GitGroup - A Web Based Software Development Management System

Runbo ZHAO
Jingyu BAO
Xuying CAO
Chengxiang YIN

Advisor: Edmund S. Yu
Course: CSE 682 Software Engineering

October 28, 2018

Contents

Chapter 1

Introduction

1.1 Purpose

The document is an official statement of what the system developers should implement. It includes both the user requirements for a system and a detailed specification of the system requirements. The document is essential for outside contractor who developing the software system. It is also useful to write a short supporting document that defines the business and dependability requirement for the system. The document is for system customers, managers, system engineers, system test engineers, system maintenance engineers.

1.2 What are GitHub APPs?

GitHub Apps (GitHub Inc., n.d.) are first-class actors within GitHub. A GitHub App acts on its own behalf, taking actions via the API directly using its own identity, which means you don't need to maintain a bot or service account as a separate user.
GitHub Apps can be installed directly on organizations and user accounts and granted access to specific repositories. They come with built-in webhooks and narrow, specific permissions. When you set up for your GitHub App, you can select the repositories you want it to access.
Some key ideas when creating GitHub Apps

1. A GitHub App should take actions independent of a user (unless the app is using a user-to-server token).
2. Make sure the GitHub App integrates with specific repositories.
3. The GitHub App should connect to a personal account or an organization.
4. Don't expect the GitHub App to know and do everything a user can.
5. Don't use a GitHub App if you just need a "Login with GitHub" service. But a GitHub App can use a user identification flow to log users in and do other things.
6. Don't build a GitHub App if you only want to act as a GitHub user and do everything that user can do.

GitHub Marketplace contains tools and services that complement and improve workflow. Anyone can discover, browse, and install free and paid GitHub App and OAuth Apps in GitHub Marketplace.

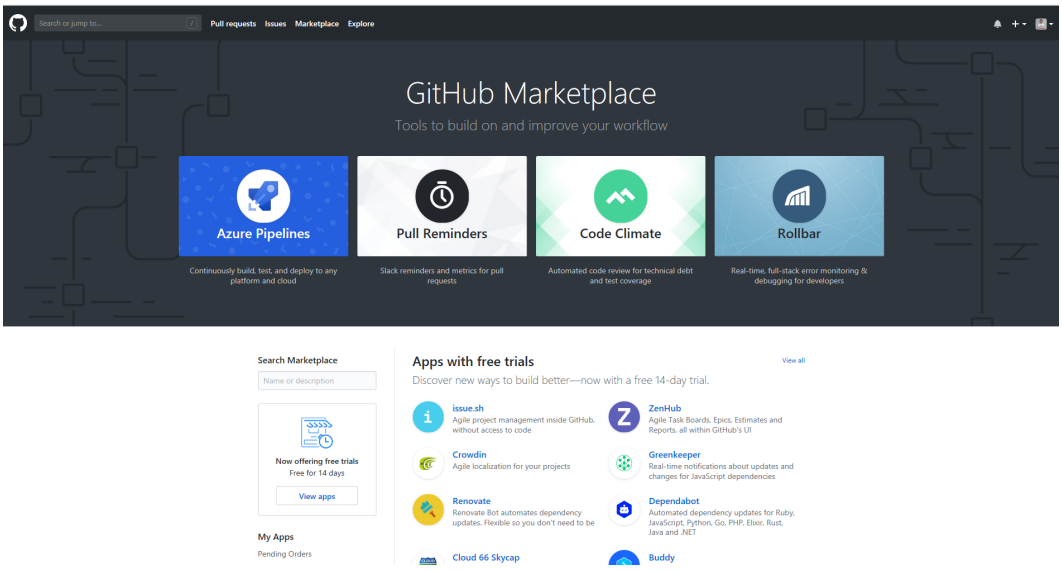

Figure 1.1: GitHub Marketplace

The following is the home page of GitGroup in GitHub.

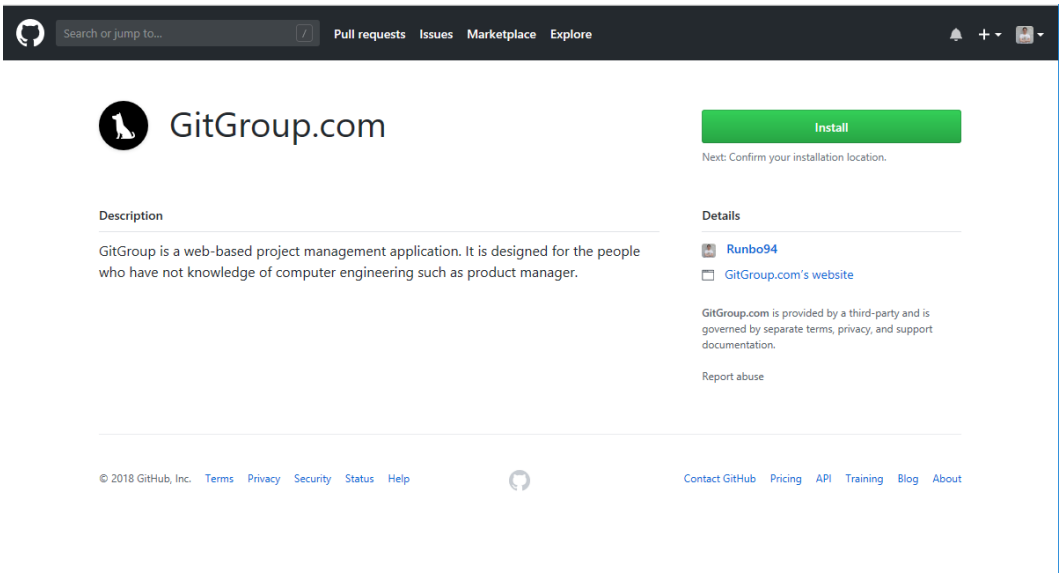

Figure 1.2: GitHub Marketplace

1.3 Product Scope

GitGroup is a web-based GitHub Application. It is a project management platform that allows everyone on your team to communicate with the developers. Therefore, GitGroup is not only designed for programmer but also for people who has less development experience such as project manager, project members and people who are related to the project. On GitGroup, it is easy to create an Agile workflow for your team with a kanban board. GitGroup's seamless integration with GitHub keeps all of your GitHub Issue data in sync across both platforms in real time. It is an easy access web-based application that served for product management team based on Github API.

GitGroup is built by several software development tools. From the perspective of programming language, we use both TypeScript and JavaScript. TypeScript is an open-source programming language.It is a strict syntactical superset of JavaScript, and adds optional static typing to the

language. And its strong Object-Oriented Programming features make it as a best developing language for building GitGroup. When the developing is done, TypeScript compiler can compile the source code to JavaScript program in order to let it run on the NodeJS. Javascript is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.
From the perspective of framework. ReactJS(Front end framework), Bulma(CSS framework), Frontawesome(Icon set and toolkit), ExpressJS(Back end framework) is chose. In the test process, our tool is Jest. It is a Javascript testing software. Jest is used by Facebook to test all JavaScript code including React applications. One of Jest's philosophies is to provide an integrated "zero-configuration" experience. This feature of Jest make the duration of developing shorter. Also we regard MongoDB as our host DBMS, which is one of the most popular database management tool. Other tools such as Git, Heroku, Visual Studio Code, Postman, are also included in developing process.

1.4 Definitions, Acronyms and Abbreviations

Git	A version-control system for tracking changes in computer files
GitHub	A web-based hosting service for version control using Git
REST	The architectural style defines constraints used for creating web services.
Owner	Who set up the project or mainly charge of the project
Collaborator	Who assists owner or project participants
Project	An element in GitGroup, including multiple repository
Repository	A folder which contains all project's files and stores each file's revision history
Issue	Information used to track ideas, enhancements, tasks, or bugs on GitHub
KanBan	A scheduling system for lean and just-in-time manufacturing
Cards	A message that signals depletion of product, parts, or inventory on KanBan

Table 1.1: Definitions, Acronyms and Abbreviations

1.5 Overview of The Document

The rest of the document will include overall description of Gitgroup in following perspectives. Product perspective, product functions, user characteristics, operating environment, design and implementation constraints, assumptions and dependencies. For interface requirement, we put efforts on user interfaces, hardware interfaces, software interfaces and communications interfaces. we will also include requirements like user requirements, system requirements, functional requirements, non-functional requirements and system models like use case diagram, activity diagram, sequence diagram, class diagram, system architecture, web design.

Chapter 2

Overall Description

This section of the SRS will include general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

2.1 Product Perspective

The background of the Gitgroup coming from project managers are often busy to managing multiple projects at the same time. Sometimes, they are not familiar with details with products especially in technology perspective. However, Git and Github have a high entrance standard which project manager hard to manipulate it. For example, when you want to create an issue to suggest a new idea or track a bug, you need to learn markdown syntax, emoji code. And it is even more complex when you do pull request and merge the project. Comparing with Git and Github, GitGroup is an easier access tool . People who has little development experience, such as product manager. They also can handle GitGroup easily. The only thing they need to do is sign up an account and access to the GitHub api. The user can create online group chat, schedual and resord conference. After finding project idea, our online app also provide the task management service. In each project, every user will have their own grade. Furthermore, the online web can have a recommendation system which help the product manager and programmer match each other. In conclusion, it is a online service based on the GitHub API, and improve the user experience.

2.2 Product Functions

Our product focus on easy access, clearly, convenience, efficiency. Fuctions are surrounded our goals. Mainly function lists in the following.

1. KanBan: The Kanban board which is similar to projects board in the Gitub. In the Kanban board, there are several columns represent different stage of the developement process. You can customize your own stage. Every column contains issue card imported from the Github, and you can easily drug them among the columns. If you drug it to the Done column. The issue in the Github will be automatically closed.

2. Chart - Project Analytics: GitGroup offers a variety of Agile charts and analytics to help you keep track of how your team is doing.

3. Online Meeting and Chatting: Online meeting group chat. Record the conference. Make conference scheduling form and alarming the coming meeting. GitGroup offers chat room for users to communication. There are two types of chat. One is global chat which is used

to communicate friends directly. Another is in-team chat which is used to communicate among the team members. for you.

