SRS: GitGroup

# 4.REQUIREMENTS

## User requirements

### Functional requirements

UR-1: User shall be able to create a new project and new team in GitGroup.

UR-2: User shall be able to manage GitHub repositories and team members like add or remove repositories and members.

UR-3: User shall be able to use Kanban board to organize issues by dragging issue cards among several columns representing different stage of the development processes.

UR-4: User shall be able to analysis the development duration, quality and the team work situation of the project by using a variety of agile charts and analytics.

also manage tasks like suggesting a new idea or tracking a bug.

UR-5: User shall be able to organize an online meeting group chat, schedule and record the conference within the team or the global scope.

### Non-Function requirements

UR-6: The application shall be able to make GitHub easy to use, especially friendly for the person who has little development experience such as product manager.

UR-7: The application shall be quickly restored to operational status after a failure occurs.

UR-8: The app shall be reliable to uses with no downtime.

UR-9: The application shall be able to provide maximum security against malicious attack.

## System requirement

### User Interfaces (Functional Requirements)

UI-1.1: Developers can create a new project and make a team with other developers.

UI-1.2: Developers can remove a project and collaborators of the project after checking if the project is empty.

UI-2.1: Developers can manage their team by inviting to or removing collaborators from their projects.

UI-2.2: Developers can manage their repositories by adding or removing repositories of their projects.

UI-3.1: Developers can classify issues by different stages of development process via putting them in different columns of Kanban board.

UI-3.2: Developers can add or remove a Kanban from their Kanban board.

UI-3.3: Developers can customize their own stage by editing column name except Done column.

UI-3.4: Developers can close issues by dragging it to the Done column, which will automatically close issues in the GitHub.

UI-4.1: Project manager can track and communicate the progress of their projects by burn down and burn up charts.

UI-4.2: Project manager can measure how much work a team can used in eXtreme Programming and Scrum from throughput chart.

UI-4.3: Project manager can analysis the velocity of project going from velocity chart.

UI-5.1: Team leader can organize an online meeting group chat within their team.

UI-5.2: Developers can organize an online meeting group chat within the global.

UI-5.3: Developers can receive a meeting notification from team leader.

UI-5.4: Developers can set an alarm for notifying the coming events on a conference schedule form.

UI-5.5: Developers shall be able to record content of meeting on a meeting notebook.

### Back-end Interfaces(Functional Requirements)

BE-1.3:

### Non-Functional Requirements

NF-6.1: The user shall be able to use all the app functions without any kind of training. The average number of questions call about how to use app shall not exceed 10 per day.

NF-6.2: The app shall be available for any kind of mobile device and PC.

NF-7.1:

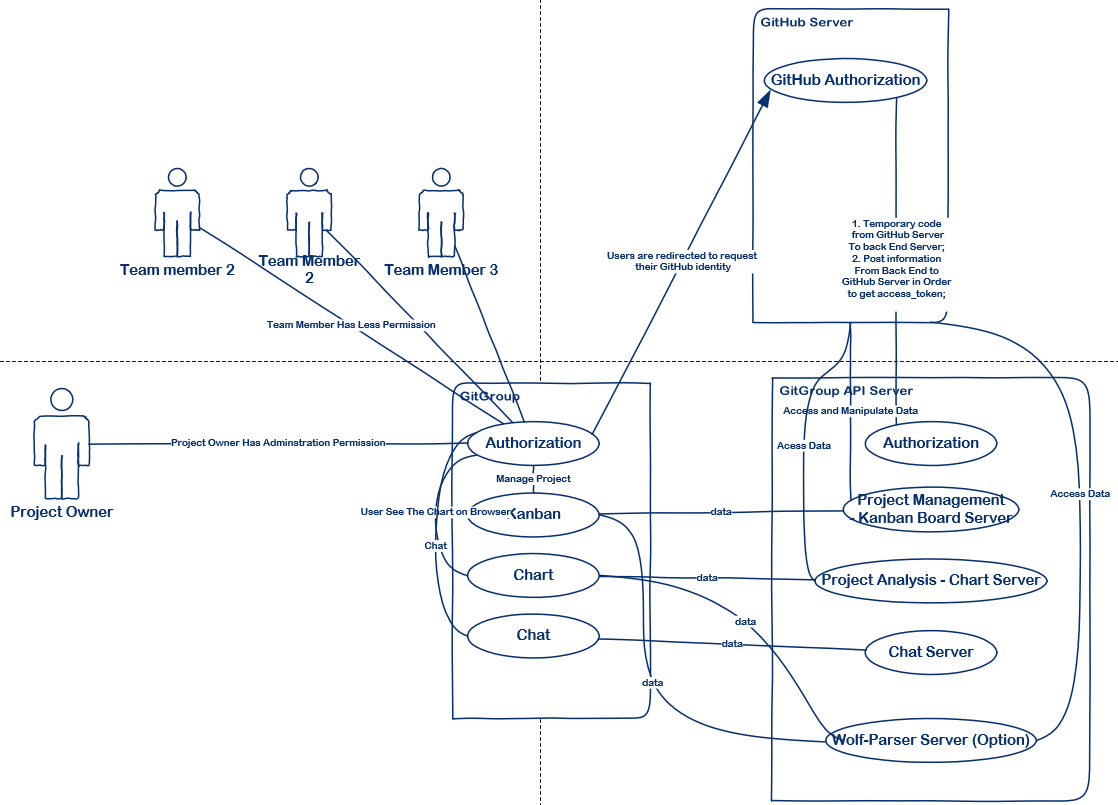
NF-8.1: The app shall be available to all users during whole day (Mon-Sun, 00:00~00:00)

