FINAL SEMESTER ASSESSMENT (FSA)
B.TECH. (CSE)
VI SEMESTER

UE18CS355 – OBJECT ORIENTED ANALYSIS AND DESIGN WITH SOFTWARE ENGINEERING LABORATORY

PROJECT REPORT

ON

SOURCE CODE MANAGEMENT AND VERSION CONTROL SYSTEM

SUBMITTED BY

NAME	SRN
1) Yathish N V	PES1201801462
2) Vijay Kumar	PES1201802071
3) Prashant	PES1201802089

JANUARY – MAY 2021
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RR CAMPUS,
BENGALURU – 560100, KARNATAKA, INDIA

TABLE OF CONTENTS

ABSTRACT

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting.One of the most popular VCS tools in use today is called Git. Git is a *Distributed* VCS,free and open source.This project focuses on handling the source code with version control and overcome the conflict of source code and gives various other features to the users.We use Repositories to handle the code and creating repositories and managing code and managing users are main features of our project.

SOFTWARE REQUIREMENTS SPECIFICATION

1. Introduction

1.1 Purpose

This document describes a version control system and repository hosting to facilitate remote code management and software collaboration. This is the version 1.0 of the Software Requirements Specification (SRS). It fully describes the working and the features of the version control system, its interface with end users, and the functional and non-functional aspects of the project undertaken.

1.2 Intended Audience

The intended audience of this document are developers, maintainers, testers, Q&A and project managers. It may also be used by users to better understand the features, interface and other required descriptions as necessary.

The rest if this document is divided into following parts :

1. Section 2 : Covers description that provides an overall picture of the product
2. Section 3 : Details the interface to end users and it's requirements
3. Section 4: Pertains to the analysis of the design of the product's features
4. Section 5 : Describes the product's features in detail

5. Section 6 : Discusses non-functional details and requirements of the product

6. Section 7 : Covers any other remaining requirements

1.3 Product Scope

The product is a remote code repository hosting service with version control system (VCS) to facilitate development of source code by multiple contributors in an organised and easy to maintain manner. The web based GUI allows for easier interaction for users, while the VCS allows faster and more robust development. The distributed VCS also allows clients to not only checkout the latest changed source code, but also fully mirror the repository locally. This allows the client to go back any number of changes. This also gives the additional feature that in the unlikely chance of the server hosting the service crashes, the client can easily restore the entire repository simply by copying back to the server. It is also possible for simultaneous edits by different developers. If a version being worked on were to crash, then the developers can revert back to the last working version. They can further analyse the mistake, also use the logs to understand the changes made and what possibly caused the crash.

1.4 References

- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
- Official git documentation: https://git-scm.com/docs
- Git remote repository protocols: https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols
- Pro Git: https://git-scm.com/book/en/v2
- Reference for version control : https://www.atlassian.com/git/tutorials/what-is-version-control

- Reference for source code management : https://www.atlassian.com/git/tutorials/source-code-management

2. Overall Description

2.1 Product Perspective

With large code bases involving teams of large numbers of developers, being able to collaborate is an essential requirement. Modern software development is done with an iterative and incremental approach, hence this means there is constant improvement to the existing code-base, and as the complexity and the size of the software increases, there is a need to have a platform which helps teams manage, track and maintain their software development process, in order to collaborate easier and release quality software faster. This also allows for better and faster integration with testing suites and QA teams, at no loss of productivity of the developer. The product also aims to provide an intuitive and friendly interface to the user, while making no assumption of a strong technical background.

2.2 Product Functions

Main features available to users are :

1. Create and maintain repositories

2. Add other developers to created repositories

3. Commit changes to files and be able to push commits, along with other metadata such as which files were changed, what contents were changed, who made the changes and why.

4. Clone remote repositories locally

5. Pull changes made by other developers on remote repositories into local clones

2.3 User Classes and Characteristics

While developers will be the primary users of our product, we have also outlined other users and their characteristics.

1. Developers:

a. Expected to be the highest frequency users

b. Have the technical background to deal with the intricacies of the software

c. Access to repository given on a basis as needed by owner

d. Will utilise the product to the maximum extent

2. Project Managers:

a. Mostly will use software to monitor progress of the project, and thus will not have the frequency of developers

b. Prefer using GUI, and hence will need less technical background

c. More access to repositories and the capability to revoke or grant access the of repositories to the developers

d. Maintaining an auditable history of changes made to the software is of utmost importance

3. Product User / Contributors:

a. Cannot make direct changes to the software repository

b. Usually used to raise issues with the project

c. No control over repositories granted

d. Almost no technical knowledge requires and can easily use the web application for their purposes.

2.4 Operating Environment

The software is to be designed to run on any remote server as a web service. The server side will need to avail cloud services to host the website and to store the repositories. It should also support devices running major OS like Linux, Windows, Mac OS etc. to be able to connect to the host remotely through the internet, using a standard web browser such as Google Chrome or Mozilla Firefox.

2.5 Design and Implementation Constraints

1. Limitations :

a. Supported types of files in repository :

i. Source code files such as .py/.c/.cpp/.js etc

ii..log

iii. .png

iv. .zip

v..pdf

vi. .txt

vii..md

viii. .doc

b. Supported languages in website : English

2. Implementation constraints :

a. A Relational Database System (RDBMS) will be used

b. Encryption systems will be in place to ensure that any user actions will be made securely

c. Digital signatures and user authentications will be used to prevent any fraudulent user action.

d. Secure Communication protocols that will be used include HTTPS and SSH to enable secure and reliable transfer of data over the network

3. Our product or our team is not responsible for the user application developed on our platform. We in no way promote or support the sharing of proprietary or illegal code on our platform.

a. 2.6 Assumptions and Dependencies

No assumptions other than any previously stated ones. Also no external dependencies with respect to requirements mentioned earlier.

3. External Interface Requirements

3.1 User Interfaces

- Landing Page:

1. Introduction and information about the product

2. Links for signup and login.

- Login:

1. Fields for username and password.

2. Option to recover password.

- Signup:

1. Fields for First Name, Last Name, Username, Password, Confirm Password

2. Submit button

- Repository Page:

1. View the files, source code, and commit messages.
2. Option to clone the repository to a local machine.
3. Readme for the project.

- Source Code Viewer:

1. View source code.

- Commit History:

1. Display the changes made in the project in sequential order.
2. Also display commit messages and files changed.

- User Profile:

1. View User public information
2. View public repositories
3. Option to follow user

3.2 Software Interfaces

The backend of the product is connected to the main database component, domain registration services and various services from the chosen cloud service provider. All components will undergo unit and integration tests and will be continuously integrated into the deployed product version. The incoming data items include the repository files as well the repository metadata such as the commit history and information about the individual commits. For the user, the information includes their profile information and their authentication details. The outgoing data items include the repository files and metadata which will be sent when the user views a repository or file. For the user, the information includes their public information. The services needed will include the Private Cloud for development of the product and a Domain for deployment of the product, as well as a database service to store user and repository data.

3.3 Communications Interfaces

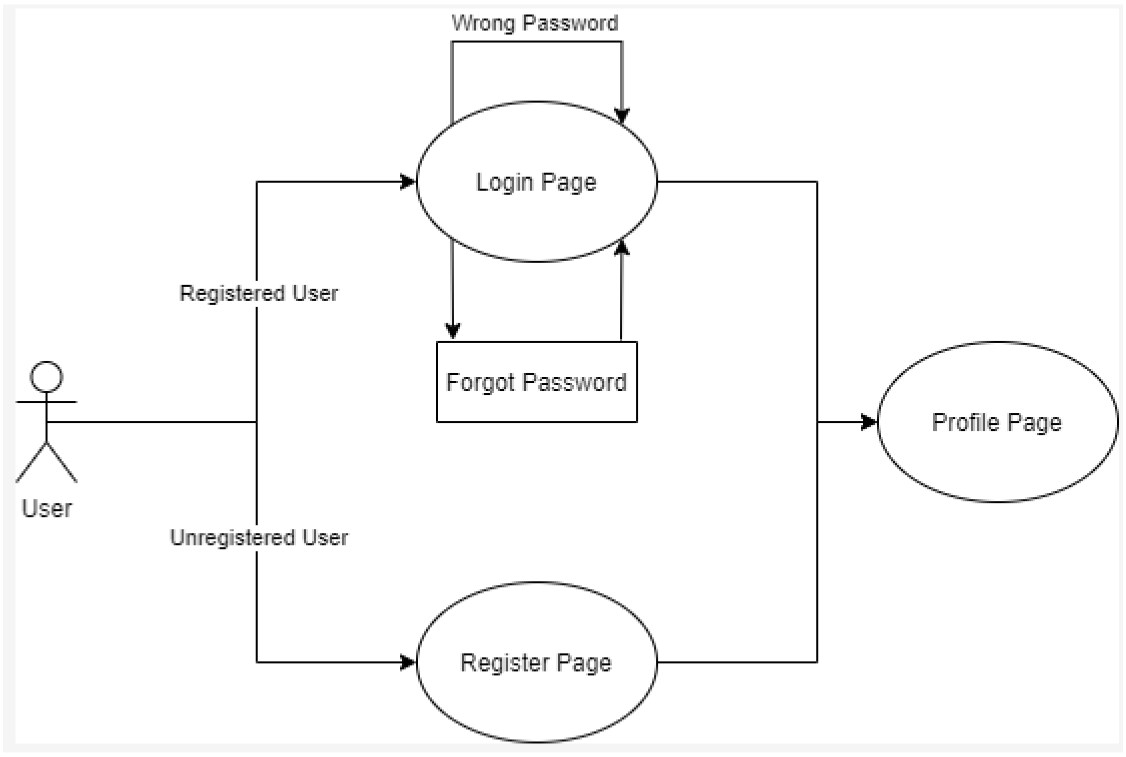
Protocols such HTTPS and SSH which have proven to be secure and reliable for communication will be used to facilitate the communication between client and the servers running the backend. The HTTPS protocol will be used for transfer of files. The user will need to authenticate before the communication, this is done so as to maintain the security and integrity of the project being made. SSH protocol is used to allow users to interact with the remote server. The users will need to maintain a local SSH key and provide the public SSH key to the server to facilitate secure interaction.

The git protocol is a fast network transfer protocol,however lacks authentication hence less secure, can be used to allow users to work with public repositories, i.e. repositories that do not need user authentication.

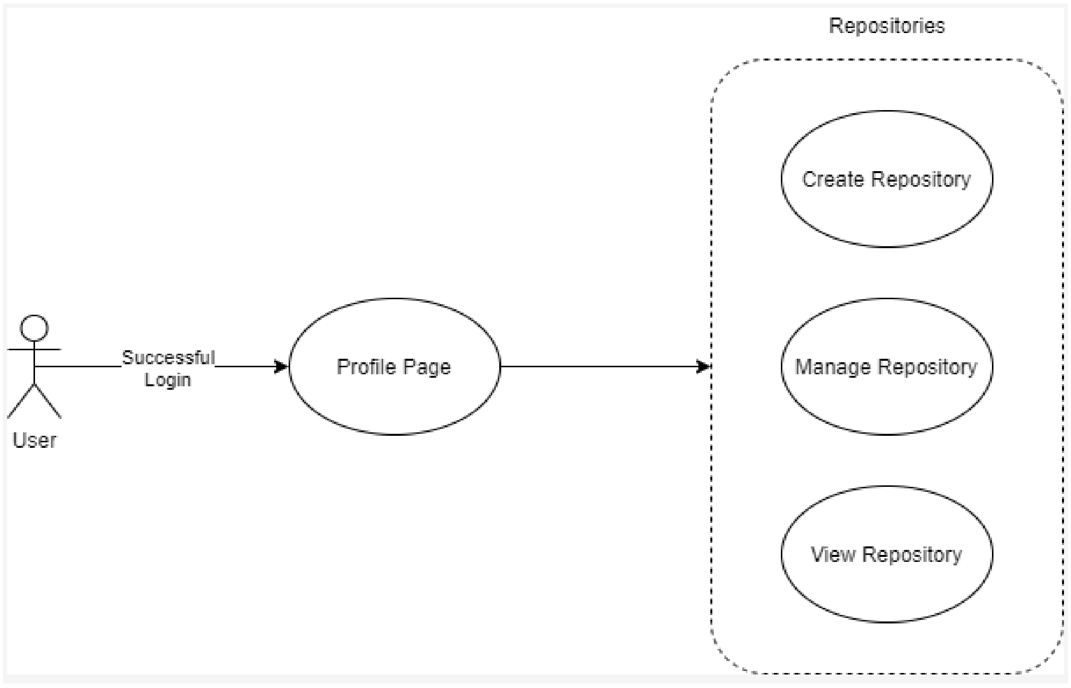
4. Analysis Models

4.1 Use Case Diagrams :

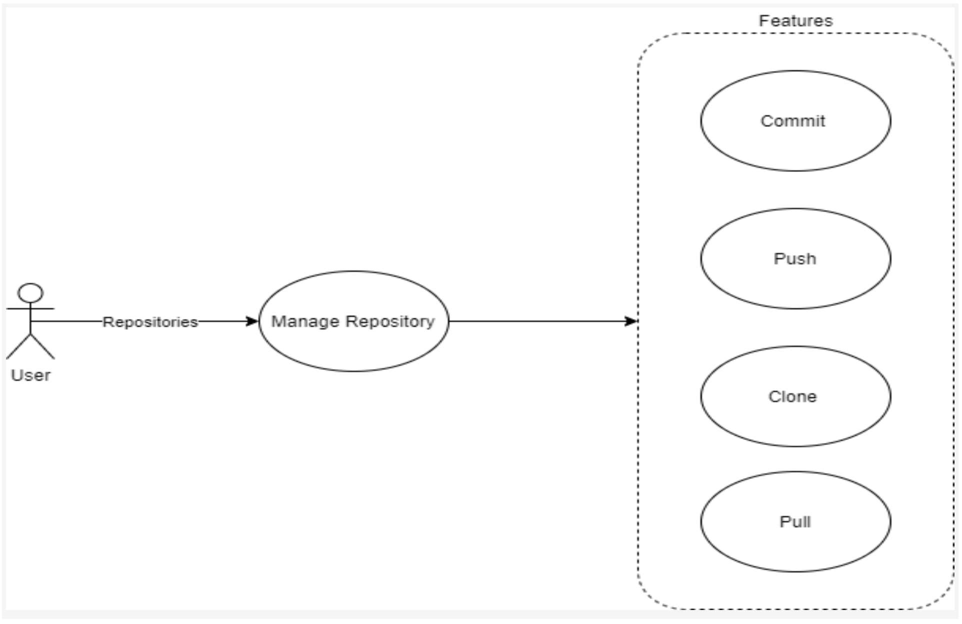
4.1.1 User Login :



