# Project Plan

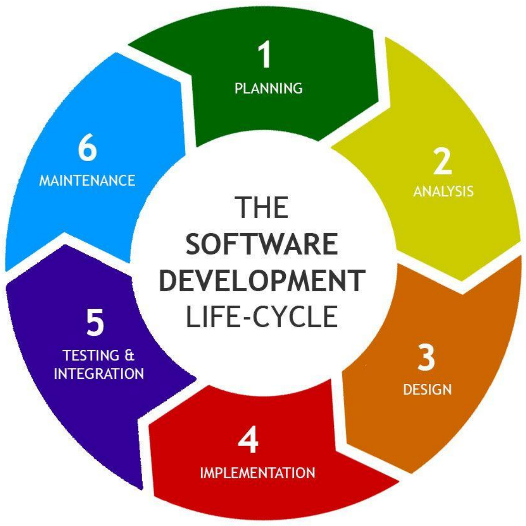
for

# Code Management and Version Control System

**Project ID:** A16
**Section:** A

**Prepared by**
Arjun Chengappa (PES1201800119)
Lavitra Kshitij Madan (PES1201800137)
Vishnu Charan Golugula (PES1201801230)

# Lifecycle



## 1) Requirements Gathering and Analysis

- **Hybrid Model** can be used here since the project consists of components which are dependent of previous phases of development (Repository Management and Profile Management are dependent on User Auth and access control) as well as components that can be worked on independently (Comments section, Search, Database creation)
- The end user requires a web app for a code management system with features like sharing files, managing repositories, committing changes , and tracking issues.
- Inputs by the developer team should include feature approval and modification, development time specifications, tech stack constraints etc.
- Upon approval, the SRS Document is created. Once the final SRS document is approved, the project moves on to the design phase.

## 2) Design

- Feature and functional requirements are specified and finalized.
- Mockups of the intended platform are created by the design team and are reviewed by developers, product managers and a small test set of customers. Use prototyping tools such as Bootstrap Studio to create design mockups.
- The test set of customers further give their feedback which are added onto the design according to the needs of the majority.

- These design ideas are constantly improved upon through various numbers of phases.

## 3) Implementation & Testing

- Provide updates to product managers.
- A test-driven approach is followed where a feature or a part of it is created, tested, reviewed and deployed.
- Frequent updates are provided and they are tested for the same using tools such as Postman to verify that each of the API calls are responded to as intended.

## 4) Deployment

- Deployment in software or web development means to push changes or updates from one deployment environment to another.
- After the web application is production-ready, you want the ability to make changes to the web application in real-time. Such environments are called production environments, or deployment environments.
- The initial development environments will essentially be set up as a local environment. Such environments are called development or staging environments. These are typically non-production-ready.

## 5) Maintenance

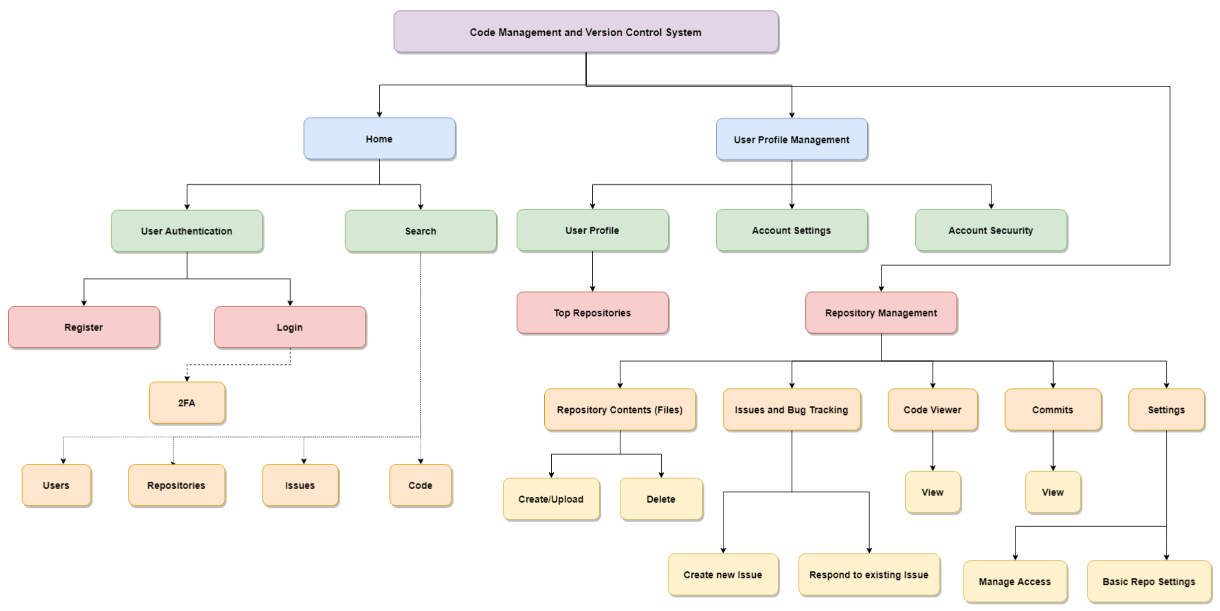
- The Maintenance Phase occurs once the system is operational.
- It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location.

