects is that an object's own procedures can access and often modify the data fields of itself.

- **HTTP**
  Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes.

- **HTTPS**
  Hypertext Transfer Protocol Secure is an extension of the Hypertext Transfer Protocol. It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security or, formerly, Secure Sockets Layer.

- **TCP**
  TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other.

- **Backend**
  The part of a computer system or application that is not directly accessed by the user, typically responsible for storing and manipulating data. The back-end is the code that runs on the server, that receives requests from the clients, and contains the logic to send the appropriate data back to the client. The back-end also includes the database, which will persistently store all of the data for the application.

- **Data Centers**
  A data center is a facility that centralizes an organization's shared IT operations and equipment for the purposes of storing, processing, and disseminating data and applications. Because they house an organization's most critical and proprietary assets, data centers are vital to the continuity of daily operations.

- **Dialog box**
  The dialog box is a graphical control element in the form of a small window that communicates information to the user and prompts them for a response. Dialog boxes are classified as "modal" or "modeless", depending on whether they block interaction with the software that initiated the dialog.

Appendix B: Field Layouts

Registering a user

Field	Length	Data Type	Description	Is Mandatory
Username	15	Alphanumeric along with a few special characters like '_', '#', '$'	Field to store the name of the user that will be used to uniquely identify the user.	Yes
Email ID	254	Alphanumeric along with the special characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~	Field to store the email id of the user	Yes
Password	20	Alphanumeric along with the special characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~	Field to store the password required to authenticate the user.	Yes

Creating a user profile

Field	Length	Data type	Description	Is Mandatory
Name	100	string	Name of the user	Yes
Preferred nickname	20	alphanumeric	Nickname if desired	No
Bio	65536	alphanumeric	Personal information user wishes to share with the community	No
Organization	50	alphanumeric	Organization the user is part of	No
Domains of interest	100	alphanumeric	Domains that the user is interested in	No

Details required in the repository:

Field	**Length**	**Data type**	**Description**	**Is Mandatory?**
Repository name	25	alphanumeric along with a few special characters like '_', '#', '$'	Name of the repository	Yes
File name	25	alphanumeric along with a few special characters like	Name of the file	Yes

		‘_’, ‘#’, ‘$’		
Branch name	25	alphanumeric along with a few special characters like ‘_’, ‘#’, ‘$’	Name of the branch	Yes
Commit message	100	alphanumeric along with a few special characters like ‘_’, ‘#’, ‘$’	A message describing the commit	No

Sample Report Requirements

User Registration Report	**Profile Creation Report**	**Repository Report**
Username	Name	Repository Name
Email ID	Preferred Nickname	File Name
Password	Bio	Branch Name
	Organization	Commit message
	Domains of interest	

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	Source code management	Entails everything that is required to manage source code across all the repositories					
2	Distributed version control	Providing the ability to collaborate among developers and the ability to maintain different versions of a file					
3	Remote hosting of repositories	Hosting the repositories on the cloud					
4	GUI	A user friendly web interface that allows the user to navigate around the platform comfortably					
5	Creating a new repository	Creating a placeholder					

		for the files and folders of the user					
6	Backend database	A database on the server side that stores and manages all the required data					
7	Communicatio n Protocols	All the necessary protocols and standards to transfer large volumes of data over the network					