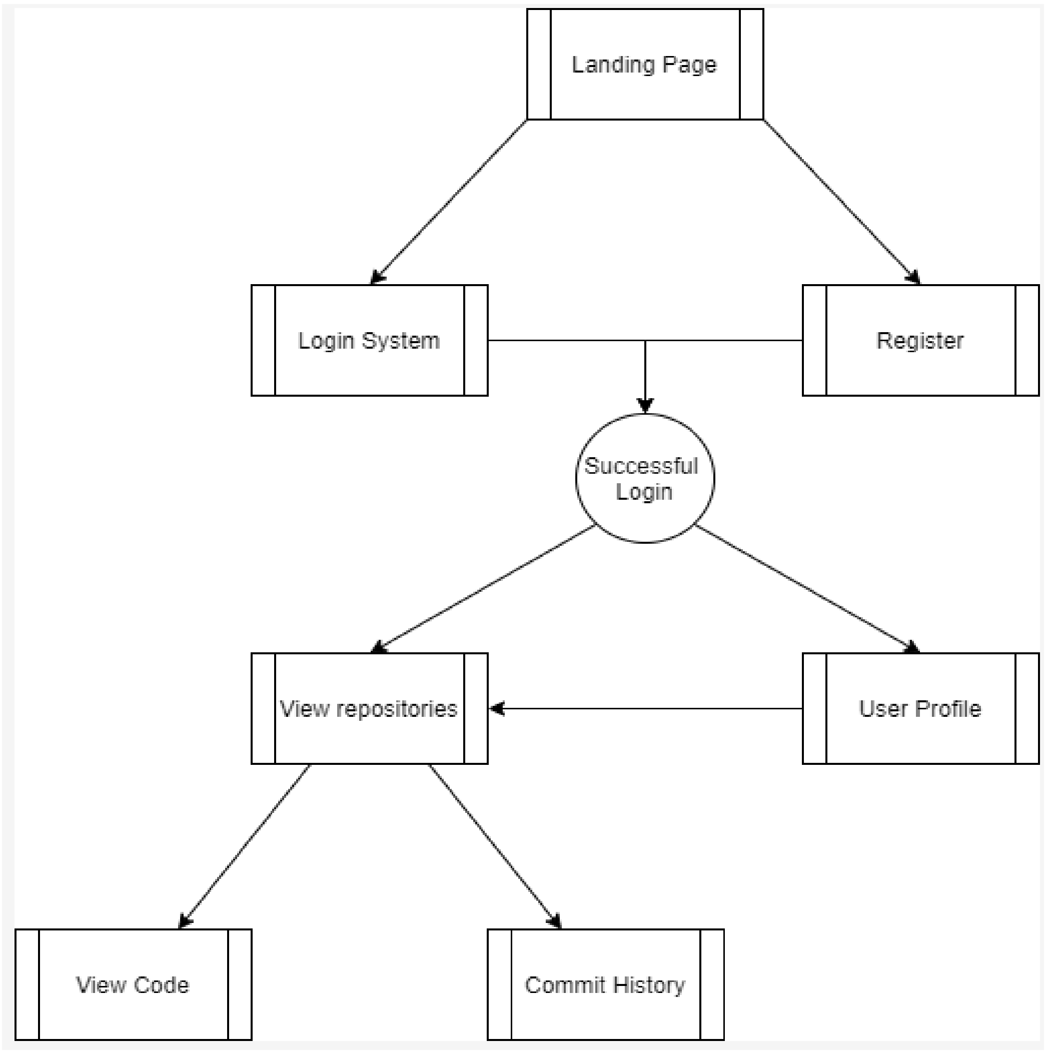
4.1.2 Repositories List



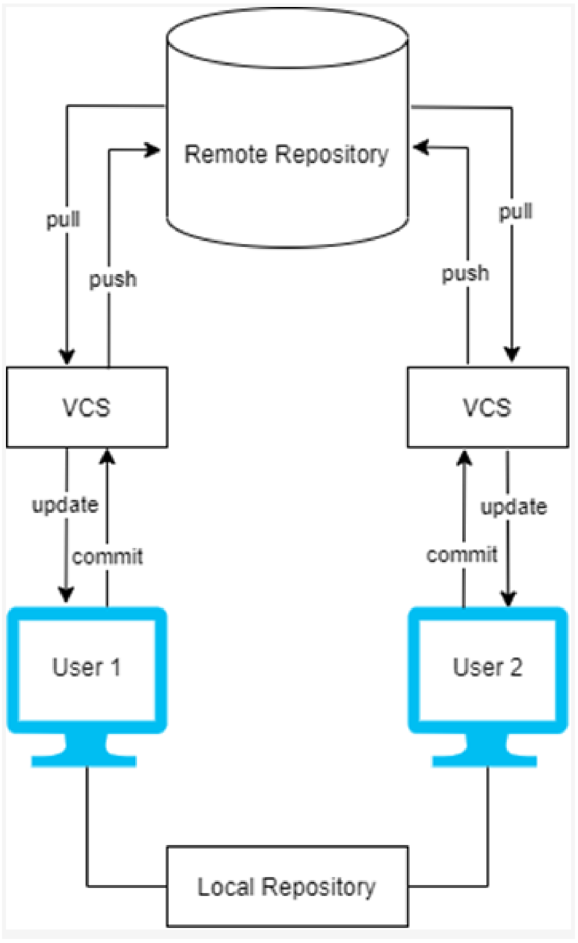
4.1.3 Manage Repository:



4.2 Application Flow Diagram



4.3 Overview Diagram



5. System Features

5.1 Create Repository

Allow users to make their own repository to store their files and programs

5.1.1 Description and Priority

1. Users can create and initialise their own repositories. A repository is the entire collection of files, programs and folders associated with a project as well as each file's history of changes.

2. This feature is one of the highest priority features in the project as repositories are at the very heart of version control projects.

5.1.2 Stimulus/Response Sequences

For CLI: By running an init command in the command line interface, users can initialise a repository in the working directory of the project.

For GUI: On the web application, the users can create a repository by clicking on a “Create Repository” button given. Upon creation, in the web application, the users will be presented with a page with fields to enter information such as repository name, collaborators, owner name, toggle privacy settings etc. The users will also be given the option to initialise a README to describe the project and edit it as required. Users can also add the list of file extensions they would like to ignore when making changes to the repository. The users can also decide upon the license of their repository as suiting their needs and intentions.

5.1.3 Functional Requirements

1. Command line interface should be given a “init” command to create a repository.

2. The web application should be provided with a button for creation of the repository.

3. If a repository is duplicate under the same user, an appropriate message must be displayed to suggest the same.

4. The database should be updated as needed with the metadata and other information about the repository upon initialisation.

5.2 Commit:

5.2.1 Description and Priority

Commit operation stages the changes made to the repository along with a “commit message” which may explain the reasons for the changes undertaken. This allows for users to track what changes were made by whom and also to track possible bugs as they may trace back through commits to find the issue, and even revert back the repository to the last known working change.

5.2.2 Stimulus/Response Sequences

The user has 2 ways to commit the changes made:

1. For CLI: The user, after authenticating with their username and password, will have to identify the files they wish to stage for the commit. They can do this using an add command. *Only those files that have been added will be committed to the repository.* The user can then commit their work into their repository. However on the CLI the commits are *only visible locally*. Only when the user pushes the changes to the remote repository will it be reflected there. While committing, the user can provide a message with the commit to describe the changes made to the repository in the current commit with respect to previous commits.

2. For GUI: After logging in, the user can choose to “add a file”. Upon choosing the option, they can either edit an existing file using the online editor available, or upload a file from their local device. Once the user is satisfied with the files added and/or edited, they can commit the changes by choosing to click on a “Commit File” button available. If the files were uploaded, then the user will be presented with fields to make a commit message and commit the changes with a button “Commit changes”.

Using either CLI or GUI, the user can choose to commit changes to the repository.

Once a commit is made, the software will capture the changes made and use it to create a snapshot of the repository.

5.2.3 Functional Requirements

For CLI:

1. Current working directory must be an initialised git repository.

2. Upon error to commit, an appropriate message should be displayed to the user and also, if possible, what can be done to remedy the error.

For GUI:

1. Appropriate button to allow users to choose required options as necessitated

2. An online editor tool to make changes to the files already on the remote repository.

3. Provide options and permission to upload files as needed by users.

5.3 Push

5.3.1 Description and Priority

Push feature allows a user to add their local commits into the remote repository. This feature will be a high priority feature, as being able to reflect the changes made locally in the remote repository for all the collaborators to be able to see is essential for project development.

5.3.2 Stimulus/Response Sequences

For CLI: The user can use the command “push” to perform the push operation. Upon the command, the software will send a pull request to the remote repository. It will check if the local copy is up to date with the remote repository on all other files other than the ones changed locally. Only then it will allow for a push to be made.

For GUI: The user can click on a button which would allow them to push their commits into the repository. They are then redirected to a page confirming push along with any description they would like to provide.

5.3.3 Functional Requirements

For CLI:

1. Command to push local changes to the remote repository

2. Appropriate error message when changes could not be pushed, either due to internet connectivity issues or due to any other issues.

For GUI:

1. Button to make a push into the repository

2. Fields to provide title, description and any other necessary information to the maintainers to better understand the content of the push.

5.4 Clone

5.4.1 Description and Priority

Clone feature allows users to download a local copy of a specified remote repository. This will allow them to continue their work related to the project locally.

This feature too is a high priority feature as it is necessary to be able to make changes locally before making changes to the remote repository to ensure stability of the project and also to allow developers to be able to work independently.

5.4.2 Stimulus/Response Sequences

For CLI: The users can clone a remote repository by using the command “clone” along with the url of the remote repository they wish to clone. The software then checks if the mentioned remote repository allows for the user requesting cloning to clone it. If allowed, the software then downloads a local copy of the repository.

For GUI: Upon visiting a repository they wish to clone, the user may clone by clicking on the button “Clone”. The user, if they have permission, will download the repository to their local device in zip format.

5.4.3 Functional Requirements

For CLI:

3. Command to clone repository, this should either by the use of SSH or HTTPS as seen better suited for the request

4. Have appropriate error messages to inform users in case of internet connectivity loss during cloning.

For GUI:

1. On the web application, a button to facilitate cloning of the repository in zip file format

5.5 Pull

5.5.1 Description and Priority

Pull feature allows users to download remote commits of a specified remote repository. This will allow them to continue their work related to the project locally after a collaborator has made changes and pushed those changes as commits to the remote repository.

This feature too is a high priority feature as it is necessary to be able to synchronize changes from the remote repository before making changes to the local repository to ensure stability of the project and also to allow developers to be able to work independently.

5.5.2 Stimulus/Response Sequences

For CLI: The users can pull changes from the remote repository by using the command “pull”. The software then checks if the mentioned remote repository allows for the user requesting pulling of commits to pull it, and if the remote repository has any changes to pull. If present, the software then downloads the commits which have been pushed to the remote repository.