NF-9.1: The app shall not expose contact information to other users.

NF-9.2: The app should minimize the amount of personally identifying information (PII) that it collects.

# 5. SYSTEM MODELS

## Use Case Diagram



User are redirected to request their GitHub identity when they sign in the GitGroup application and then get authorization from GitHub server.

After getting user identity, user can see several columns of Kanban board showing project process status of which data can be toke from Project Management Kanban Board Server and Wolf-Parser Server.

GitGroup API server get a temporary code GitHub server to back end server, and post information from back end to GitHub server in order to get access\_token.

Then project Management Kanban Board Server access and manipulate data from GitHub Server and show on the Kanban board.

Chart data is got from GitHub server and accessed by project analysis – chart server.

Online chat

## Activity Diagram

The user tries to log in and this generates an authentication request which uses GitHub to verify the identity of the user.

If it is an unregistered user, the user can sign up.

After successful login, the user is redirected to his/her dashboard. Here the user can manage their projects and team. The data is fetched from MongoDB.

The user has functionalities of creating a new project, removing a project after checking if it is empty.

After user creates a project, he/she has administration privilege of organizing a team to finish the software development, managing team members and GitHub repositories like adding or removing collaborators and repositories.

The major part of their dashboard is Kanban boar, which is a significant tool to help developer manage the development process. In the Kanban board, there are several columns represent different stage of the development process, such as TODO, Ready, In Progress, Done...

User can add or remove Kanban board and customize your own Kanban board column which can classify you cards of different development stage. Every column contains issue card imported from the GitHub, and you can easily drag them among the columns. If you drag it to the Done column. The issue in the GitHub will be automatically closed.

The second part of dashboard is four charts, Burndown and Burnup charts, Throughput chart, Users Throughput chart and Velocity Chart, to analysis the development duration, quality and the team work situation of the project.

The communication among team members can also be done in the page of the GitGroup. Team member can easily chat with each other after integrating the chat room into GitGroup. The issue cards are easily transported among the team member through the chat system. Team owner can send notification of online meeting chat. User can also start a group chat in the global scope.

User also has functionalities of viewing and editing his/her profile.

## Sequence Diagram

GitGroup acts on behalf of a user, it performs user-to-server requests. These requests must be authorized with a user's access token. User-to-server requests include requesting data for a user, like determining which repositories to display to a particular user. These requests also include actions triggered by a user, like running a build, in order to identifying and authorizing users on GitGroup, it must identify GitHub users when they visit GitGroup using OAuth.

When GitGroup user click authorization button they are redirected to request his/her GitHub identity. The web browser will send a request for GitHub access token to GitGroup server, and GitGroup server redirected to request a GitHub identity to GitHub Authorization Server.

Then GitHub redirects back to GitGroup authorization page, wait for user to accept. After user accept the request, web server submit the agree form to GitHub authorization server, and GitHub redirects back with a temporary code in a code parameter as well as the state GitGroup server provided in the previous step in a state parameter. If the states don’t match, the request was created by a third party and the process should be aborted. GitGroup exchange this code for an access token, then save this token in GitGroup database for later using and user get the GitHub access token.

When user request data with access token, user are directed to request data related to the GitHub protected data, and GitGroup server request to access GitHub user protected resource and handle GitHub resource, do business logic and re-package the resource. At last GitGroup return data to web server, web server visualize the data and display it to user.

Once user request data without access token or with wrong access token which means GitGroup cannot access GitHub user protected resource. After GitHub reject the access, GitGroup server return authorization fail information to user. User will request data successfully only if user request correct GitHub access token.

## Class Diagram

The system characterizes users to be either owners or collaborators of projects, or collaborators who are team member of project owner.