**TEAM 2**

**Project Name**: GitHub Repository

**Team Members:**

013530530 - Devanshi Trivedi

013858819 - Keerthi Akella

013820599 - Nithin Gollanapally

012050571 - Thinh Manh

013829179 - Rajesh Thummala

013854295 - Siddarth Varanasi

* **Miniworld:**

GitHub repository world is a place for everyone to share their code from different platforms and programming languages. In other word, it is a version control tool for coders and developers. In there, people can come together and work on a same project individually. They will be able to make edits for the code that has been shared thus allowing them to work on their part of the project. When they finished their edit, they can merge all the versions into one.

Many times, it happens that the programmer wants to revert to a change he made. GitHub makes this possible by providing a feature called commits. Whenever a coder makes necessary changes, he commits the code. Each commit marks the changes in the code made at that point in time. In our project we are going to store the whole code at that timestamp and not the changes.

Coders can also follow each other and can be inspired by others work. They can post their projects which can be viewed by people.

* **Purpose of Application/database and Intended Users:**

The purpose of application is creating a database that store the different versions of source code. Thus, the developers, coders, recruiters and students who are the main users of the application can easily retrieve code from the database to review their own work or share it with other people. Besides, the database helps developer keep track of their versions.

It brings together coders from all over the world to develop new ideas and work on it together.

**Intended users:**

The intended users are coders, developers and teams working on same project in the enterprise. Each person who uses GitHub has their own user account protected by an encrypted password, and can create and manage unlimited public and private repositories.

* **Objects/Actors/Roles:**

**Objects:**

1. Users
2. Repositories
3. Project Files
4. Branches
5. Pull requests
6. Commits

**Actors and Roles:**

1. **User**

Creates a private/public repository. Each repository has project files. If it is a team work then he adds other members as repository members. Each user can follow one or more users.

1. **Repository Member**

All members along with the repository creator can make branches to the repository and can make a code pull request from a branch and commit the edited code to a branch.

* **Planned functionality, operations:**

1. **Sign up**

A user can sign up for a Github account with an encrypted password.

1. **Login**

A user logs into his account with an encrypted password.

1. **Follow users**

Each user can follow one or more users.

1. **Create a Repository**

A user can create a repository which can be public or private. A private repository is not visible to other users.

1. **Adding repository members**

For a group project, the user who created the repository can add his team members as repository members who can make changes to the repository.

1. **Adding Project files**

Each repository has project files, which contains the project related details and source code.

1. **Creating branches**

The repository members can create as many branches of the repository as he wants. Usually, each repository member is assigned a branch. The code in each branch is separate and so all members can work on same project without editing each other’s work.

1. **Pull Request**

A repository member can make a pull request from a particular branch to take the code into his local repository.

1. **Commit Code**

Each repository member can commit his code to a particular branch of the repository with a comment.

* **Scenarios:**

Sam is the manager of an enterprise and is assigned a project to create an IOT based application to detect real time weather changes at SJSU. He has 4 subordinates who are going to work on this project with him.

He decides to break down the problem into multiple modules and divide the work between all the team members. Now some modules are interrelated which are assigned to different members. How to manage this so that all the members can code independently without editing the others work and at the same time they can view changes made by the others? Here is when Sam decides to take advantage of the features provided by Github.

He signs up for a Github account and logs into it. He then creates a repository which will have all the project files related to that repository.

Sam does not want anyone to have access to this code so he makes the repository private and adds his team members as repository members. So now only the team members have access to this repository.

He then creates branches of the repository for each repository member so that they can code independently and at the same time view others work thus solving the problem he was facing earlier.

The manager uploads a start up code. Each repository member makes a pull request from their own branches to get the code into their local machine. They can each make changes and then can commit the changed code into their own branches with necessary comments.

At regular intervals, the manager merges the code of different branch into a separate branch to check if the code is running after integration.

At the end, after satisfactory completion of the results, the manager submits the final merged code from his branch.

(Our project does not handle the merge feature as it involves the use of an independent software which checks the difference between the codes and manipulates the code accordingly.)