## Tools

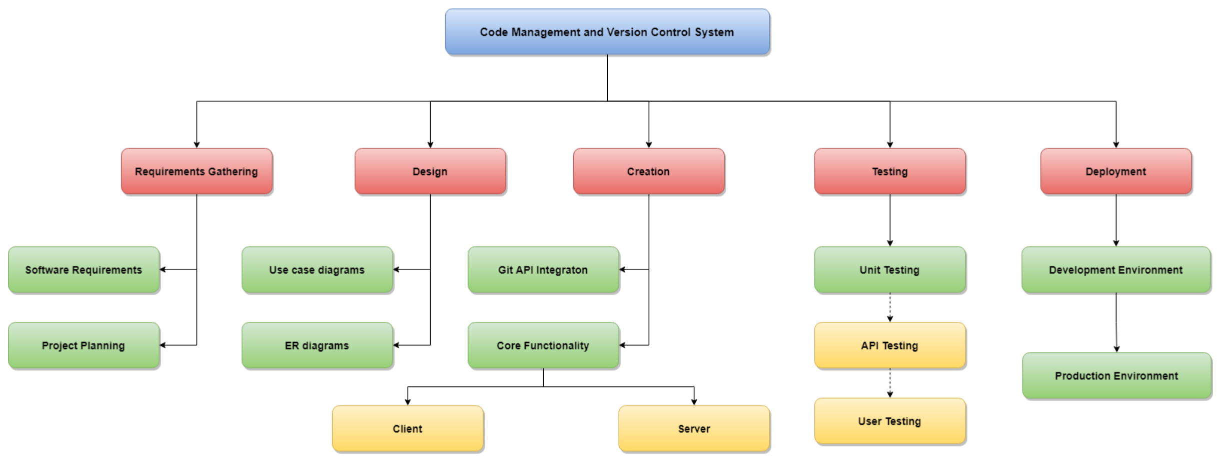
| **Project Phase** | **Tools Required** |
|---|---|
| Analysis and Planning | Draw.io, StarUML, Google Sheets, Google Docs |
| Detailed Design | Google Sheets, Google Docs, StarUML, Bootstrap Studio |
| Code Review | Atom |
| Implementation | Atom, Flask, HTML, CSS, JavaScript, Bootstrap, Cryptography.Fernet, SQLAlchemy |
| Testing | Postman, Google Chrome |
| Version Control | Git |

# Work Breakdown Structure

## Functionality WBS



## Workflow WBS



## Determine all the deliverables and categorise them as reuse/build components and justify the same.

### Deliverables:

| **Build** | **Reuse** |
|---|---|
| User Authentication | Search bar |
| Repository Management | Code viewer |
| Profile Management | Comments section |
| CRUD operations on database | |

### Reusable components:

- The search bar is universally accessible throughout the website and developers will only need to use a vendor such as HubSpot or Google to implement the search functionality once on the landing page. This component can be reused across the website
- The code viewer need not be implemented from scratch. Editing existing implementations of web-based source code viewers to view files across the code management system will suffice.
- The comments section will again be common to the “issues” section and the “commit” section and can thus be reused.

All other components of the web app involving user authentication, 2FA, repository-specific settings, implementing commits to the repo resulting in updation of the database and managing user profiles will need to be built from scratch.

## Do a rough estimate of effort required to accomplish each task in days.

Assuming a deadline after 12 weeks, given below is a rough estimate of the effort required to accomplish each task in days:

1. Planning and Learning Phase
    a. Basic idea of a code management and version control system. **[4 days]**
    b. Deciding on the tech stack to be used to build the platform. **[4 days]**
    c. Going through documentations for the tech stack chosen and the required JavaScript libraries. **[7 days]**
    d. Deciding on the overall application design. **[3 days]**
2. Implement the register and login pages vital for user authentication along with placeholders for the remaining components. **[7 days]**
3. Implement the user dashboard which has a list of the user's top repositories along with a search bar at the top of the page allowing users to search for public code, issues, repositories and other users on the platform. **[7 days]**
4. Implement the repository-specific dashboard (Create and delete files, commit list, and repository-specific settings). (excluding issues and bug tracking) **[7 days]**
5. Fully implement issues (along with the Q&A section where fellow developers can discuss on the concerned issues) and bug tracking under the repository-specific dashboard. **[7 days]**
6. Implement the code viewer allowing users to view the code contained in a file under the repositories. **[4 days]**
7. Implement user-specific settings. **[4 days]**
8. Testing [**2 days]**
9. Fixing critical bugs (if encountered upon testing). **[3 days]**
10. Deployment **[1 day]**

## Create the Gantt chart for scheduling the previously defined tasks.

Code Management and Version Control System Gantt Chart