**TEAM 2**

**Project Name**: GitHub Repository

**Team Members:**

Devanshi Trivedi (013530530)

Keerthi Akella (013858819)

Nithin Gollanapally (013820599)

Thinh Manh(012050571)

Rajesh Thummala (013829179)

Siddarth Varanasi (013854295)

**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 coder and developer. 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. When they finished their edit, they can merge all the version 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. Here a commit is done whenever some changes are made which can be accessed later.

Coders can also follow each other and can be inspired by others work. They can post their projects which can be accessed 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 which are the main user of the application can easily retrieve from the database to review their own code or share it with other people. Besides, the database helps developer keep track 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 personal users who have personal user accounts. Each person who uses GitHub has their own user account protected by an encrypted password, which includes access to unlimited public and private repositories.

**Objects/Actors/Roles:**

Objects: Repository, Project, Commits, Comments, Pull Requests

Actors and Roles:

1. User : The user can create a github account and manage a project. It can pull the code from github to his local desktop and perform commits on his code. He can also add comments with the commits that he makes.

2. Admin : Each repository and project has its own admin who can make it public or private and give edit rights to other users.

3. Followers : Each user is followed by many users and can follow many users.

**Planned functionality, operations:**

1. User: -

* Can create Account and login.
* Only Admin can edit a repository and a project.
* Admin can give access to other users for project(s) in a repository.
* User with access to a project can perform commit operations.
* Each user can perform a pull on a public project.
* Each user can perform a pull/commit on a private project if provided with access.

2. Repository:-

* Each repository has an admin.
* A repository can have one or more projects.
* It can be accessed by one or more users based on the permission given by admin.

3. Projects:-

* The user who creates a project is the project admin.
* It can be accessed by one or more users based on the permission given by admin.
* A project can be committed and pulled by user with access.

4. Followers:-

* Each user can have one or more followers.
* Each follower can follow one or more users.

5. Commits:-

* A user with access to a project can commit his code one or more times.
* Commit operation can be performed by one or more users who have access.

6. Pull Request:-

* A user with access to a project can pull code one or more times.
* Pull operation can be performed by one or more users who have access.

7. Comments:-

* A user can give his/her comments be he/she can commit code changes to project
* A project can have one or more comments.

**Scenarios**

**Users –** (user\_id, password(Hashed), name, email, created\_at(timestamp), type (admin or not), country, state, city)

1. Each user has unique user-ID and email ID

2. The date and time of user created will be saved in created\_at

3. The user can be admin or general user based on the permissions or access user has. (The admin of a repository can give permission to other users to access their repository/project)

4. The user’s address(Country, state and city) is asked for the first time when the user is signing up.

**Followers- (**user\_id, followers, follower\_created\_at)

1. A user can follow one or more users.

2. Each user has a unique user\_id and the number of users following that particular user.

3. The latest date at which the follower follows a user is stored in the **follower\_created\_at**.

**Repositories- (**repo\_id, repo\_url, repo\_owner\_id, repo\_name, repo\_description, repo\_created\_at)

1. A repository is a collection of projects. Many users can work on one repository.

2. Each repository is identified by a unique ID, repo\_id and has its own URL, repo\_url.

3. The user who create the repo is the owner indicated by repo\_owner\_id.

4. It also has its name and description and the timestamp it was created.

**Projects** – (project\_id, repo\_id, project\_url, project\_owner\_id, project\_name, project\_description, language, project\_created\_at, updated\_at, commit\_id)

1. Each Project in the repository will have a project id

2. Each project will be in certain repository.

3. A Project in a repository will have unique URL

4. Each project can only be created by owner of the repository and only he can define the project name, project description and the language to be used.

5. Each project will have created time, where it is created from and its updated time.

6. Commit\_id marks the files at every commit.

**Pull\_Requests- (**pull\_req\_id, project\_id, pull\_date)

Each pull request performed by user will have a pull request id and a project id to which the repository belongs and the time stamp at which pull request is performed.

**Commits** – (commit\_id, user\_id(a log is created), project\_id, author\_id, commit\_created\_at, file)

1. A commit is an individual change to a file (or a set of files).

2. When you save a file, every time a unique commit\_id (i.e, “SHA” or “Hash”) is created that marks the new file uploaded.

3. Commits also contains user\_id (id of the project owner), project\_id (each project has an id), author\_id (person who has committed), commit\_created\_at (created time).

4. File is the code submitted by the user.

**Commit comments (weak entity)** - (commit\_id, user\_id, body, comment\_id, commit\_comment\_created\_at)

1. Each commit comments will contain commit id which identify the commit. Next, it must have the user id since user id show the user who did comment on the commit.

2. The commit comments include body for the user to comment into.

3. Each comment will consist of comment\_id. There will be multiple comments in a commit, so it is important to recognize the location of each comment which is the attribute commit\_comment\_created\_at.

**Project members** – (repo\_id, user\_id, created\_at, project\_id)

They all are the users working on project. They are given permission by their respective project owners for the pull and commit operations. The other users who do not have permission cannot access the project.

The project members entity has the following attributes:

1. The **repo\_id** is the unique repository ID.

2. Each user has a unique **user\_id**.

3. The **created\_at** field is filled in with the latest date that the corresponding user or project has been created.

4. Each project has a unique **project\_id**.

.