For GUI: N/A

5.5.3 Functional Requirements

For CLI:

1. Command to pull changes from a remote repository, this should either by the use of SSH or HTTPS as seen better suited for the request

2. Have appropriate error messages to inform users in case of internet connectivity loss during cloning.

For GUI: N/A

5.6 Delete Repository

5.6.1 Description and Priority

Delete repository feature allows users to delete a remote repository. By doing this, in the case the hosting service allows only for a limited number of repositories per user. Then the user can remove repositories they are no longer interested to host remotely and make space for new repositories.

This feature is of priority mid level, cause this depends on the hosting service and the kind of storage available to the service.

5.6.2 Stimulus/Response Sequences

For CLI: The user can issue a delete command that will delete the repository remotely. When command is issued, the user should be asked to authenticate their username and password. Also the user must be notified that their action is irreversible and they can no longer retrieve the remote repository they are deleting.

For GUI: The user can go to settings of the repository and choose to delete by selecting the delete option. The user will then be asked to authenticate themself. Upon successful authentication, confirm with the user the action they are about to perform and that it is irreversible. Once confirmed, delete the remote repository, redirect users to their profile.

5.6.3 Functional Requirements

For CLI:

1. Command to delete repository that will send a message to the hosting service where remote repository is stored

2. Authentication of users in a secure manner.

For GUI:

1. Option to delete on the web app

2. Confirmation of delete

5.7 Add Collaborator

5.7.1 Description and Priority

Add collaborator feature allows the owners of a remote repository to add other users as collaborators to a remote repository. Using this, the newly added user will be able to push and pull from the repository thereby allowing multiple users to collaborate on a single repository.

This feature holds a mid-level priority as it is dependent on the proper functioning of higher priority features like push and pull.

5.7.2 Stimulus/Response Sequences

For CLI: N/A

For GUI: The owner of the repository can add a user given his username. The latter will then receive a message/mail about the invite to collaborate. If they choose to accept, they will be authenticated before being able to push to and pull from the repository. The user will also be able to see the list of all collaborators.

5.7.3 Functional Requirements

For CLI:N/A

For GUI:

1. Option to add a user as collaborator in the settings of the repository.
2. Authenticate both owner and user on request for addition of collaborator.

5.8 Remove Collaborator

5.8.1 Description and Priority

Remove collaborator feature allows owners of repositories to remove existing from a remote repository. This would allow for an owner to remove a user after they have contributed to the repository which removes the ability to push to and pull from the remote repository.

This feature holds a mid-level priority as it is dependent on the proper functioning of higher priority features like push and pull.

5.8.2 Stimulus/Response Sequences

For CLI: N/A

For GUI: The owner of the repository can see an option to remove a user in the list of users. After removal of the user, they will receive a message/mail informing them of the same.

5.8.3 Functional Requirements

For CLI: N/A

For GUI:

1. Option to add a user as collaborator in the settings of the repository.
2. Authenticate owner on request of removal of collaborator.

5.9 Log

5.9.1 Description and Priority

Log feature displays the commit log of a given repository. The commit log includes information like - a unique 40 character checksum, the user who made the commit, the commit message and timestamp. It allows a user to observe the different commits made to the repository.

This feature holds a low priority as it is a non essential feature that serves to track the usage of the application.

5.9.2 Stimulus/Response Sequences

For CLI: The user can use the command ‘log’ in order to see the required data in tabular form on the command line.

For GUI: The user can see the same data mentioned above when using the web application in the settings of the repository.

5.9.3 Functional Requirements

For CLI:

1. Command to display the commit log as specified

2. Appropriate error message should be displayed if there are any issues while fetching the log data

For GUI:

1. The commit log data should be visible in the settings of the repository.

5.10 Stash

5.10.1 Description and Priority

Stash feature allows users to save their committed and uncommitted changes for possible later use. It reverts back to the last working copy of the repository. This feature allows users to make changes and be able to undo them so as to not deal with possible conflicts when they pull changes from a remote repository

This feature holds a low priority as it is a non essential feature that serves to track the usage of the application.

5.10.2 Stimulus/Response Sequences

For CLI: User uses a stash command that carries out the operation. User can also add the specific file names, if they wish to stash changes only on those files.

For GUI: N/A

5.10.3 Functional Requirements

For CLI:

1. A command to execute stash operation along with the option to provide file names

For GUI: N/A

6. Other Nonfunctional Requirements

6.1 Performance Requirements

1. The software must support 1 user per local repository

2. Each user account must be capable to managing up to 20 repositories locally, simultaneously

3. Any changes made, should be reflected within a reasonable time frame, for example, changes with 10000 character changes should be reflected within 2 seconds.

4. Provided with a stable internet connection, the software must be capable of downloading/uploading files with at most half the bandwidth.

5. The server hosting the web application must be capable of handling large traffic and also be able to retrieve the required data about repositories in a reasonable amount of time.

6.2 Safety Requirements

● Users must avoid making changes to their local repository before pulling all changes made to the remote repository. Thus we encourage effective communication between the collaborators over the repository.

● Users must avoid using this product to track changes to files in binary format such as videos, images, music etc. We encourage users to only use this product with files which facilitate line based versioning such as code files and text data.

● The product makers are not responsible for any loss of code or any other intellectual property due to misuse or misunderstanding of the product.

6.3 Security Requirements

1. Users will be required to authenticate themself before making any changes to the remote repository. This ensures no malicious changes take place onto the project.

2. Use of SSH is promoted over HTTPS to ensure better security with regard to access of remote repository

3. Adherence to the Information Technology Act, 2000 and its corresponding Information Technology Rules, 2011.

6.4 Software Quality Attributes

1. The software has high accessibility as it can be used both from CLI and GUI, hence catering to the needs of the entire spectrum of technical abilities, and allowing for easy access from cloud infrastructure.

2. The software also gives higher portability as with very little configuration it can be used on almost any device.

3. Different versions of the same project can be maintained with ease.

6.5 Business Rules

- The project is open sourced under the MIT License.
- The project administrators are as follows

YATHISH NV

PRASHANT

VIJAY KUMAR

---

7. Other Requirements

Cloud service requirements : Exact requirements based on scale of user base, hence TBD

Monetary requirements : TBD

Appendix A: Glossary

- SSH : SSH or Secure Shell is a cryptographic network protocol for operating network services securely over an unsecured network
- 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.
- Version Control System (VCS) : version control is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.
- Repository : a repository is a data structure that stores metadata for a set of files or directory structure.

● Commit : commit is an operation which sends the latest changes to the source code to the repository, making these changes part of the head revision of the repository

● CLI : A command-line interface is used to process commands in the form of lines of text.

● GUI : Graphical User Interface provides the user with an interaction oriented interface allowing for a seamless experience on a platform

● Distributed Version Control : It is a form of version control in which the complete codebase, including its full history, is mirrored on every developer's computer

Appendix B: Field Layouts

Register the user

Field	**Length**	**Data Type**	**Description**	**Is Mandatory**
Username	50	Alphanumeric	Name of the user	Y
Email ID	50	Alphanumeric		Y
Password	30	Alphanumeric		Y
Confirm	30	Alphanumer	Confirm passwords	Y

Password		ic	match	

Login user

Field	Length	Data Type	Description	Is Mandatory
Email ID	50	Alphanumeric		Y
Password	30	Alphanumeric		Y

Creating and managing a repository

Field	Length	Data Type	Description	Is Mandatory
Repository name	40	Alphanumeric	Should be unique compared to other repositories owned by user	Y

Description	200	Alphanumeric	Briefly explains the project	N
Readme	Unspecified	Alphanumeric	Detailed explanation, amy contain more information that is need to users upon landing on repository	N

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID
1	Create Repository	Creating repository	Arch1	AD-1	repo.php	U105
2	Commit	Commit changes to the repository	Arch2	AD-2	manage_file.php	U202
3	Push	Pushing changes from local to remote repository	Arch2	SD-2	push.py	U205
4	Clone	Downloading remote repository	Arch2	AD-2	clone.py	U208
5	Pull request	Pulling changes from remote to local repository	Arch2	SD-2	pull.py	U206
6	Delete Repository	Deleting remote	Arch2	AD-2	delete.php	U209

		repository				
7	Add Collaborator	Adding user as collaborator to repository	Arch3	SD-3	users.php	U301
8	Remove Collaborator	Removing user as collaborator to repository	Arch3	SD-3	user.php	U304
9	Log	Display log of changes to a repository	Arch3	AD-3	home.php	U308
10	Stash	Save changes made separately before pulling, to avoid conflicts	Arch3	AD-3	home.php	U306

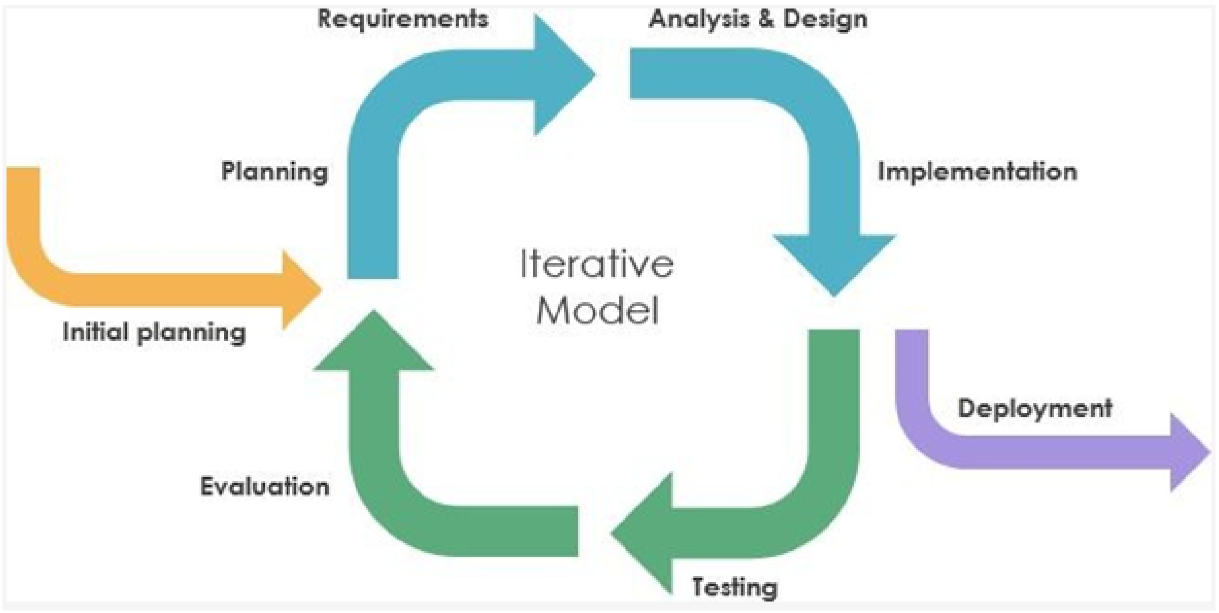
PROJECT PLAN

1. Project Lifecycle

The lifecycle we intend to follow for the execution of our project, of version control system, is the **Iterative Model**. We intend to do so for the following reasons :

- The SRS strictly defines the requirements of the final product
- While the main software deliverable is predefined by SRS, it may advance over time and may need to be adapted to the modified requirements.
- Easier to track problems, defects and any other issues and solve them effectively.
- By getting user feedback and improving iteratively, we can assure the project is upto the expected standards and satisfactory to the users.
- This approach also reduces the documentation overhead and hence ensuring more time towards the design, development and testing.

Lifecycle Diagram :



2. Tools to be Used

2.1. Version Control:

- Github

2.2. Design Tools:

- StarUML
- draw.io
- LucidChart

2.3. Development Tools:

- Vscode
- Python3
- Postgresql

2.4. Planning Tools:

- Trello

2.5. Bug Tracking:

- JIRA

2.6. Testing Tools:

- Katalon Studio
- Postman for testing REST API’s
- Unittest

3. Deliverables:

We intend to deliver a version control system, it would allow users to carry out software development with usage of source code management, collaborate and maintain their project in a more efficient manner. By also providing a web app,we allow users to also store their code in a remote repository, hence providing a back up if necessary and making collaboration easier for developers.