2.3 User Characteristic

First of all, every use can create their own projects. If they are in their own projects, they are owners, verse and vice, they are collaborator. Therefore, one user can be both owner and collaborator according the project they are managing.
After entering a project, GitGroup will consist of 2 user flows, one for the owner of project, and the other for the collaborator. A high-level summary of the most important features offered by the system with respect to each of these users are as follows,

2.3.1 For Every User

1. Every user can create their own project.
2. Every user can use global chat to communicate all your friends directly.
3. Every user can get notification from system or other users.
4. Every user can view and edit their own personal profile.

2.3.2 For Collaborator

1. The collaborator can enter the KanBan board. In the KanBan board, they can add cards, drug and move current cards, edit the content of cards.
2. The collaborator can view the analytics which is different types of charts. to help keep track of how your team is doing.
3. The collaborator can use in-team chat to communicate with other developers in the team.

2.3.3 For Owner

1. The owner can do anything the collaborators can do.
2. The owner can invite other user to your project as collaborator.
3. The owner can remove any collaborators in your project.
4. The owner can add any your GitHub repositories to your project.
5. The owner can remove any GitHub repositories in your project.
6. The owner can add new KanBans in your KanBan board.
7. The owner can remove any KanBans in your KanBan board.
8. The owner can add new columns to any Kanbans and remove add columns in Kanban.

2.4 Operating Environment

The system will implement the model-view-controller software architecture pattern. The operating environment for the system consists of a browser that renders web pages, an external web server that handles user request-response transactions, an authentication and authorization server as well as a database for storing application data. Servers with reasonable load handling capacity will be sufficient for web hosting and data storage. Central administration is vested in the web/application server.

2.5 Design And Implementation Constraints

1. The key constraints of the system is the inherent reliability on an internet connection. Since the product is implemented using the client-server model, it is imperative to have an active internet connection.

2. Because GitGroup is deployed on Heroku. There are some constraints from Heroku including the application will sleep after 30 mins of inactivity, monthly pool of 1000 Free dyno hours. When a Free dyno is active, it draws from the pool; the application can run as long as there are dyno hours left in your monthly pool.

3. Because GitGroup will use GitHub API, the constraints come from the status of the GitHub API.

2.6 Assumptions And Dependencies

2.6.1 Assumptions

1. Since the system will be deployed using a cloud based hosting service, an assumption about its continuous, uninterrupted availability is made.

2. The system will depend on a trusted third party financial institution for handling monetary transactions. Hence, an assumption about total financial security is made.

2.6.2 Dependencies

1. The system will rely heavily on frameworks for user interface development, routing and data handling.

2. The system depends on the Heroku deployment platform.

3. The system depends on mLab Cloud Database.

4. The system will rely on some third part package, all the dependencies information is in the package.json file in the TypeScript and JavaScript projet.

Chapter 3

Interface Requirements

3.1 User Interfaces

Thu style of GitGroup UI must be clean and easy to use. The CSS framework GitGroup choose is Bulma. It is a beautiful, lightweight and stylish CSS framework. and its grid is fully built with flexbox. Achieving flexible layout with same-size columns is as simple as adding **.column** class to HTML element. Furthermore, Bulma has some GitHub design style component like panel. It can be easy reuse to build GitGroup project lists.
The icon set and toolkits chose for GitGroup is Font Awesome. Font Awesome is a font and icon toolkit based on CSS and LESS. It was made by Dave Gandy for use with Twitter Bootstrap, and later was incorporated into the BootstrapCDN. Font Awesome (Wikipedia contributors, 2018b) has a 20% market share among those websites which use third-party Font Scripts on their platform, ranking it second place after Google Fonts.

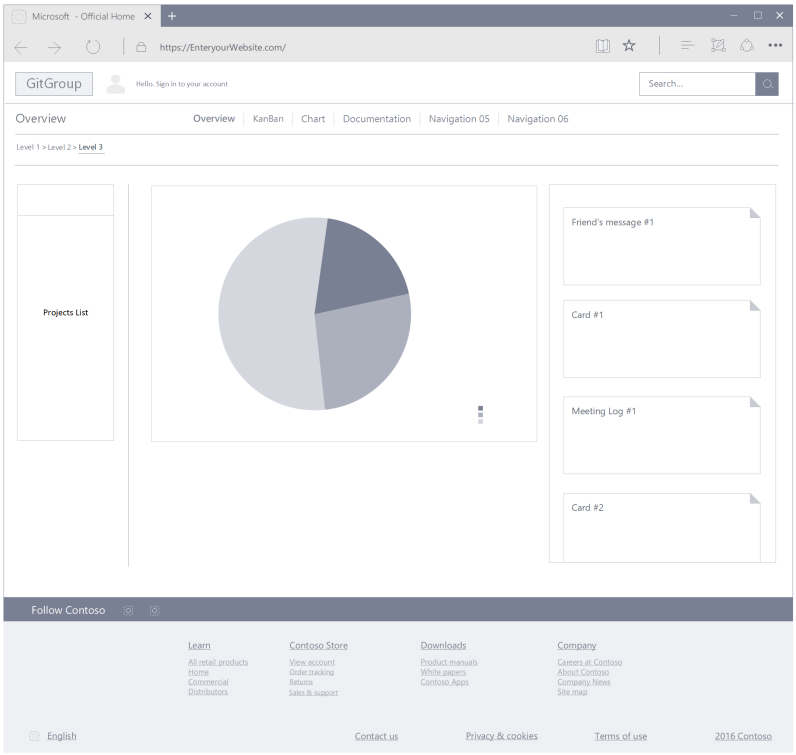

Figure 3.1: Overview User Interface (Details in Appendix A)

The overview is the home page of GitGroup. Because our design aim is ease and simple, all the important information should display on this main page. The main navigation bar shows the personal information. and the second navigation bar is used for navigate among the Overview,

Kanban, Chart and Documentation page. The next line is the tool bar, all the tool button should be here. Specially, the left side in this line is the bread crumb which is used to improve the navigation experience. The most important part is the workspace which is divided into 3 parts. The left side bar is the Project list where users can search, choose, edit, add and remove project. The combination of the project list and second navigation bar, which is called two-dimension navigation, is stronger, more accurate.
The middle part of workspace is user's current status, and it is shown as some diagrams and charts.
The right side bar is all the to-do things including cards which are close to deadline, message and meeting log.

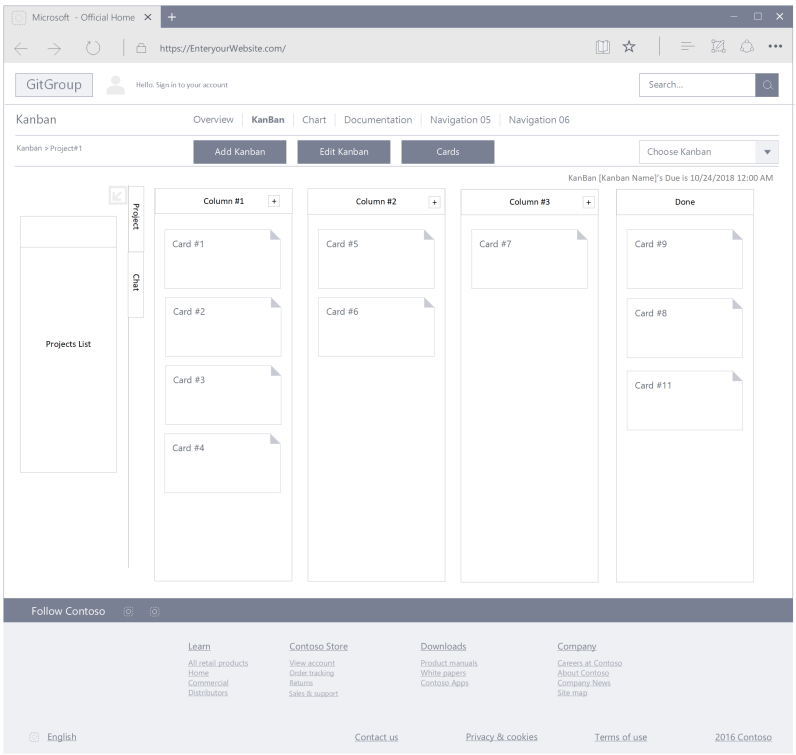

Figure 3.2: KanBan User Interface (Details in Appendix A)

The Kanban board is set of columns that allow you to track the progress of tasks as they move through your workflow. The GitGroup Kanban board default columns are Inbox, Backlog, Ready, In Progress, In Review and Done. Cards that are newly added to the Workspace (such as when an issue is made on GitHub) will go into the Inbox column by default. Cards that are moved into the Done column on GitGroup will be automatically closed and cards closed via GitHub will be automatically moved into the Done column. And the right top of Kanban board shows the due day of this KanBan.

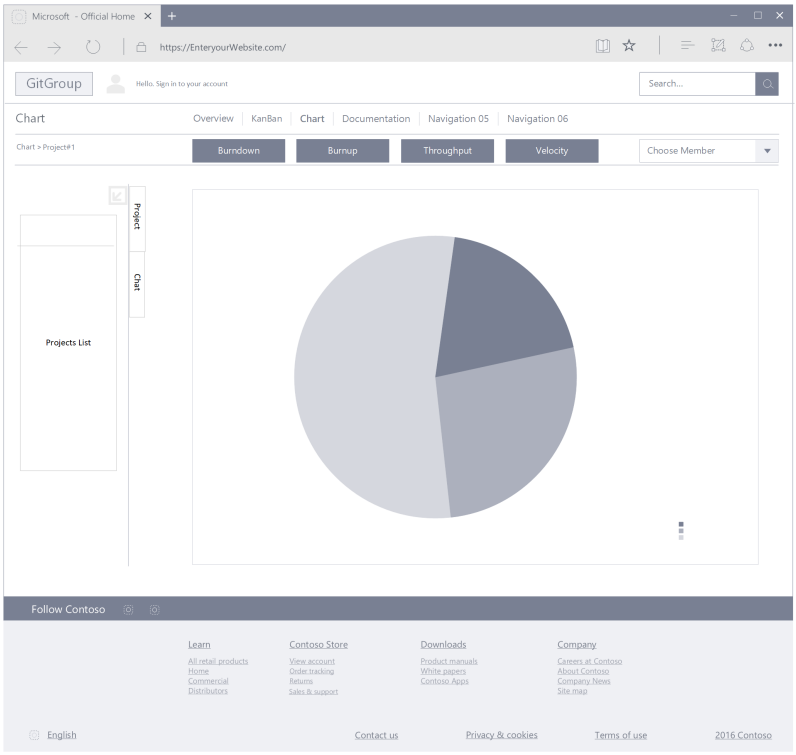

Figure 3.3: Chart User Interface (Details in Appendix A)

The Chart UI is easy and understandable. The main part shows different charts and you can change them via buttons on the tool bar.

3.2 Hardware Interfaces

Owing to the responsive nature of the product, it would require nothing more than a simple browser installed on a mobile phone, a tablet or a personal computer. A reasonable amount of RAM on the device would be enough to support the browser and the associated product. (Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

3.3 Software Interfaces

The product will use certain 3rd party libraries or api's to supplement overall system functionality. The google maps api is used for geocoding street addresses to geographical coordinates that can be used to place markers on the map. These markers provide visual representation for locating parking spaces for drivers. Api's for financial transactions will be used by the product for booking spots. Communication between the core-application and the api implementation normally happens through SOAP messages.
Software interacts with the database using JSON objects. MongoDB, the primary database vendor stores these objects in binary encoded form called BSON (or Binary JSON). The underlying operating system that runs the application server (web host) will be Linux. (Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

3.4 Communication Interfaces

The product will make use of a set of communications interface standards. They are summarized below. (Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

1. Client-Server communication needs to be encrypted in order to ensure secure data transfer. Thus, we require a HTTPS/SSL (or TLS) connections between the client and server always.

Chapter 4

Requirements

4.1 User Requirements

4.1.1 Functional User Requirement

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.

4.1.2 Non-Functional User Requirement

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.

4.2 System Requirements

4.2.1 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 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.

4.2.2 Back-end Interfaces(Functional Requirements)

BE-1.1 The server shall create and store project data in database.

BE-1.2 The server shall remove the project from database.

BE-2.1 The server shall store the collaborator data to the project document in database.

BE-2.2 The server shall get repository data from GitHub and then store the data to the project document in databse.

BE-3.1 The server shall change the state of the issue cards.

BE-3.2 The server shall create, store and remove the KanBan data to the project document in database.

BE-3.3 The server shall create, store and remove the KanBan Column data to the KanBan document in database.

BE-3.4 The server shall change the state of the cards in KanBan document.

BE-4.1 The server shall return analytics data of project when user request.

BE-5.1 The server shall provide chat service and provide service to create, store and remove the meeting log data.

4.2.3 Non-Functional System Requirement

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 When an fail occurs, the warning block shall display on the application window, and the page shall be navigated to the previous page or authorization page.

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.

Chapter 5

System Models

5.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system. There are two type user in GitGroup that are owner and collaborator. Both of them can use KanBan, view charts, and do chat. But the owner have higher privilege, they have the permission to invite or remove collaborator, delete the project or create and remove KanBan etc.

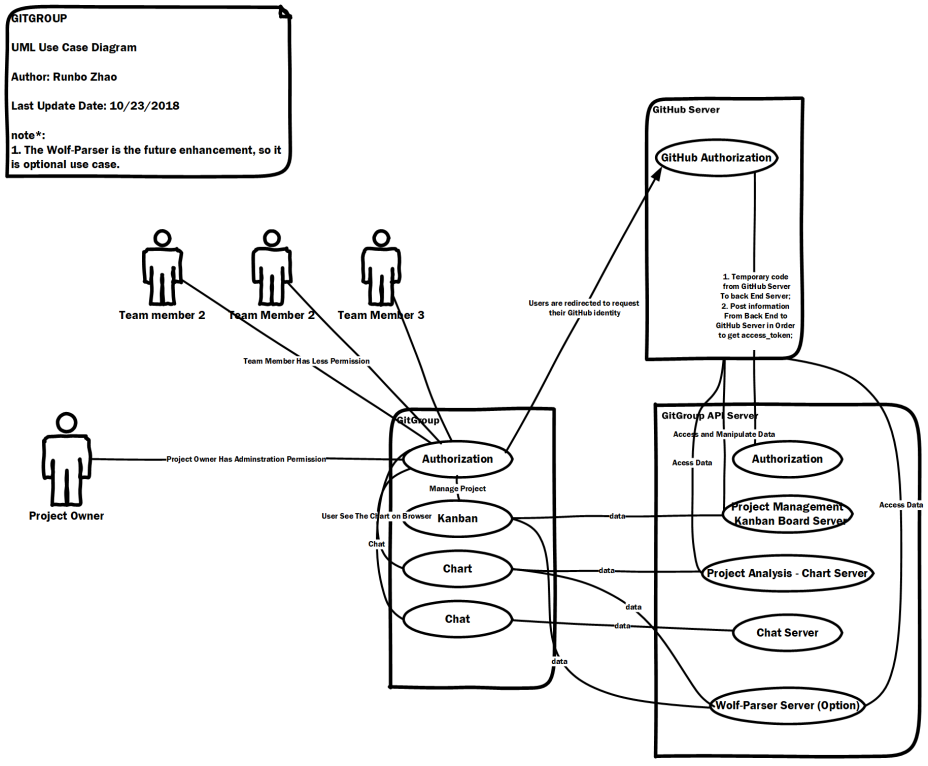

Figure 5.1: Use Case Diagram

5.1.1 KanBan

The Kanban board which is similar to projects board in the Gitub. In the Kanban board, there are several columns represent different stage of the developement process, such as TODO, Ready, In Progress, Done... You can customize your own stage. Both owner and collaborators can access KanBan, However the owner has higher privilege.
Every column contains issue card imported from the Github, and you can easily drug them among

the columns. If you drug it to the Done column. The issue in the Github will be automatically closed.(Zube team, n.d.)

5.1.2 Chart

GitGroup offers a variety of Agile charts and analytics to help you keep track of how your team is doing. Both owner and collaborators can view these charts.(Zube team, n.d.)

1. Burndown and Burnup Charts
   A Burnup chart tracks how much work has been completed during the Sprint. Burnup charts have three lines, Goal, Closed Points/Cards, and Ideal Closed Points/Cards. The Goal line allows you to easily see when new work is added to a sprint. Similar to the Burndown Ideal line, the Burnup ideal line marks how much work should have been closed in order to ensure all of the work in your Sprint is completed on time. The Closed line shows the total number of points or cards closed that have been closed.
   Both Burndown and Burnup charts can be filtered by source and can be set to exclude weekends in the Ideal line calculation. They can also display data using point totals or card totals.

2. Throughput Chart
   The Throughput chart shows you how much work your team is completing everyday. The chart is a stacked bar chart with each segment representing the work done on each of your sources. Throughput charts can help you spot changes in your team's productivity or workflow. Zube's Throughput chart has many filters that allow you dig into the details of what work is being completed everyday.
   The Throughput chart can be filtered by card type, source, assignee, epic, label, milestone and sprint. You can also adjust the date range displayed. Data can be presented as point totals or card totals.

3. Users Throughput Chart
   The Users Throughput chart tracks how much is being done by each team member weekly. Like the Throughput chart, the Users throughput chart allows you to track how efficiently your team is working and spot potential problems in your team members' workflows. The Users Throughput chart also offers a rich set of filters. The chart can be displayed as a stacked area chart or stacked bar chart.
   The Users Throughput chart can be filtered by source, assignee, label, milestone and sprint. You can also adjust the date range displayed. Data can be presented as point totals or card totals.

4. Velocity Chart
   If you are using Sprints, you can track how much work your team is getting done per sprint with the Velocity chart. The Velocity chart is a stacked bar chart showing how many cards or points were closed during each sprint in the selected date range. Each section of the bars represents one of your sources. The Velocity chart allows you to see potential problems in your team's efficiency as well as providing a way to predict how much work your team will be able to accomplish in future sprints.
   The Velocity chart can be filtered by source and label. You can also adjust the date range displayed. Data can be presented as point totals or card totals.

5.1.3 Chat

GitGroup offers chat room for users to communication. There are two types of chat. One is global chat which is used to communicate friends directly. Another is in-team chat which is used

to communicate among the team members.

5.2 State Diagram

A state diagram, sometimes known as a state machine diagram, is a type of behavioral diagram in the Unified Modeling Language (UML) that shows transitions between various objects.
The following state diagram shows the transition of the user type. Every use can create their own projects. If they are in their own projects, they are owners, verse and vice, they are collaborator. Therefore, one user can be both owner and collaborator according the project they are managing.

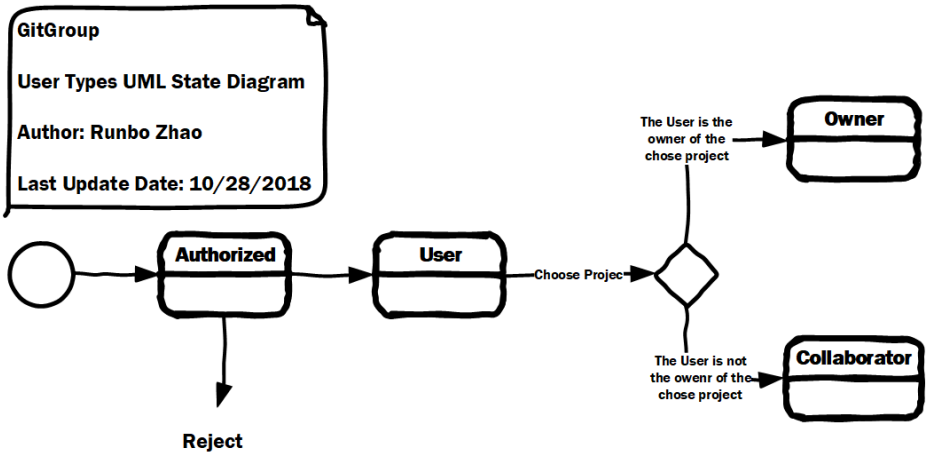

Figure 5.2: User Type State Diagram

GitGroup offers every project a KanBan board. You can create multiple KanBans for the project and every KanBan has its own due day, after the due day the KanBan will finished no matter how many cards are not finished. These unfinished cards' information is used to analyzed the project development status.
In one KanBan, some cards which represent some task flow through a set of step, such as developing, testing, done... . Each moving of a card will change its state in the KanBan.