Following are the deliverables of out project :

1. Version control :

a. Creating repository

b. Commiting changes to repository

c. Push changes to remote repository

d. Clone remote repository

e. Pull changes made on remote repository

2. WebApp:

a. Landing page

b. Signup and login page

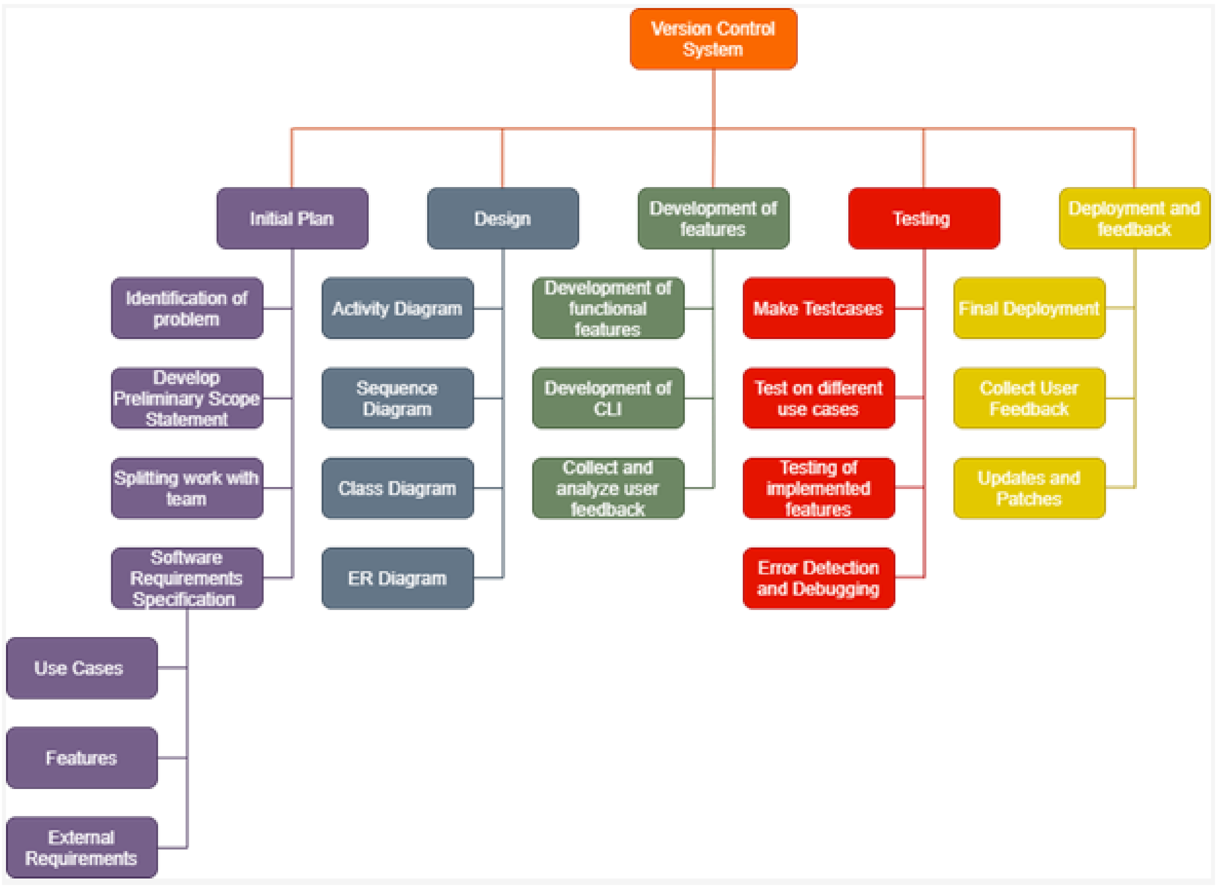
c. Repository viewing page

d. Source code reader

e. User profile

f. Commit history view

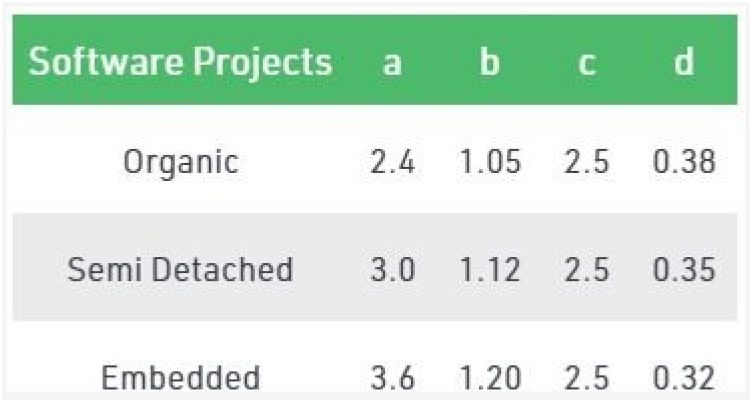
4. Work Breakdown Structure :



5. Estimation of Efforts :

Constructive Cost Model:

The Constructive Cost Model is a procedural software cost estimation model.The model parameters are derived from fitting a regression formula using data from historical projects. The various parameters used are the likes of size, effort, cost, time and quality.



Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

We can classify the project as Organic since the requirements are lucid and the team size is small.

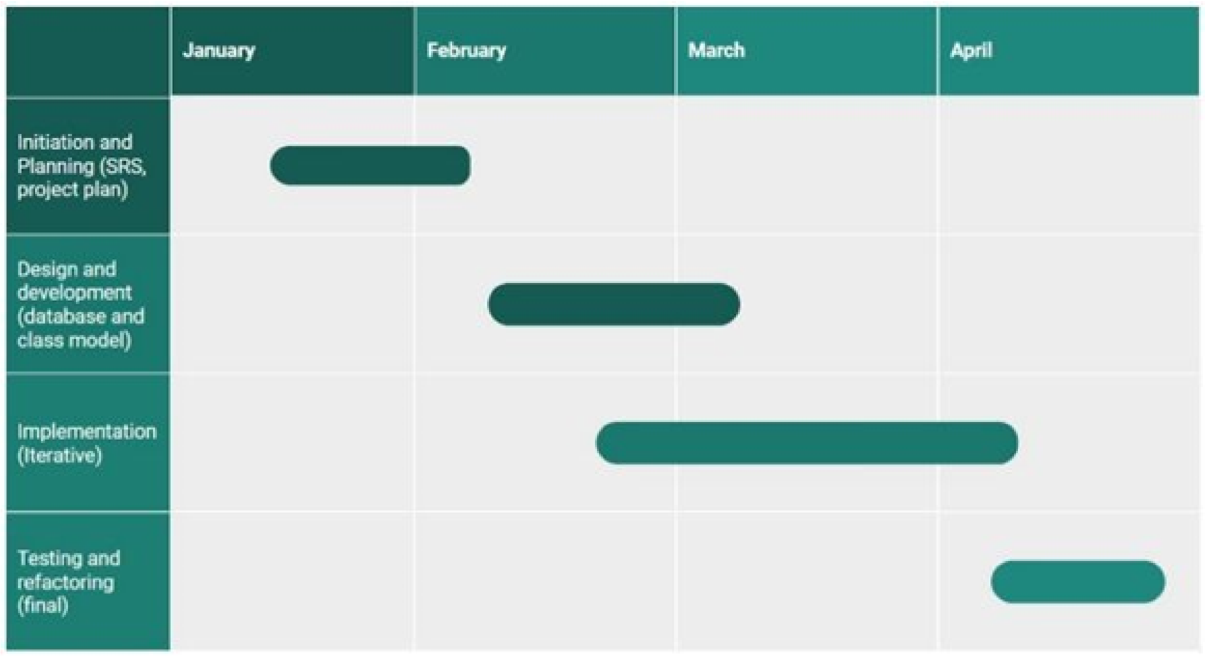
- Kilo lines of Code: Approximately 5,000 lines = 5K lines
- Effort = a (Kilo lines of Code) b

Effort=2.4
(5))$^{1.05}$
Effort =
13.005 pm

- Duration = c (E)^{d}

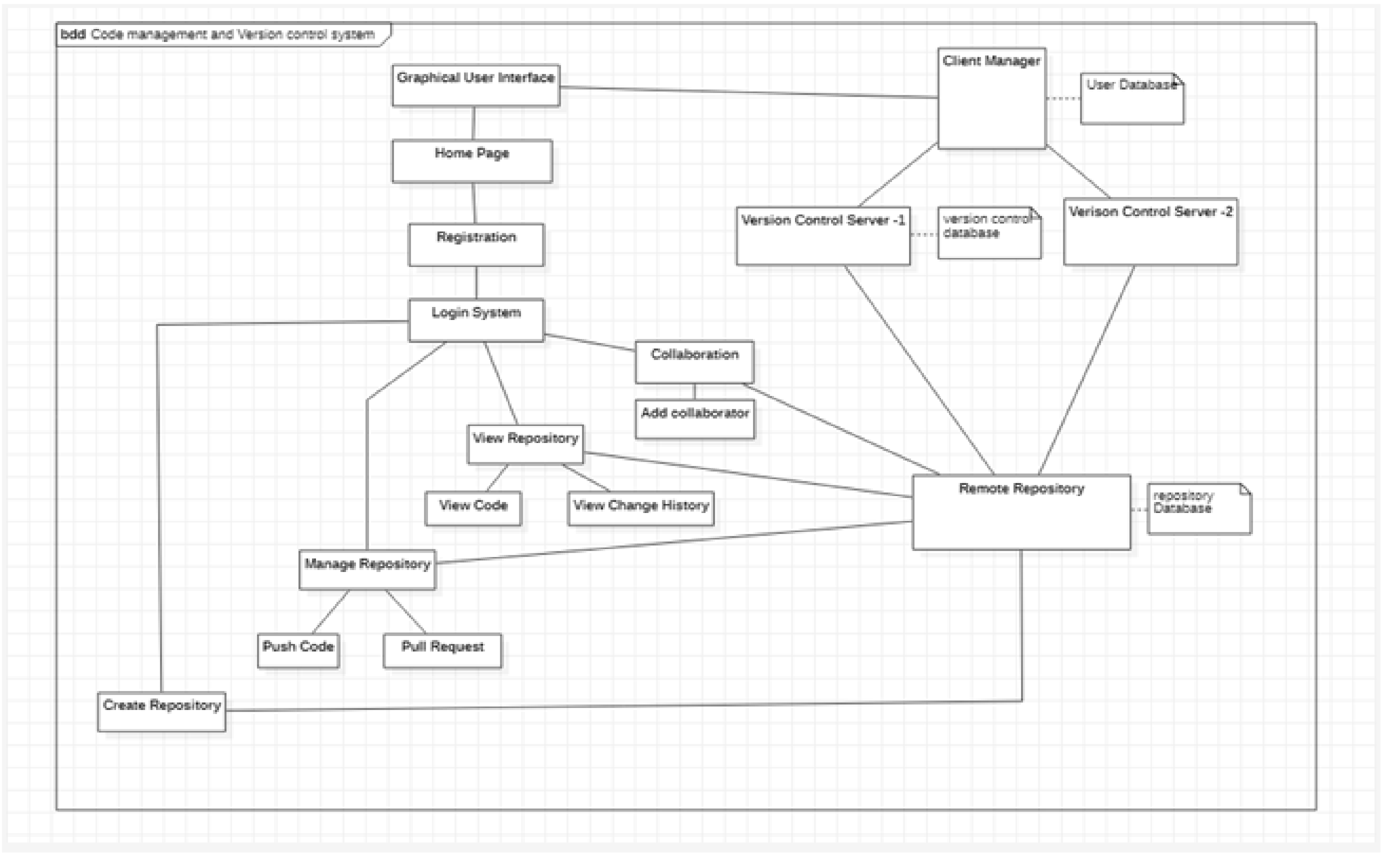
Duration = 2.5 x (13.005)$^{0.38}$ = 6.26 months
- Team Strength = 3