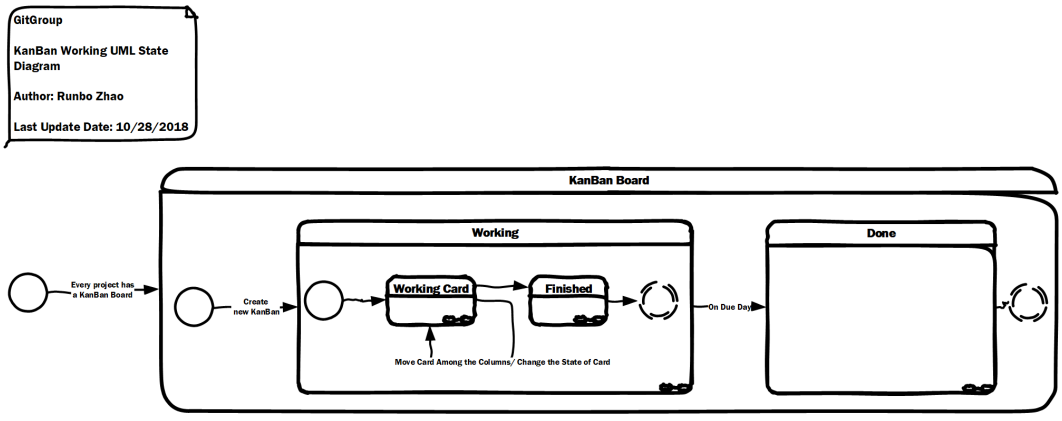

Figure 5.3: KanBan State Diagram

5.3 Activity Diagram

Activity diagram is used to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can

be described as an operation of the system.

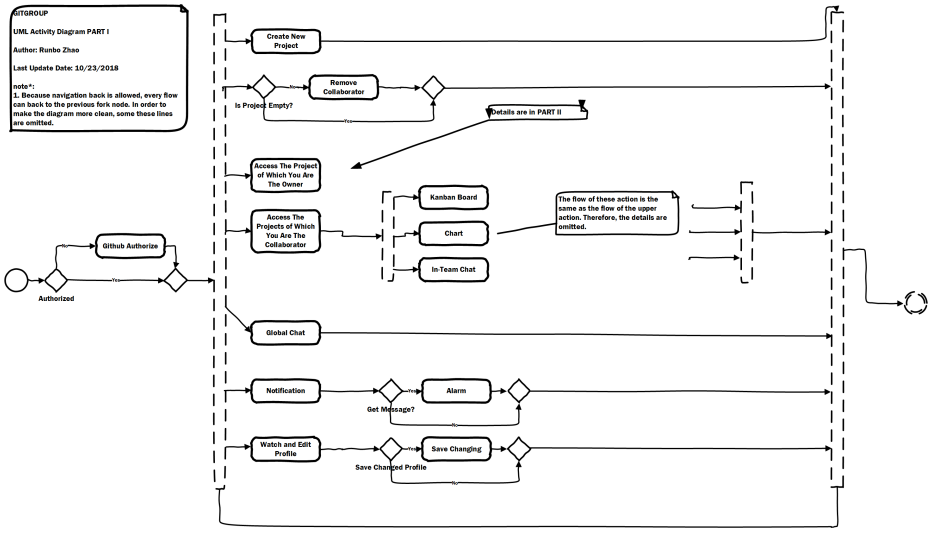

Figure 5.4: Activity Diagram PART I (Details in Appendix A)

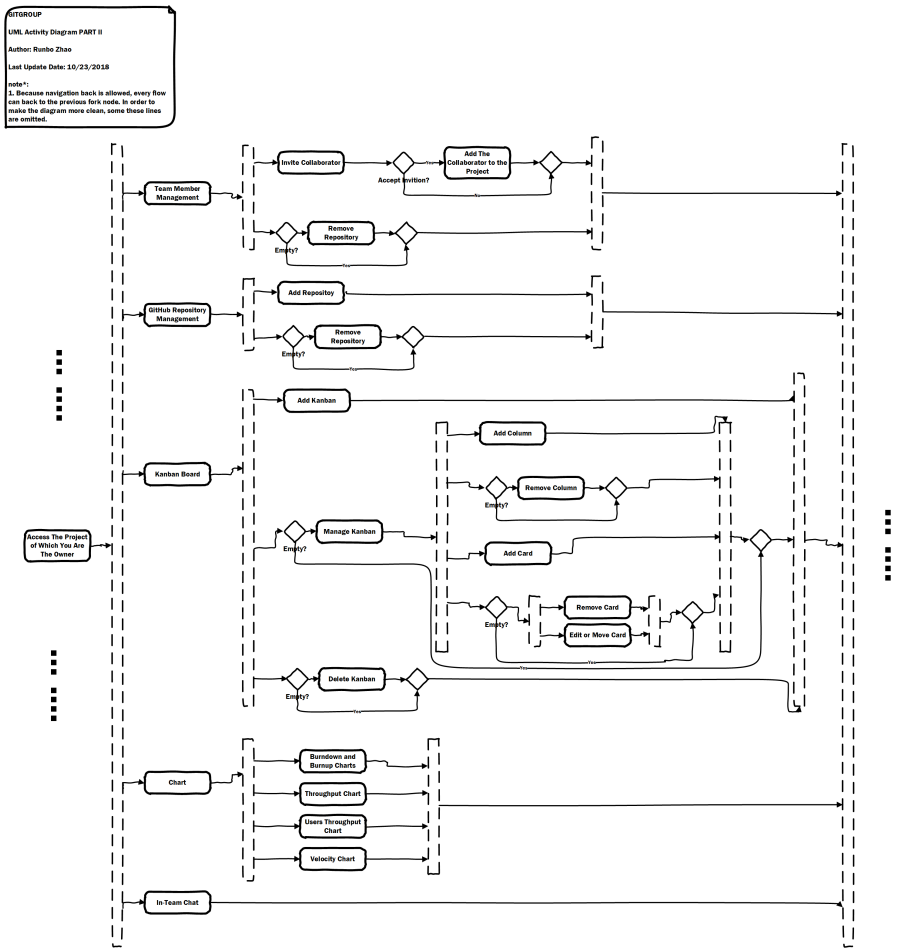

Figure 5.5: Activity Diagram PART II (Details in Appendix A)

According to the activity diagram, the workflow of GitGroup is:
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.

5.4 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The following sequence diagram is used to show how GitHub authorization works.

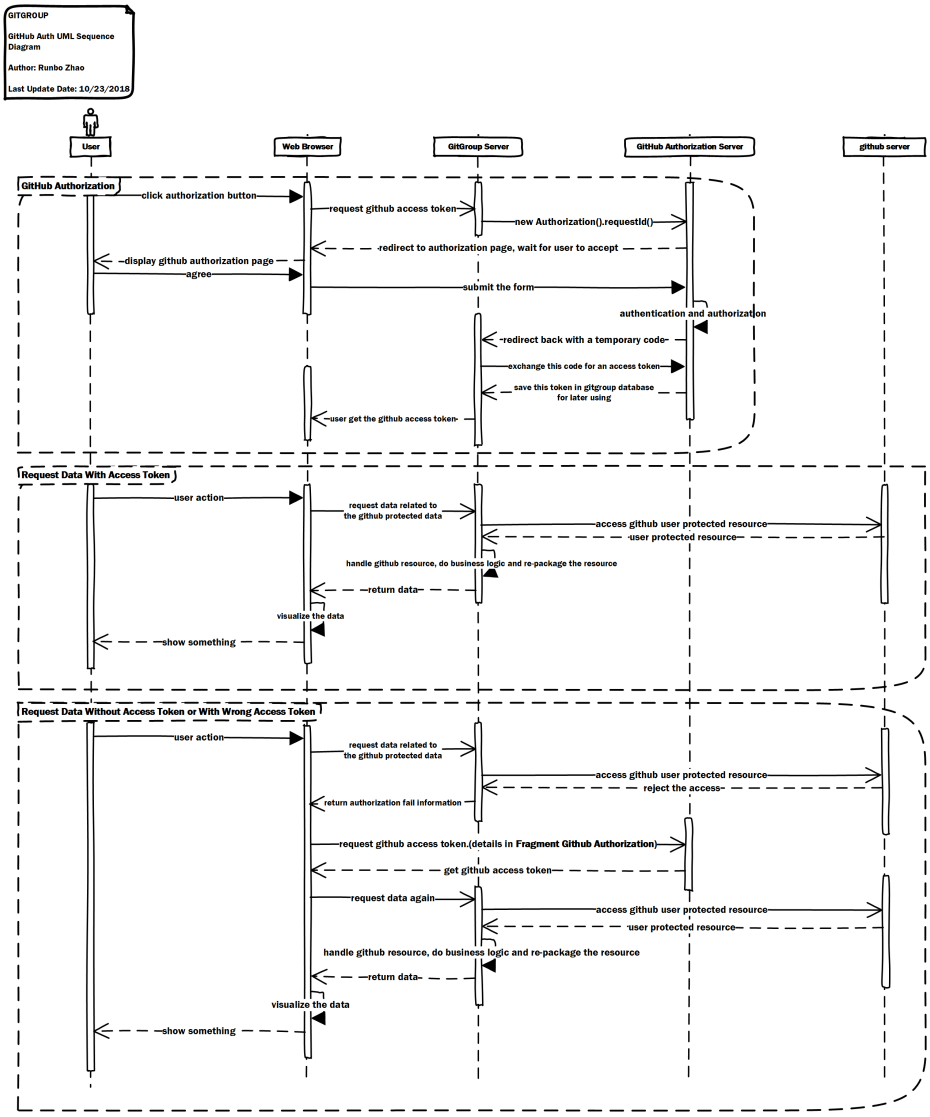

Figure 5.6: Sequence Diagram For Authorization (Details in Appendix A)

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.

The following sequence diagram shows the how the 3 sub-system including project management system, project analytics system and chat system interact with each other.

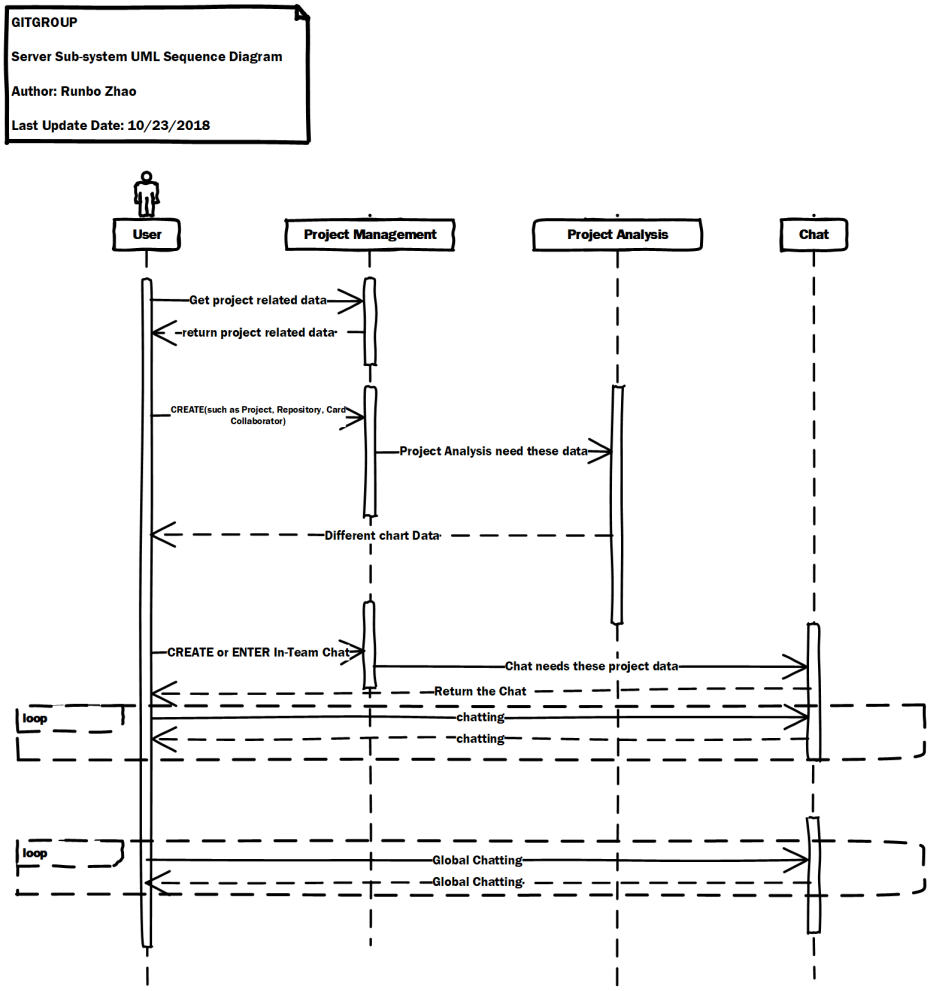

Figure 5.7: Sequence Diagram For Sub-Systems (Details in Appendix A)

Users can get and edit data via project management system. And the Project Analytics system must get data from project management system firstly and then return the analytics data to build chart. Also in-team chat need associate with project management system. But the global can work alone without any other systems.

5.5 Class Diagram

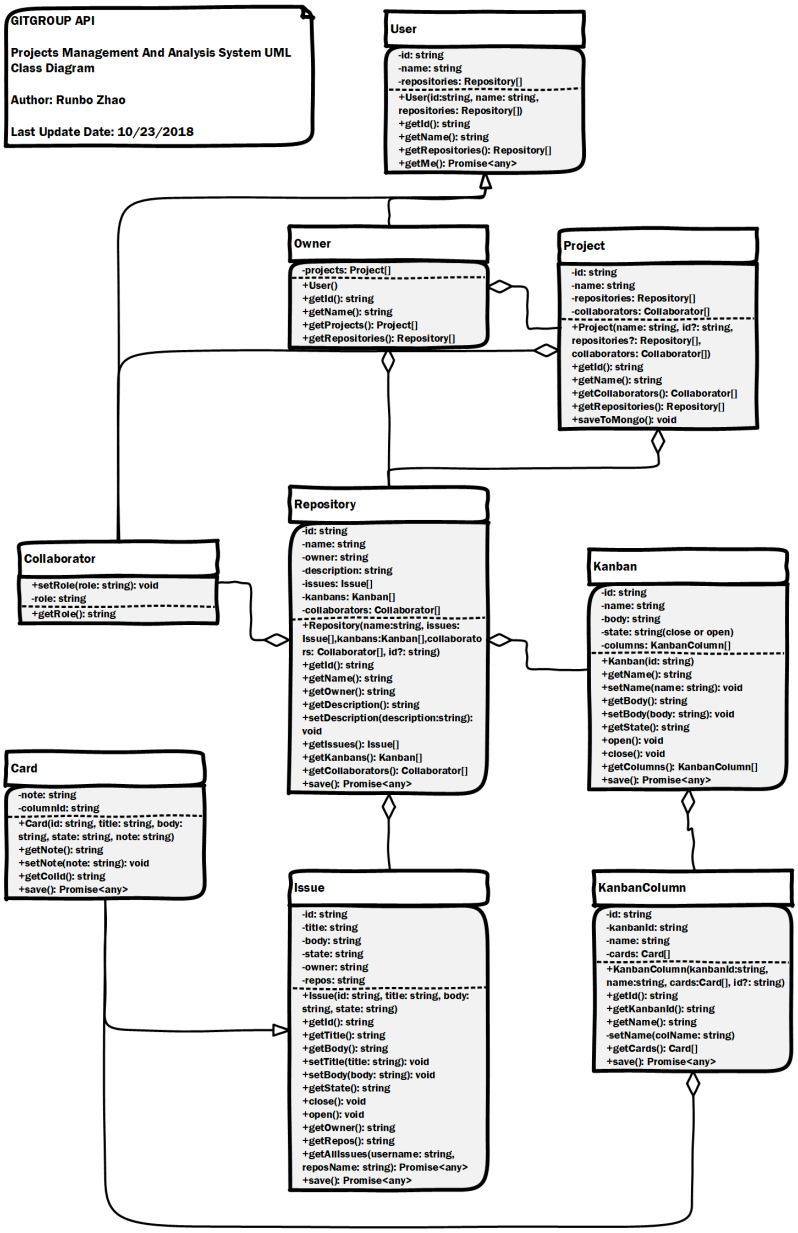

Figure 5.8: UML Class Diagram (Details in Appendix A)

The project manage management system is the most important part of GitGroup.
User class represents the user of GitGroup, its properties are id, name and repositories. Because there are two types of users, there are two class Owner and Collaborator extend from User.
Project class represents the project in GitGroup, the project is the combination of repositories. Its properties have id, name, repositories and collaborators.
Repository class represents the repository in GitHub. Its properties have id, name, owner, description, issues, kanbans and collaborators. And also the Issue class represents the issue in GitHub.
KanBan, KanBanColumn and Card class are key classes for KanBan board functionality. Card's properties are note and the column id it belongs to. KanBan's properties are id, name, body, state(close or open) and columns it has. KanBanColumn's properties are id, name, KanBan which the column belong to, Cards which the column has.

Chapter 6

System Architecture

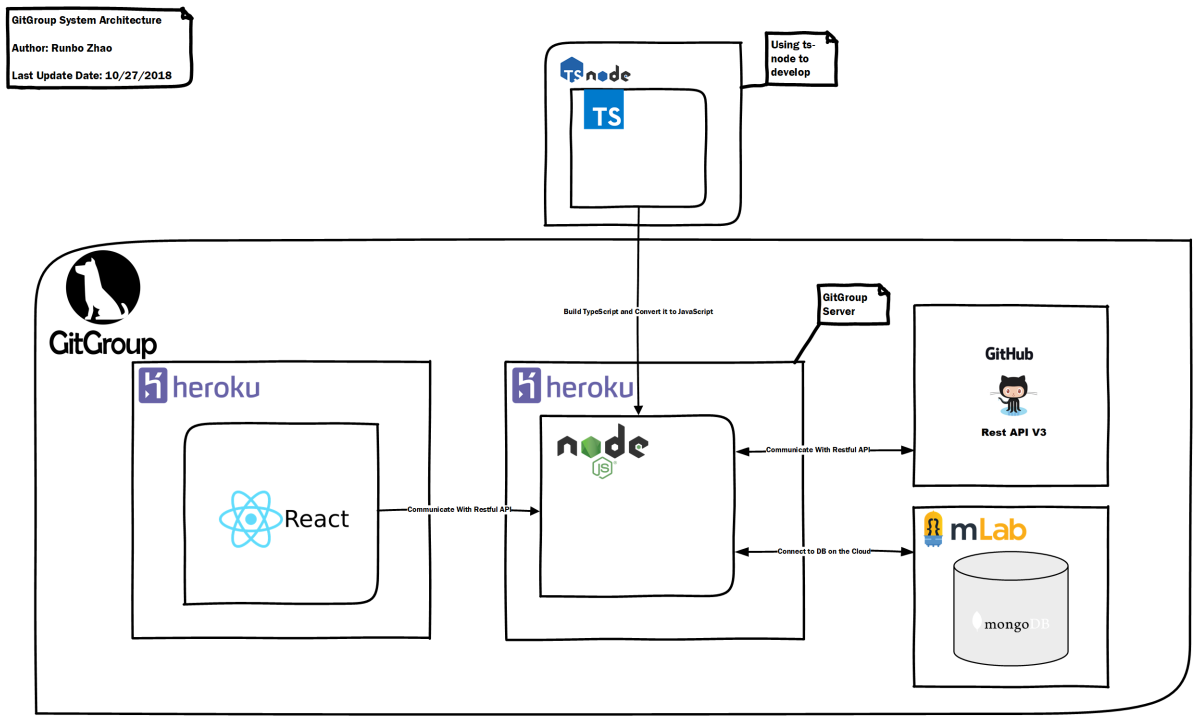

Figure 6.1: GitGroup System Architecture

6.1 TypeScript

TypeScript (Wikipedia contributors, 2018h) is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language.
TypeScript is used to develop GitGroup because its strong Object-Oriented-Programming features.
TypeScript is designed for development of large applications and transcompiles to JavaScript.As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs. TypeScript may be used to develop JavaScript applications for both client-side and server-side (Node.js) execution.

6.2 mLab

mLab (Wikipedia contributors, 2018d) is a fully managed cloud database service that hosts MongoDB databases. mLab runs on cloud providers Amazon, Google, and Microsoft Azure, and has partnered with platform-as-a-service providers.

6.3 MongoDB

MongoDB (Wikipedia contributors, 2018e) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata. MongoDB is developed by MongoDB Inc., and is published under a combination of the Server Side Public License and the Apache License.
MongoDB is used via mLab in this project. And Mongoose is chosen as the ORM(Object-relational mapping).