6. Scheduling

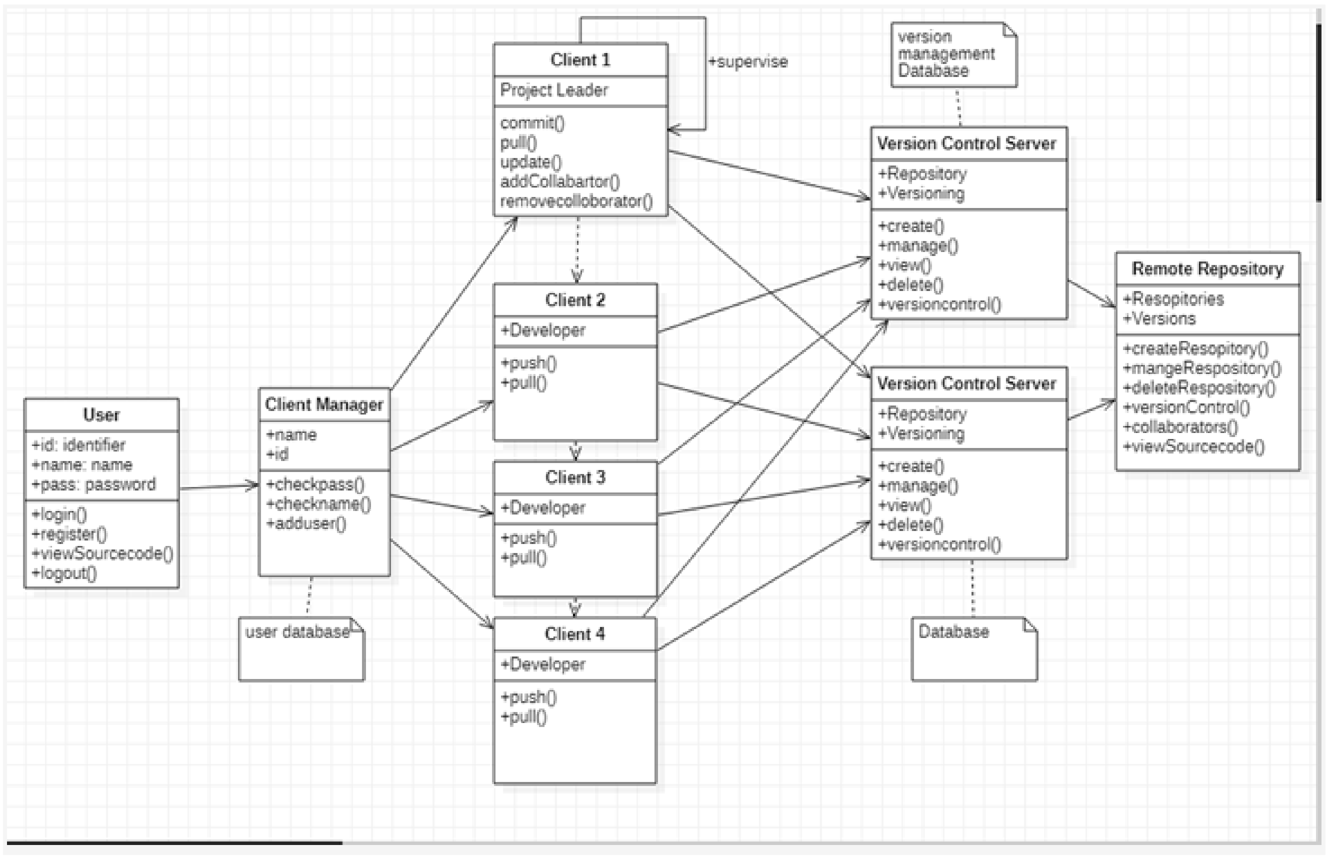


DESIGN DIAGRAMS

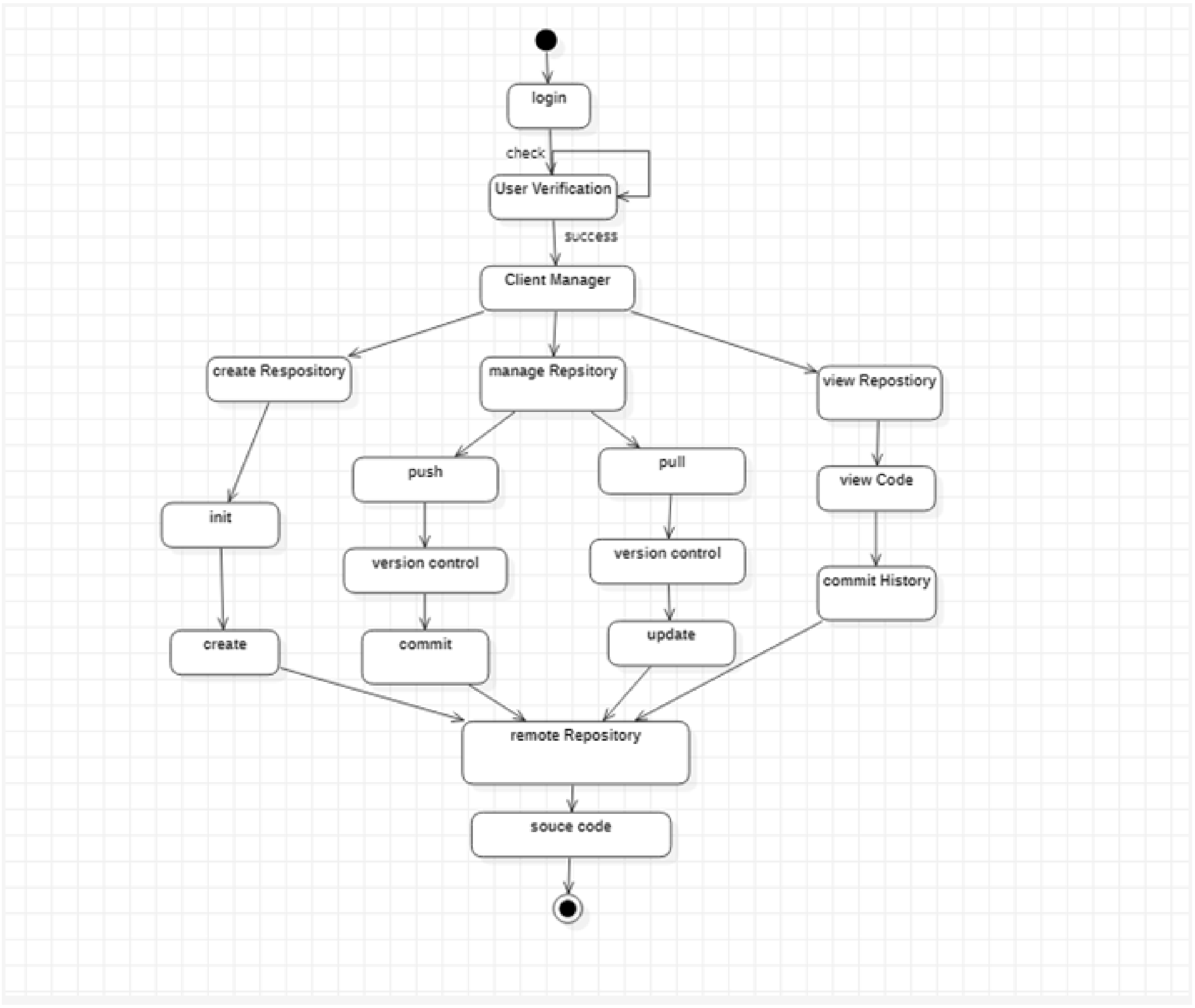
Block Diagram (BD-1)



Class Diagram (CD-1)



State Diagram (SD)

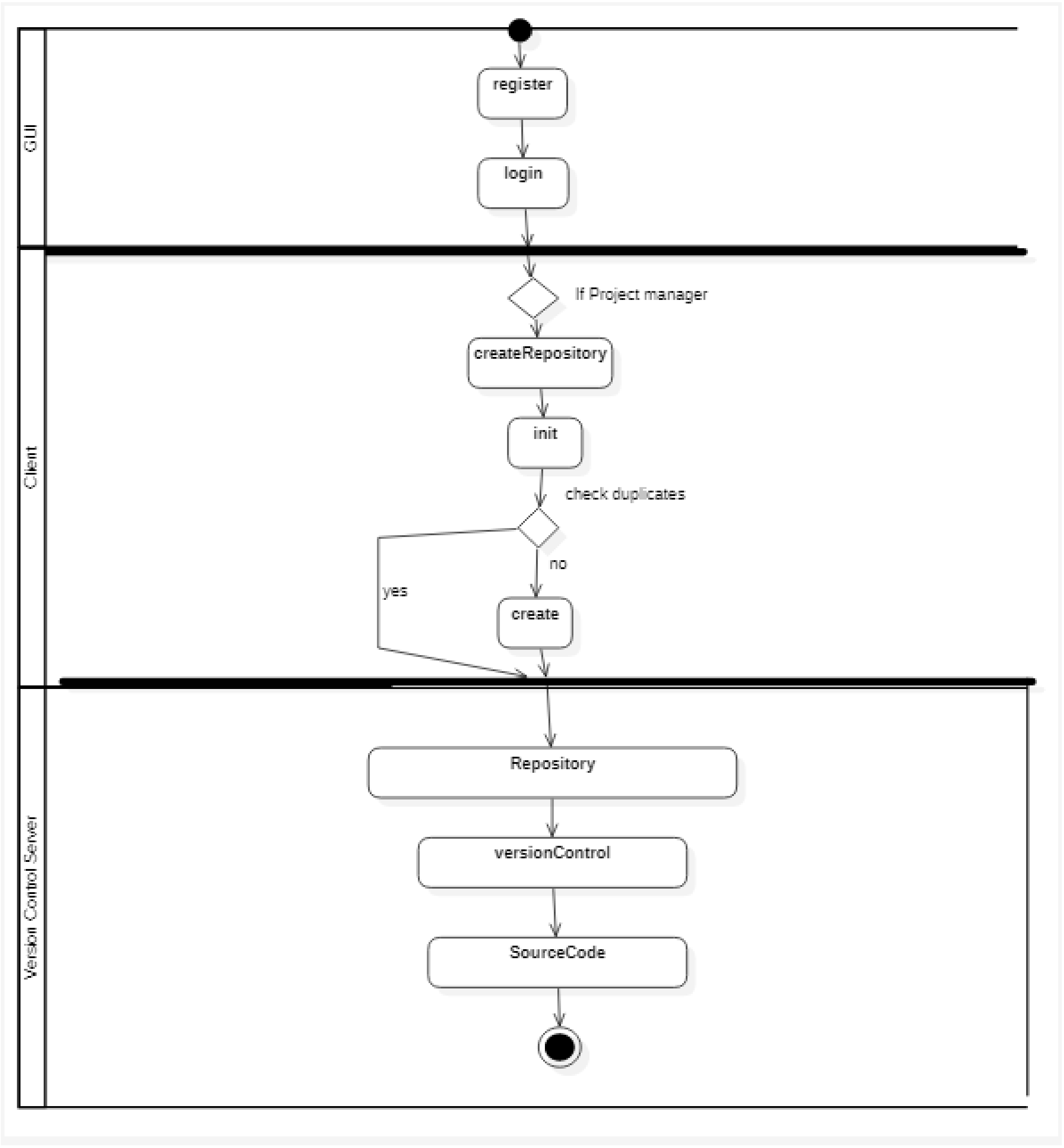


Arch-1

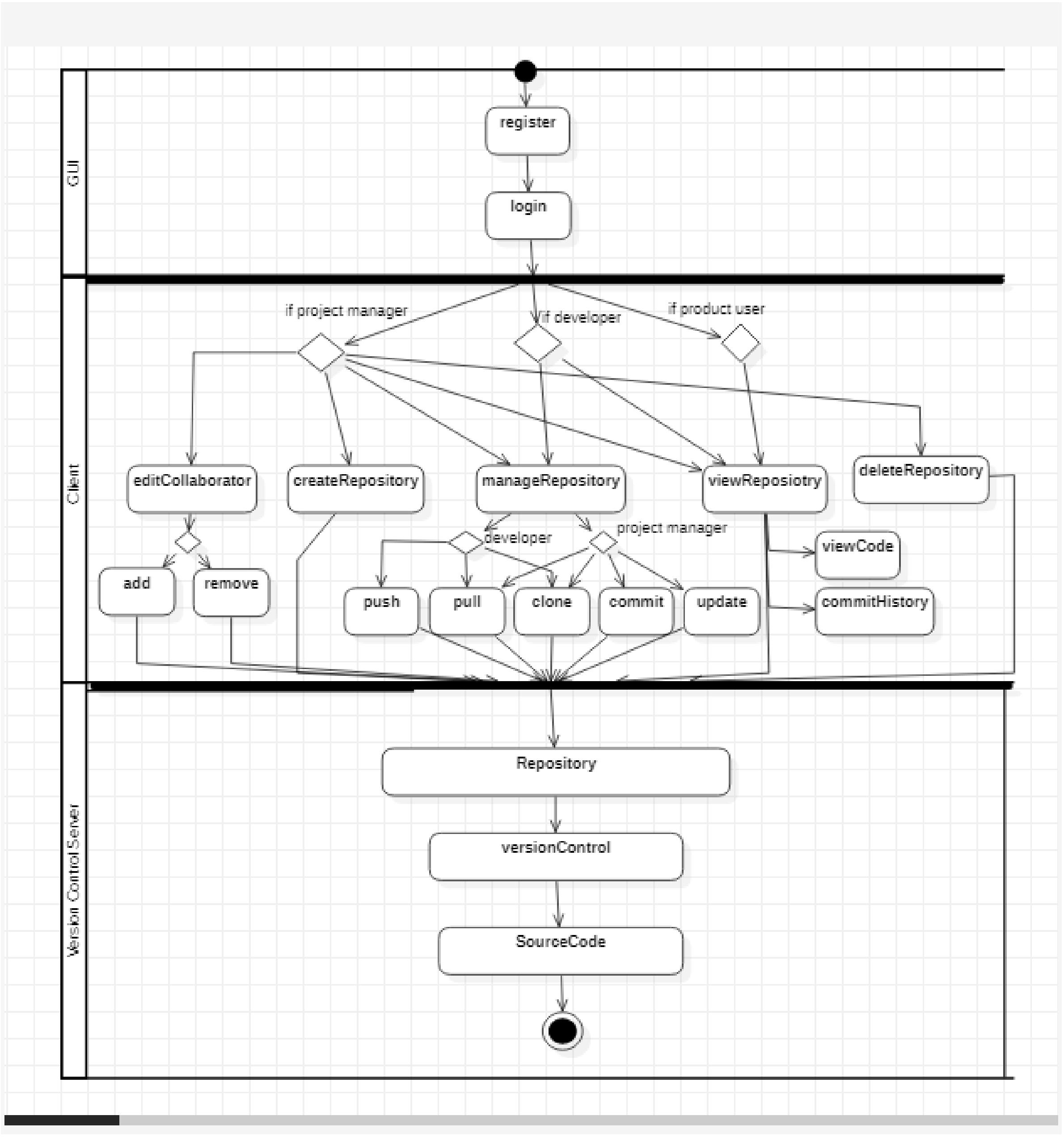
Activity Diagram

Create Repository (AD-1)