6.3.1 NoSQL language

(Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

1. It has the ability to horizontally scale by distributing the load over multiple servers.
2. We don't have to think of our data in terms of rows and columns of tables. The difference in the representation in the application on disk is sometimes called impedance mismatch. In MongoDB, instead, we can think of the persisted data just as we see it in our application code; that is, as objects or documents.
3. It helps a programmer avoid a translation layer, whereby one has to convert or map the object that the code deal with to relational tables. Such translation are called object relational mapping (ORM).

6.3.2 Document-Oriented

Every document in a collection has a unique identifier by which it can be accessed because the identifier is indexed automatically.

6.3.3 Schema-Less

Storing an object in a MongoDB database does not have to follow a prescribed schema. All documents in a collection need not have the same set of fields. MongoDB's flexible data model also means that our database schema can evolve with business requirements. For instance, schema changes that took days of weeks in The Weather Channel's MySQL databases could be made in just hours with MongoDB.

6.3.4 JavaScript Based

MongoDB's language is JavaScript. For relational database, there is a query language called SQL. For MongoDB, the query language is based on JSON: you create, search for, make changes, and delete documents by specifying the operation in a JSON object.

6.4 Express

Express, (Wikipedia contributors, 2018a) a web server built on Node.js, forms the middle tier or the web server. It defines routes, specifications of what to do when a HTTP request matching a certain pattern arrives. The matching specification is regular expression (regex) based and is very flexible. The what-to-do part is just a function that is given the parsed HTTP request. Express parses request URL, headers, and parameters for us. On the response side, it has, as expected, all of the functionality required by web applications including setting response codes, setting cookies, sending custom headers, etc.

6.5 React

React (Wikipedia contributors, 2018g) is a front-end technology from Facebook. It's not a full-fledged MVC framework, but it's a JavaScript library for building user interfaces, so in some sense it's the view part of MVC. We use React to render a view because it tries to solve the problem from the View layer by creating abstract representations of views. It breaks down parts of the view in the Components. These components encompass both the logic to handle the display of view and the view itself. It can contain data that it uses to render the state of the app.

6.5.1 Why use ReactJs?

(Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

1. DOM manipulation is the heart of the modern, interactive web. Unfortunately, it is also a lot slower than most JavaScript operations.

2. This slowness is made worse by the fact that most JavaScript frameworks update the DOM much more than they have to.

3. React views are declarative. We use it so that we don't have to worry about managing the effect of changes in the view's state or the data, meaning we do not worry about transitions or mutations in the DOM caused by changes to the view's state.

6.5.2 How does it works?

(Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj, 2018)

1. React is founded on the idea that DOM manipulation is an expensive operation and should be minimized. It also recognizes that optimizing DOM manipulation by hand will result in a lot of boilerplate code, which is error-prone, boring, and repetitive. React solves this by giving the developer a virtual DOM to render to instead of the actual DOM. It finds difference between the real DOM and virtual DOM and conducts the minimum number of DOM operations required to achieve the new state.

2. When the data changes, React conceptually hits the refresh button and knows to only update the changed parts.

6.6 NodeJs

NodeJs, (Wikipedia contributors, 2018f) a server-side JavaScript runtime environment, has an asynchronous eventdriven, non-blocking input/output (I/O) model, as opposed to using threads to achieve multitasking. It relies heavily on callbacks and promises to let you know that a pending task is completed. Node.js achieves multitasking using an event-loop. This is nothing but a queue of events that need to be processed and callbacks to be run on those events. An event-based approach also makes Node.js applications fast and lets the programmer be blissfully oblivious of the semaphores and locks that are utilized to synchronize multithreaded events.

6.7 Heroku

Heroku (Wikipedia contributors, 2018c) is a cloud platform as a service (PaaS) supporting several programming languages. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go.
In this project, Heroku is the platform where GitGroup is deployed to.