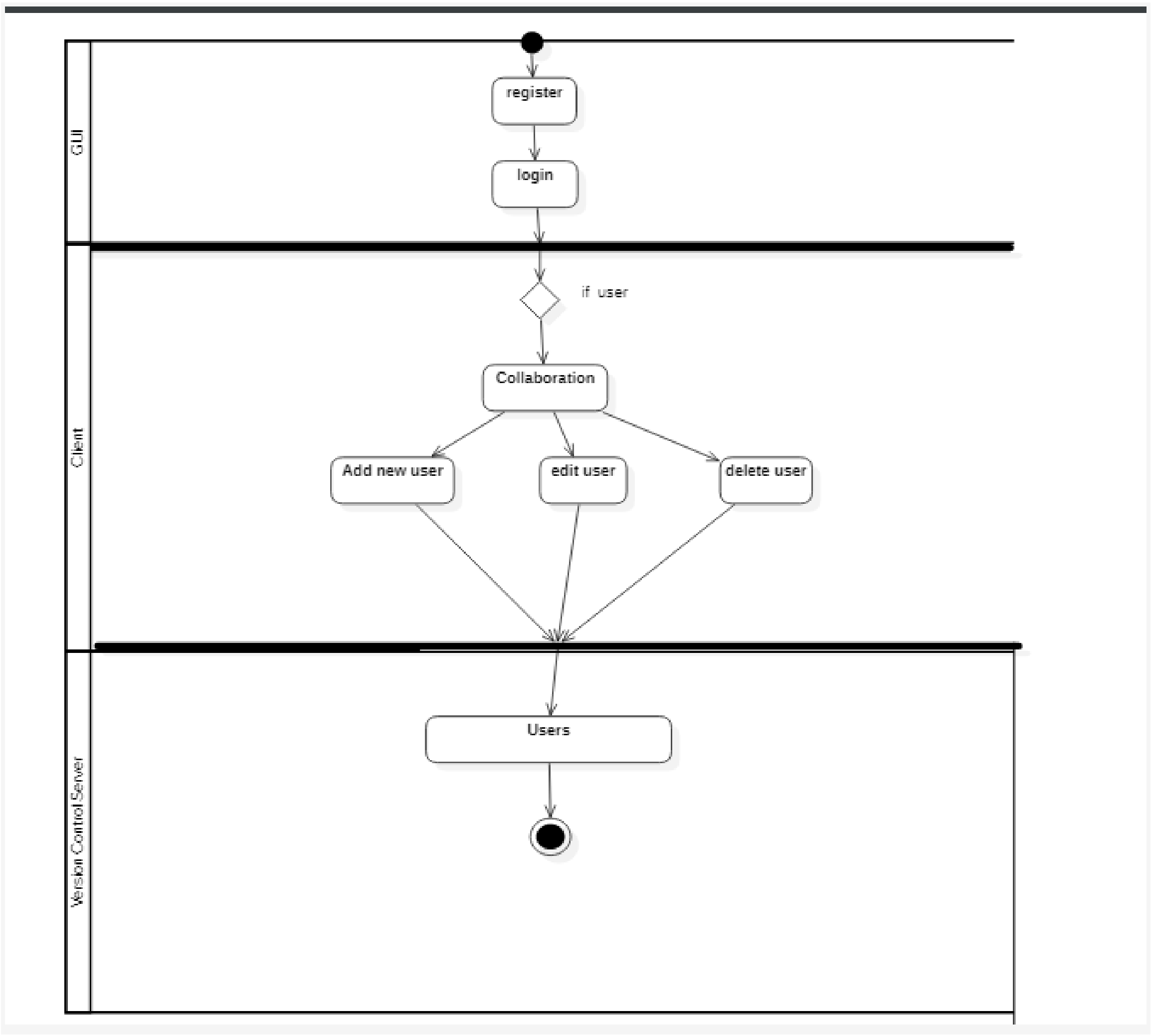
Manage Code(AD-2)


GUI
Client
Version Control Server
register
login
If Project manager
createRepository
init
check duplicates
yes
no
create
Repository
versionControl
SourceCode

Arch-2



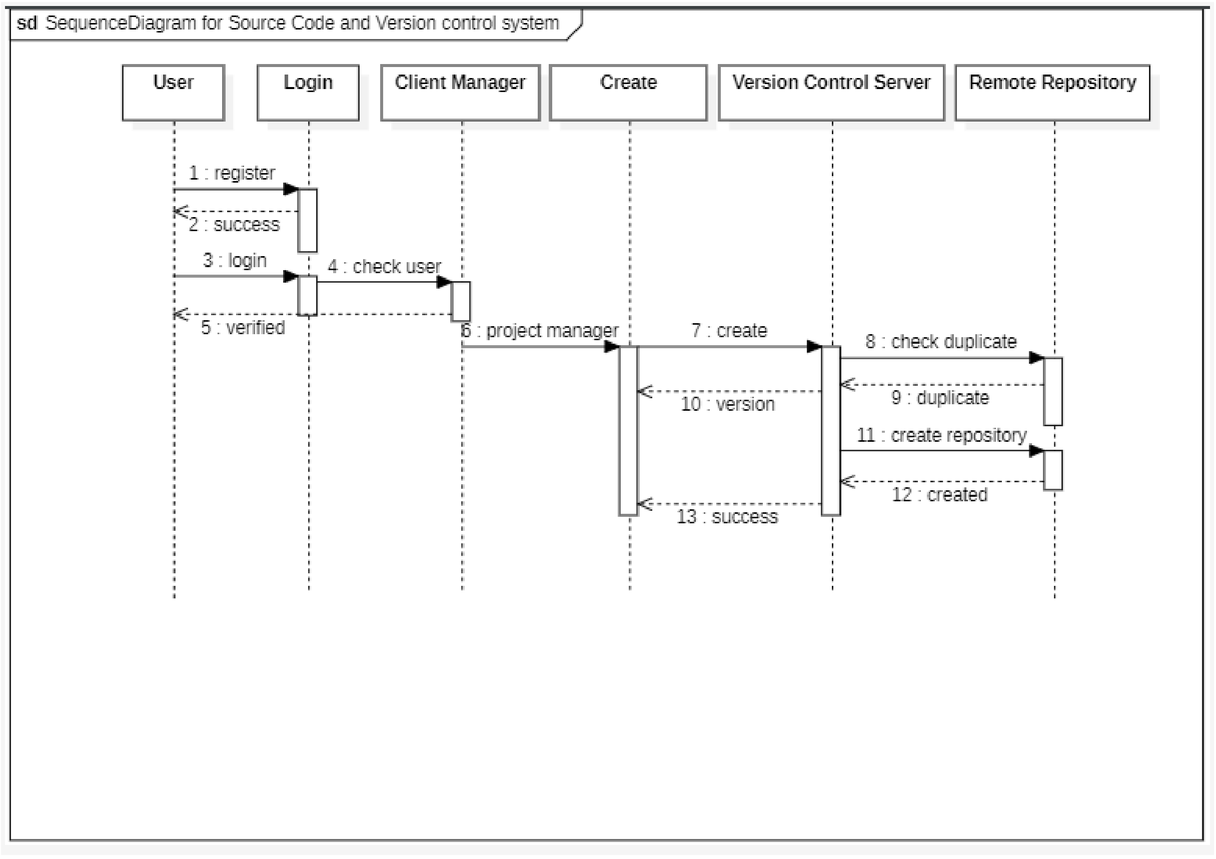
GUI
Client
Version Control Server
register
login
if project manager
if developer
if product user
editCollaborator
createRepository
manageRepository
viewReposiotry
deleteRepository
add
remove
developer
project manager
push
pull
clone
commit
update
viewCode
commitHistory
Repository
versionControl
SourceCode