Appendix A

Figures

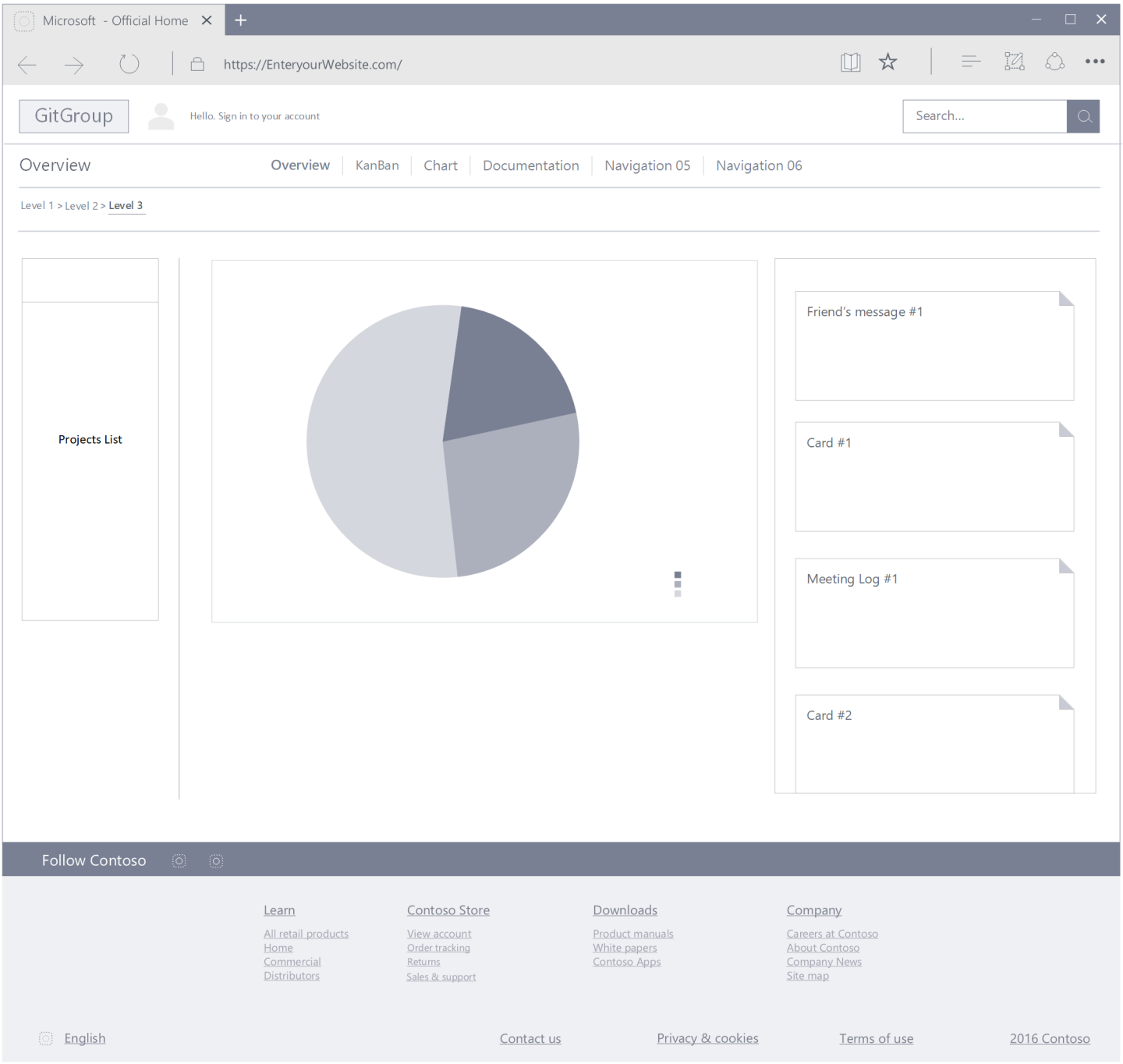

Figure A.1: **DETAIL** Overview User Interface

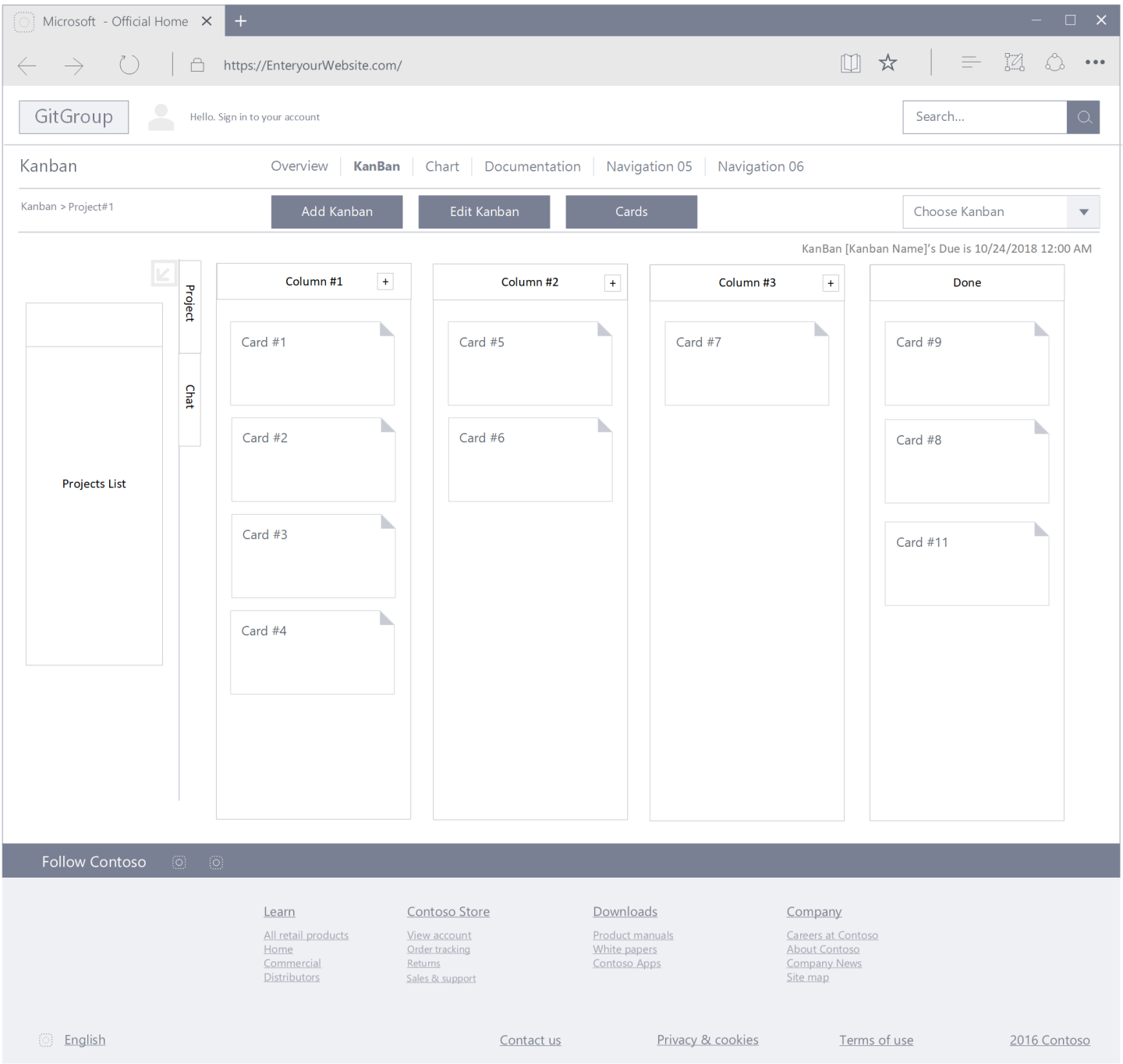

Figure A.2: **DETAIL** KanBan User Interface

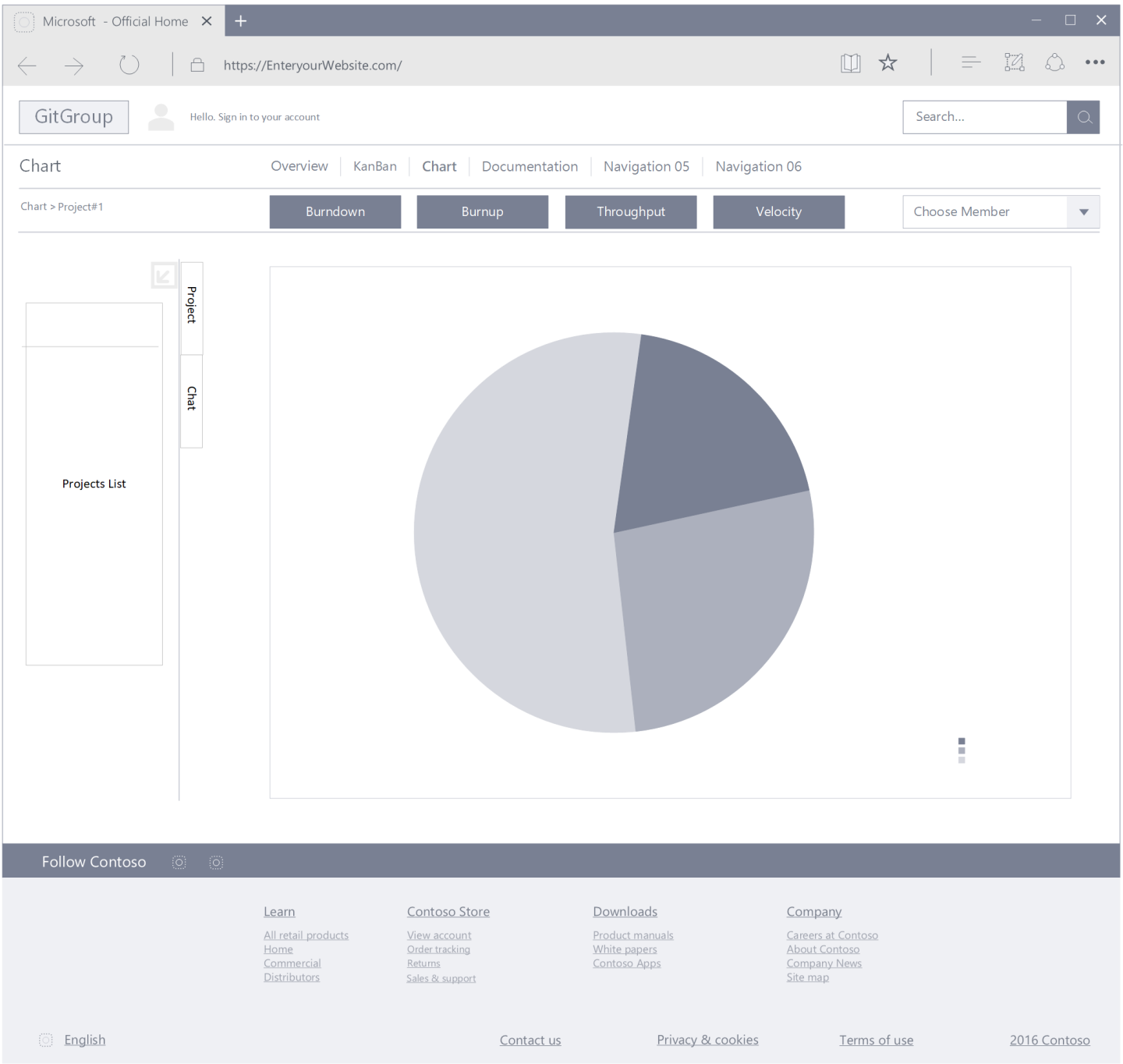

Figure A.3: **DETAIL** Chart User Interface

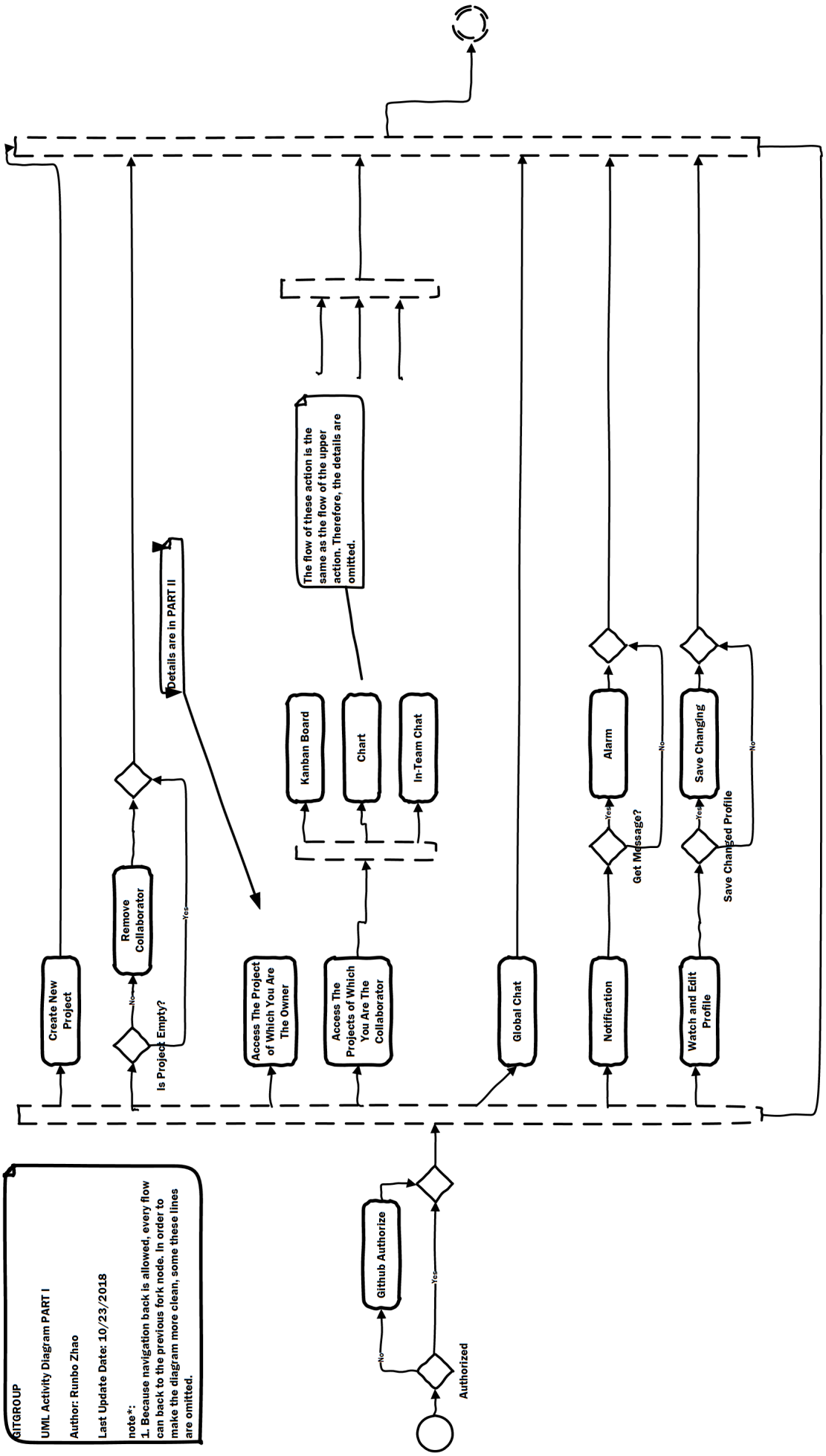