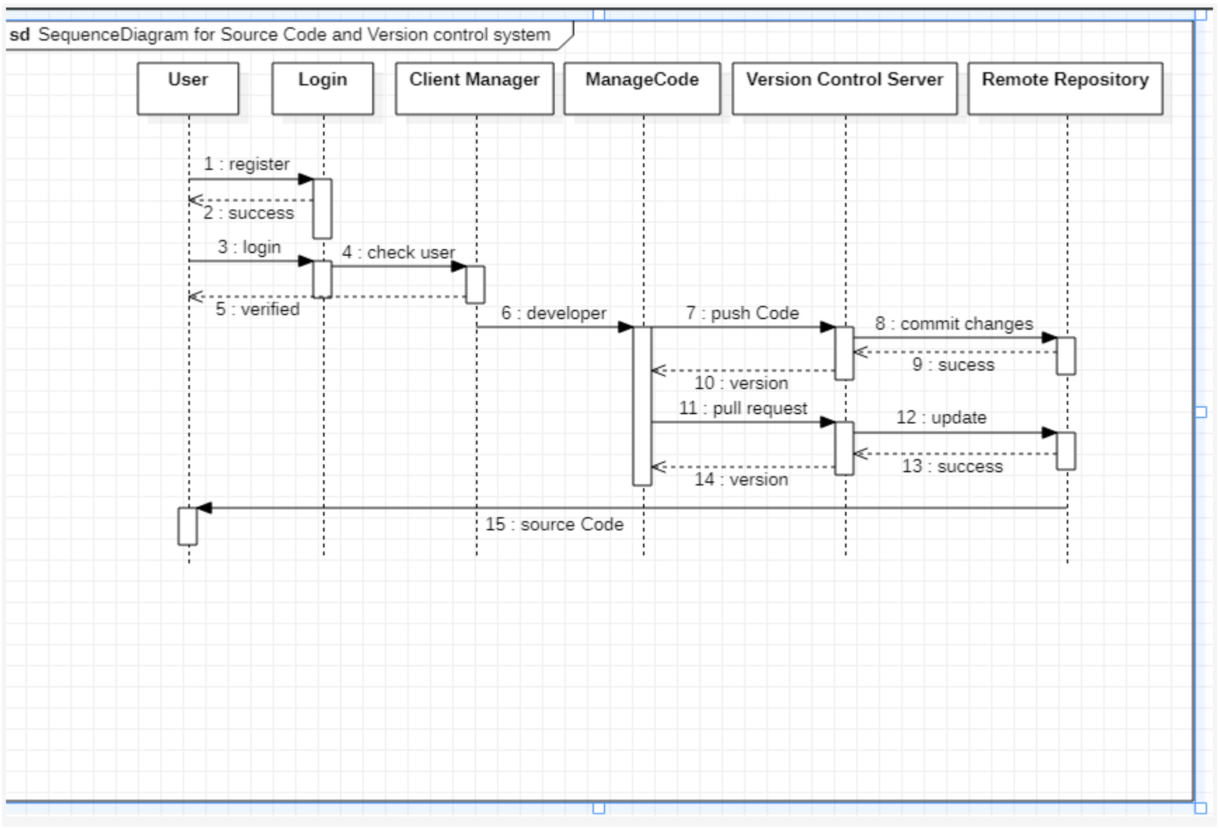
Collaboration (AD-3) Arch-3

Sequence Diagram

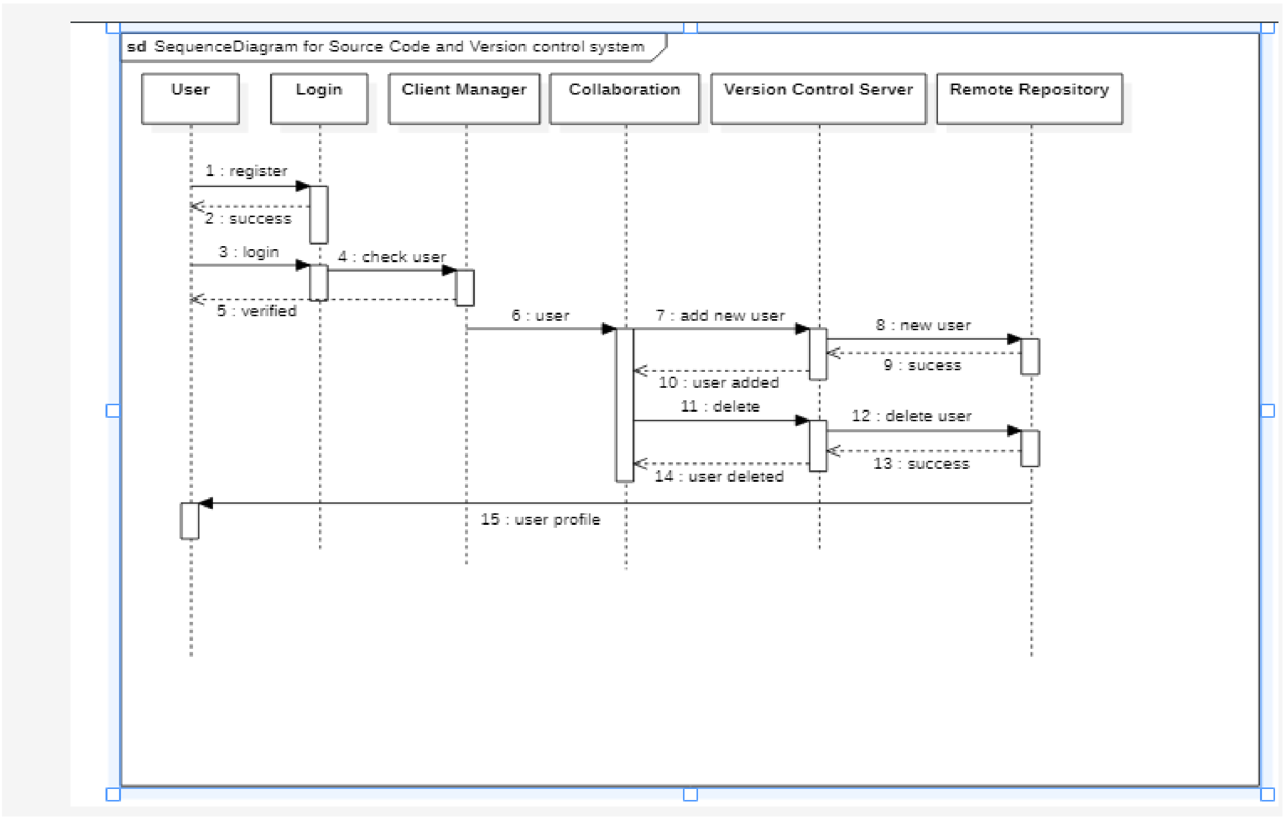
Create Repository(SD-1)



Manage Code (SD-2)



Collaboration (SD-3)



MODULE DESCRIPTION

1.Create Repository

In our project first module is create repository here user wants to create repository they have to login first by using username and password for this we created login page here administrator as project manager and he has control access over all user data and file and he can add,delete,edit the user and its profile.

If a user can create a repository after login here we have three features. Firstly, the user can create a new repository. secondly if the repository is already created then the user wants to create a repository within the repository, thirdly it checks the name of the repository already exists it will error message. Users can add source code or file within the repository. and we have logout to exit from the site.

2.Manage Code

We have 2 types of user Admin for project manager and Non-Admin for developers.Main features are to view, rename,delete repositories and can push code and do pull requests.Users can add multiple files and any file format.User can rename files,delete files.Each file is added timestamp is recorded and helps in Version code and added to Remote Repository. User can do pull request to check files and compare files and add different versions to Repositories.To view the file user can clone the file and can be viewed in the system.

3.Collaboration

This feature is used to manage users or collaborators.Only Admin users can add new users and edit the user settings and also can delete users.Share code is the important feature in this module,where users can share code to all users while uploading the source code files.All the code files can be downloaded and can add new version to the Repository.Finally we have Dashboard which displays number of Repositories and users and files in remote repository.

TEST CASES

1.Create Repository

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
U101	Login system	To test the login functionality	Access to View repository	1: start site on localhost 2: Enter Username and Password 3: Click Submit	User name: admin Password: admin123	Login should be successful with "valid username or password" message	Login successful	Pass
U102	Validating login	To check the valid or invalid username and password	Access to View repository	1.Enter Username and Password 2. Click Submit	User name: xxxx Password: xx123	Login should be unsuccessful if invalid username or password	Login unsuccessful	Pass
U103	Edit, Rename, Delete users	To edit, rename, delete users	-	1.Login as administrator(project manager) 2.in menu click on users 3.then administrator only can do these operations	-	If any changes has been done that displayed	must display any changes done by administrator	Pass
U104	Access to user profile	To Access to user profile	-	1.Login with admin	-	login with Admin only have access to user profile which shown on menu	Display user profile in menu	Pass
U105	Create Repository	To Create repository	-	1.Login as user 2. click on create repository 3.create repository with some name	sample repository	Displays created repository	Displays created repository	Pass
U106	Create Repository within Repository	To Create Repository within Repository	-	1.Login as user 2.if repository is created open that repository 3.click on create new repository 4.create repository with some name other than existing repository name.	sample1 repository	Displays new repository within existing repository if same name repository is created shows "name already exist" message.	Displays new repository within existing repository	Pass
U107	Add files	To Add files like source code within repository	-	1.Click on Add file button	sample.pdf	It should Added the selected file	It should Added the selected file	Pass

U108	Rename and Delete files/Repository	To Rename And delete files/repository	-	1.Double Click on file which has to be renamed 2.It will show menu rename and delete files.	-	If you click on rename then it will be renamed by the user using some name. and if you click on delete file will be deleted.	It must show the renamed file and if it deleted the file it did not show that file.	Pass
U109	Logout	To exit from account(logout)	User save all repository	1.click on logout	-	If any user login they can logout from account	it show login page	Pass

2.Manage Code

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
U201	User Login System for Non admin Users	To test the login of non admin users	Access to Repositories	1: Start site on localhost 2: Enter Username and Password 3: Click Login	User name: yathish Password: yathish27	Login should be successful with “valid username or password” message	Login successful with “logging in” message displayed	Pass
U202	View, Rename Repository	To view and rename particular repository	Access to Repositories	1: Double click on Repository. 2: Click Rename	-	Repository renamed.	Successful renamed.	Pass
U203	Search	To search the transactions based on the company	After Login	1.Type the Repository name that you want to view. 2.Click on the search button.	Sample	Must not display if that company is not present	Must display if that company is present	Pass
U204	Check file compatibility (.py,.png,.csv)	To check supported file formats.	After Login	1.Click on upload source code. 2. Browse any files to upload	py1.py	Accept the required file format	File format is successfully added	Pass
U205	Push code/file to Repository	To push code to repository from local repository to remote repository	After Login	1.After file/source code upload, select push to remote repository. 2. Add description of versions 2. Click commit changes	-	Source code must be push into the repository	File containing code successfully added with message “File added successfully”	Pass
U206	Pull request	To pull changes from remote to local repository	After Login	1.Click Pull request button 2. Add file that you want pull request 3.Click Save	py1.py	Accept the pull request of the source code.	Displays “Pull Request successfully done." message	Pass

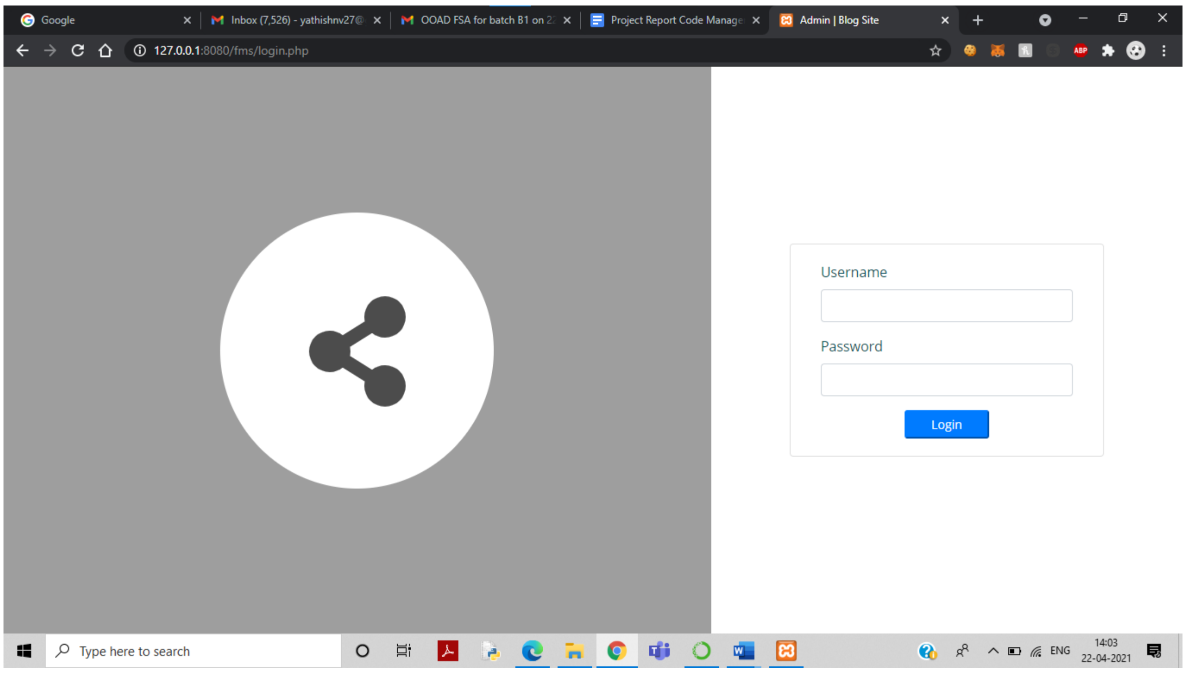
U207	Rename, Delete files	To rename and delete a particular file	Access to Source Files	1: Double click on File 2: Click Rename to rename and Delete to delete source file	-	File renamed. File must be deleted after selecting delete.	Successfully Renamed. Successfully Deleted	Pass
U208	Clone	To download the source file	Access to Source Files	1. Double click on File. 2. Click on Clone	-	File must be downloaded	File downloaded successfully	Pass
U209	Delete Repository	To delete repository	Access to Repositories	1: Double click on Repository. 2: Click Delete to delete repository		Repository must be deleted.	Repository deleted successfully.	Pass
U210	Logout from Non-Admin user	To exit from User account		1.Click Logout Button		Logout from the account	Logout successfully and back to login page	Pass

3.Collaboration

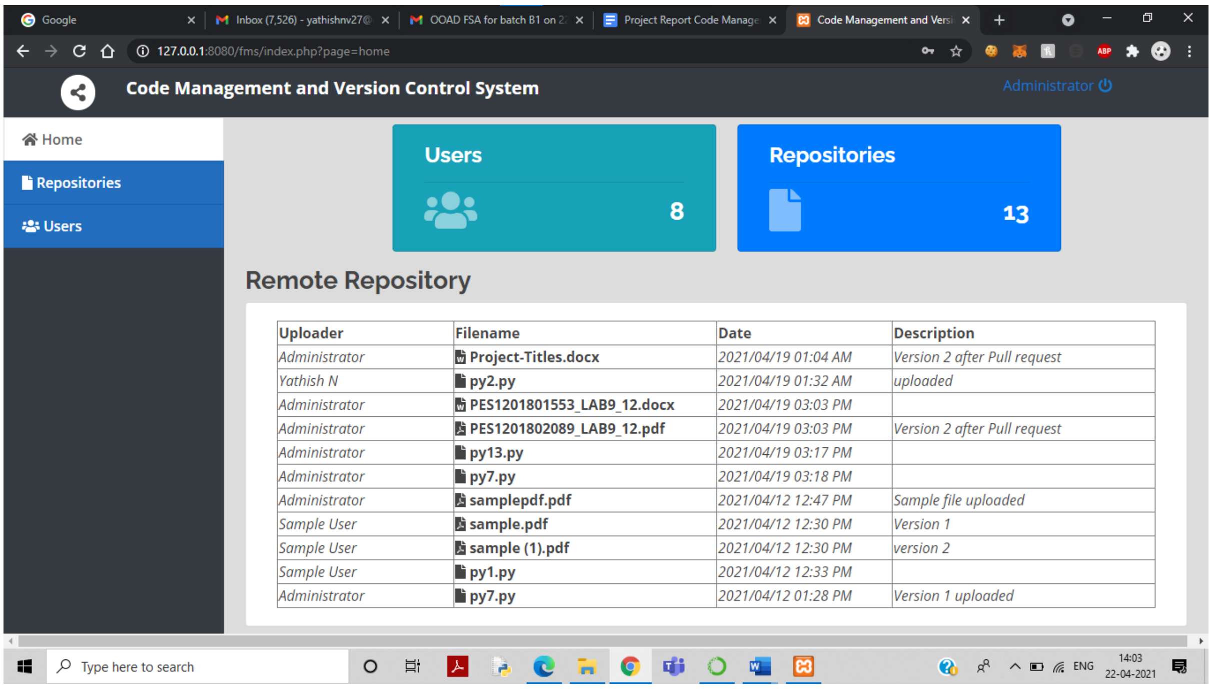
Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
U301	Add New User/collaborator	To add a new user/collaborator.		1: Click Users section 2: Click on New User 3.Create user with username and password	Username: vijay Password: 12345	New user must be added to the table of users	User created successfully	Pass
U302	Login with new user credentials	To login with a new user created.	New User must be created	1: Click Logout 2: Enter user details and Login	-	Login should be successful with "valid username or password" message	Login successful with "logging in" message displayed	Pass
U303	Edit new user	To edit and rename new user	-	1: Click Users section 2: Click on Actions 3.Select Edit option	-	User details must be changed.	Changes done successfully	Pass
U304	Delete new user	To delete new user	-	1: Click Users section 2: Click on Actions 3.Select Delete option	-	User must be deleted	No changes in users	Fail
U305	Download files from Remote Repository	To download any file from remote repository	-	1.Double Click on any file 2.Select Download	-	File must be downloaded	File downloaded successfully	Pass
U306	Share code to all users	To share code/file to all users including non admins	-	1.Click upload source code button. 2.Upload the desired file. 3.Select shares to all users and save.	-	File must be shared to all users	File is available to all users.	Pass
U307	Delete shared code from remote repository	To delete shared code.	-	1.Click on delete button	-	Shared File must be deleted	File deleted successfully and not available in Remote Repository.	Pass

U30 8	View changes History	To view files and details of changes in table		1.Click Repositories section. 2.View table of any Repository	-	File details must be present	Table is viewed successfully.	Pass
U30 9	View Users and Repositories	To view number of users and repositories		1.Click Home		Users and number of Repositories	Numbers are present in Home	Pass
U31 0	Logout from new user	To logout		1.Click Logout Button		Logout	Logout successful	Pass

SCREENSHOTS OF OUTPUT



Google
Inbox (7,526) - yathishnv27@
OOAD FSA for batch B1 on 2
Project Report Code Manage
Admin | Blog Site
127.0.0.1:8080/fms/login.php
Username
Password
Login
Type here to search
ENG
14:03
22-04-2021


Google
Inbox (7,526) - yathishnv27@
OOAD FSA for batch B1 on 2
Project Report Code Manage
Code Management and Versi
127.0.0.1:8080/fms/index.php?page=home
Type here to search
ENG
14:03
22-04-2021

Code Management and Version Control System

Administrator

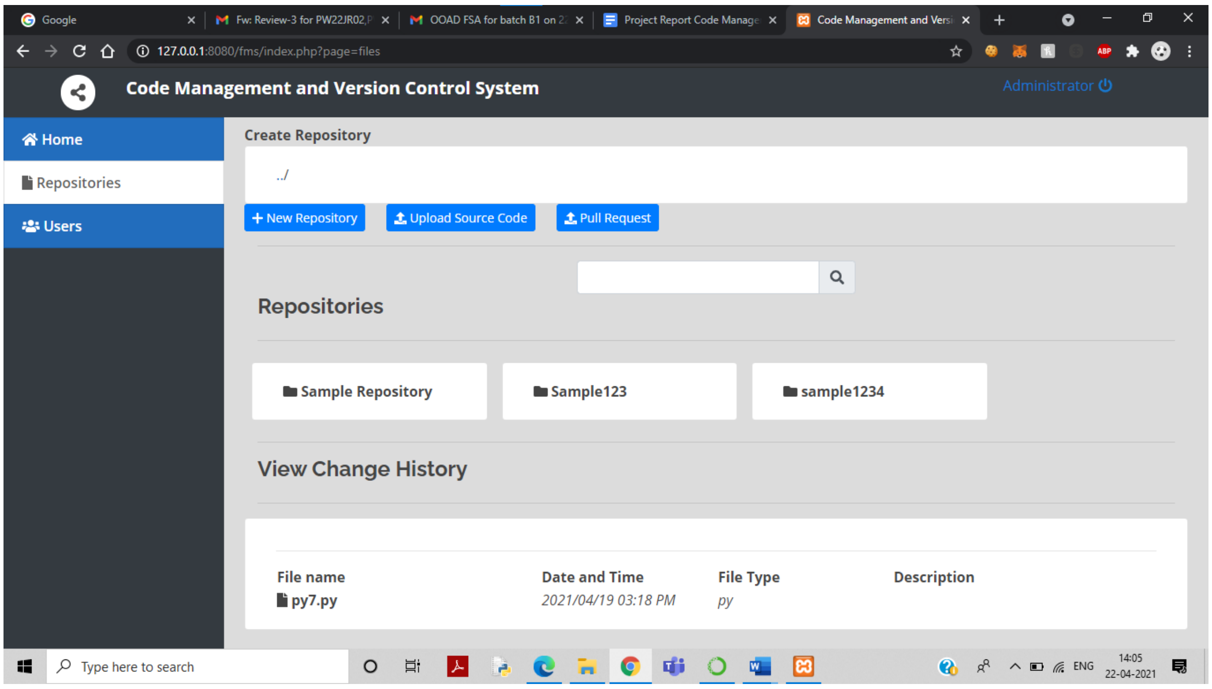
Home
Repositories
Users

Users 8

Repositories 13

Remote Repository

Uploader	Filename	Date	Description
Administrator	**Project-Titles.docx**	*2021/04/19 01:04 AM*	*Version 2 after Pull request*
Yathish N	**py2.py**	*2021/04/19 01:32 AM*	*uploaded*
Administrator	**PES1201801553_LAB9_12.docx**	*2021/04/19 03:03 PM*	
Administrator	**PES1201802089_LAB9_12.pdf**	*2021/04/19 03:03 PM*	*Version 2 after Pull request*
Administrator	**py13.py**	*2021/04/19 03:17 PM*	
Administrator	**py7.py**	*2021/04/19 03:18 PM*	
Administrator	**samplepdf.pdf**	*2021/04/12 12:47 PM*	*Sample file uploaded*
Sample User	**sample.pdf**	*2021/04/12 12:30 PM*	*Version 1*
Sample User	**sample (1).pdf**	*2021/04/12 12:30 PM*	*version 2*
Sample User	**py1.py**	*2021/04/12 12:33 PM*	
Administrator	**py7.py**	*2021/04/12 01:28 PM*	*Version 1 uploaded*


Google
Fw: Review-3 for PW22JR02_P
OOAD FSA for batch B1 on 2
Project Report Code Manage
Code Management and Versi
127.0.0.1:8080/fms/index.php?page=files
Type here to search
ENG
14:05
22-04-2021

Code Management and Version Control System

Administrator

Home
Repositories
Users

Create Repository

../

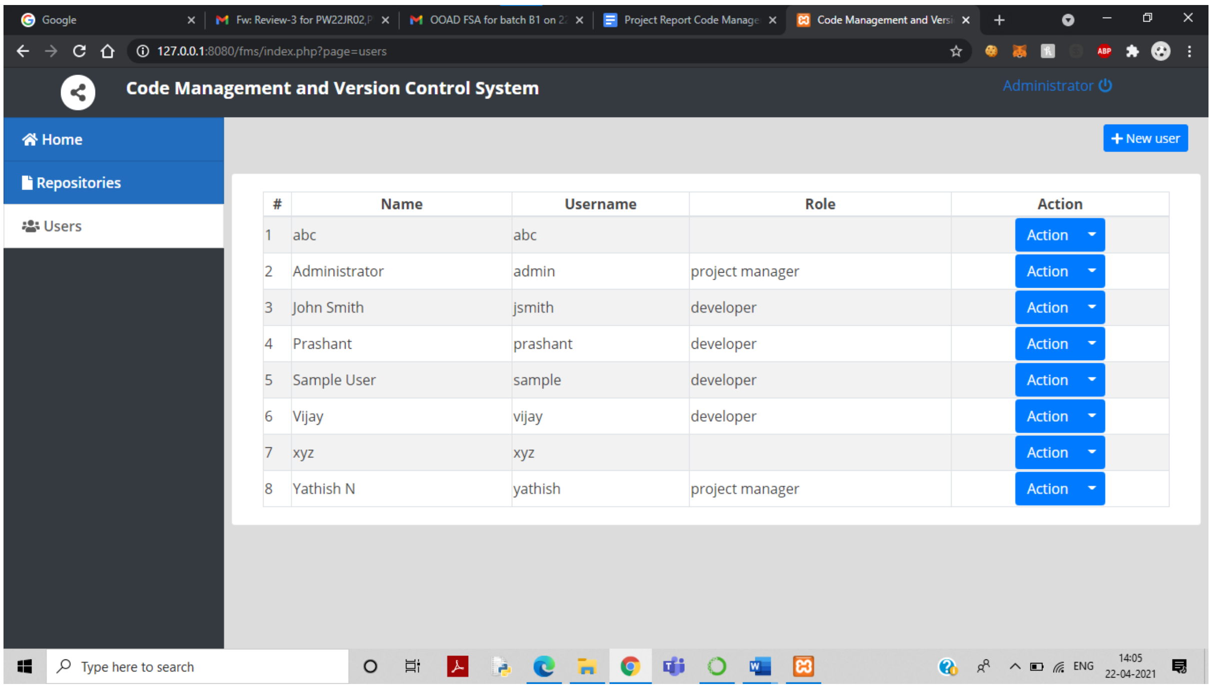
+ New Repository | Upload Source Code | Pull Request

Repositories

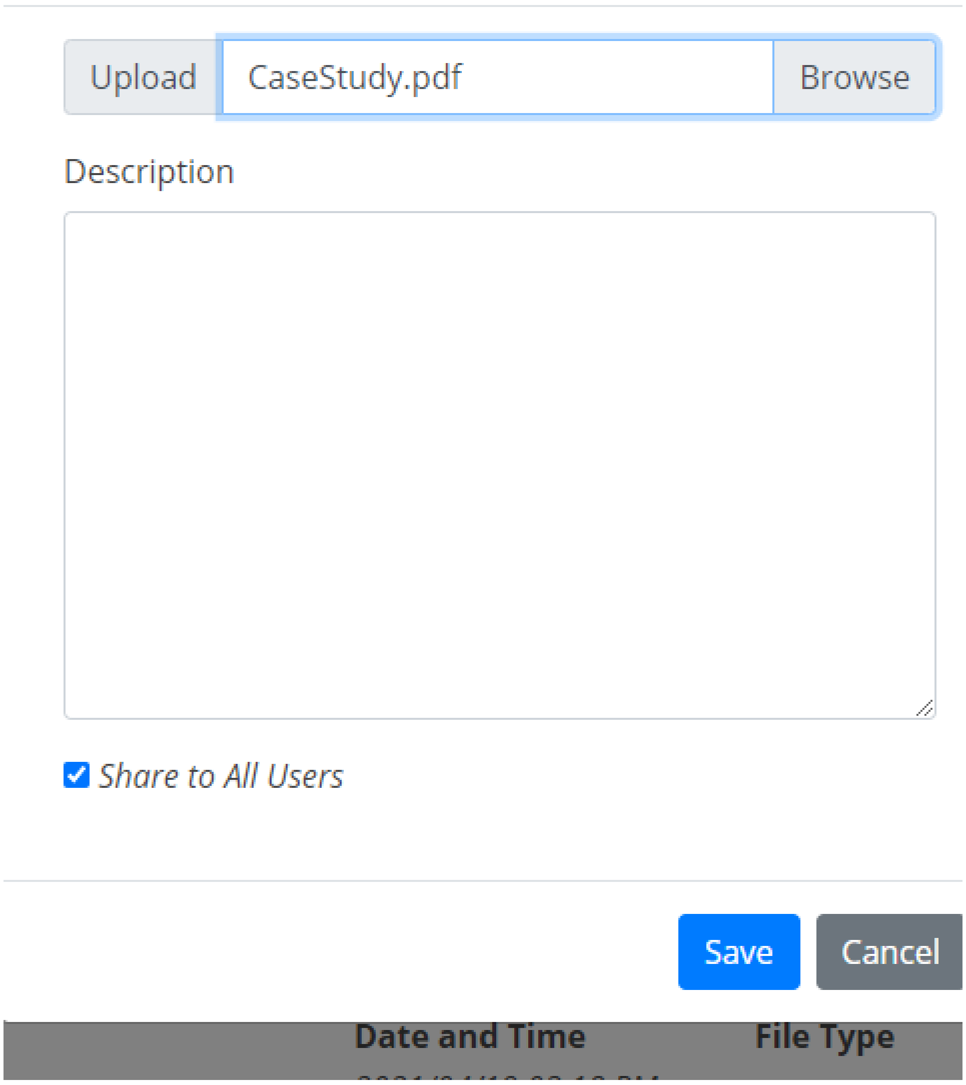
Sample Repository | Sample123 | sample1234

View Change History

File name	Date and Time	File Type	Description
py7.py	*2021/04/19 03:18 PM*	*py*	


127.0.0.1:8080/fms/index.php?page=users
Code Management and Version Control System
Administrator
Home
Repositories
Users
New user

#	Name	Username	Role	Action
1	abc	abc		Action
2	Administrator	admin	project manager	Action
3	John Smith	jsmith	developer	Action
4	Prashant	prashant	developer	Action
5	Sample User	sample	developer	Action
6	Vijay	vijay	developer	Action
7	xyz	xyz		Action
8	Yathish N	yathish	project manager	Action



Upload: CaseStudy.pdf

Description

[✓ Share to All Users]