Figure A.4: **DETAIL** Activity Diagram PART I

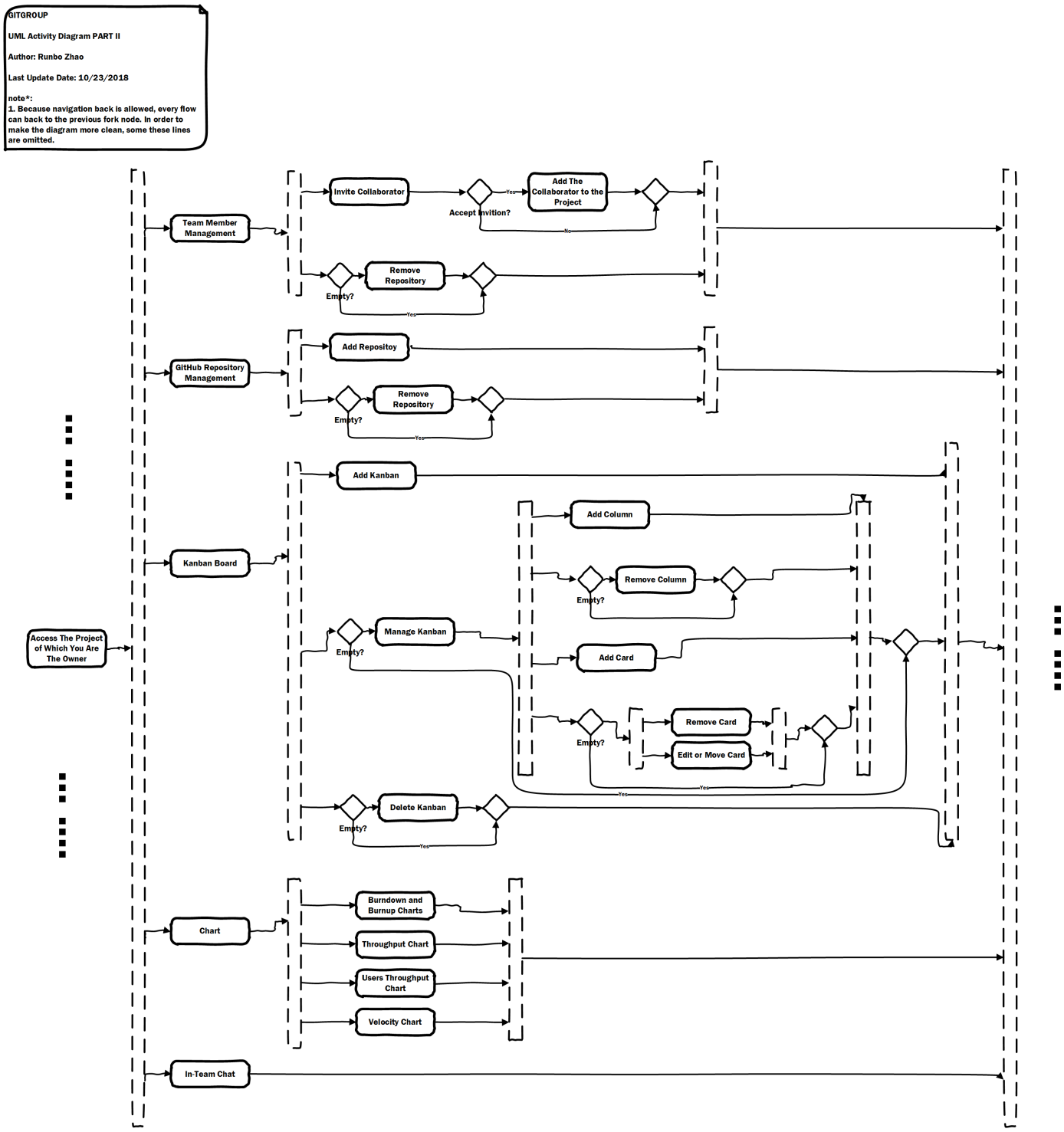

Figure A.5: **DETAIL** Activity Diagram PART II

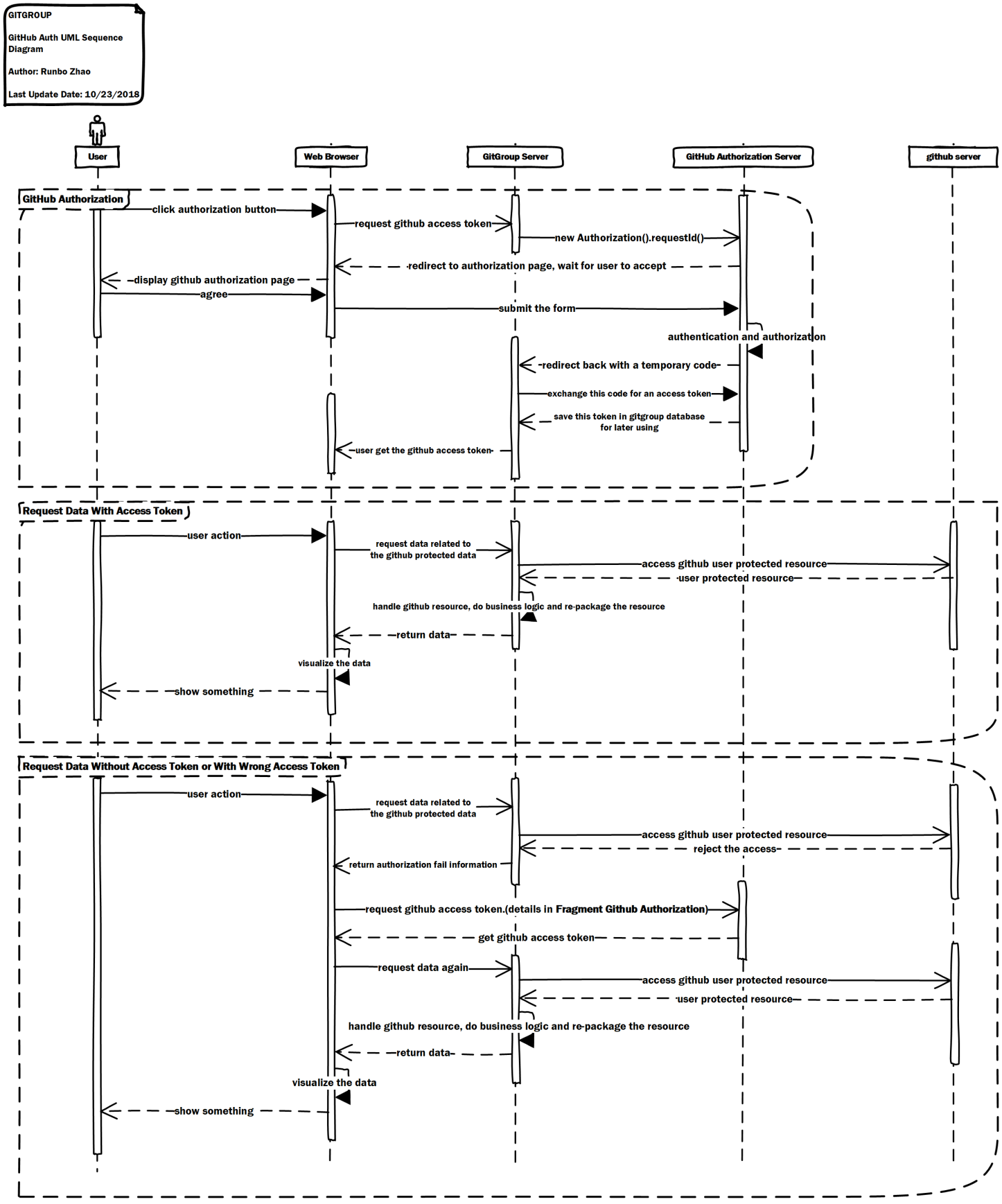

Figure A.6: **DETAIL** Sequence Diagram For Authorization

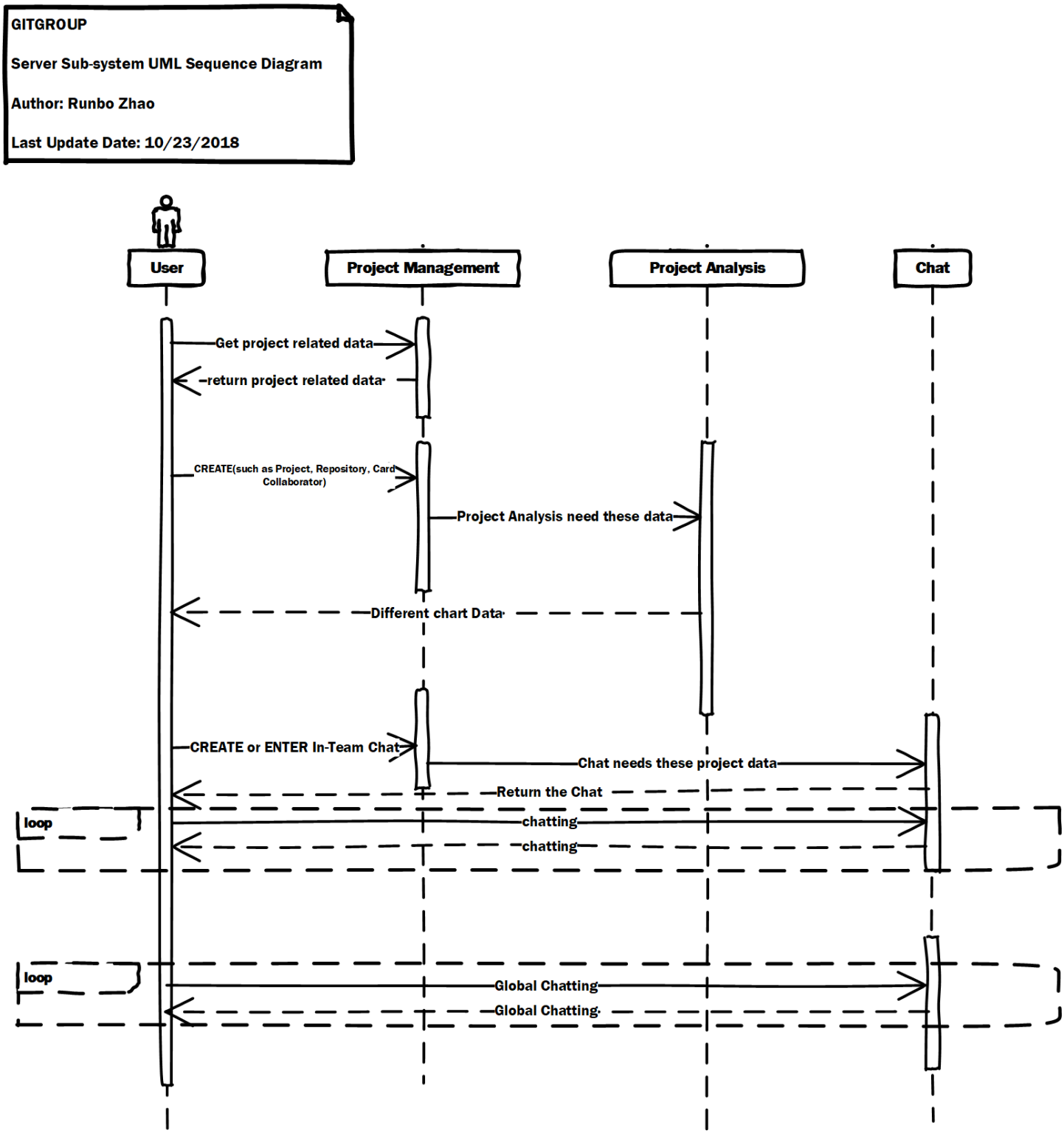

Figure A.7: **DETAIL** Sequence Diagram For Sub-Systems

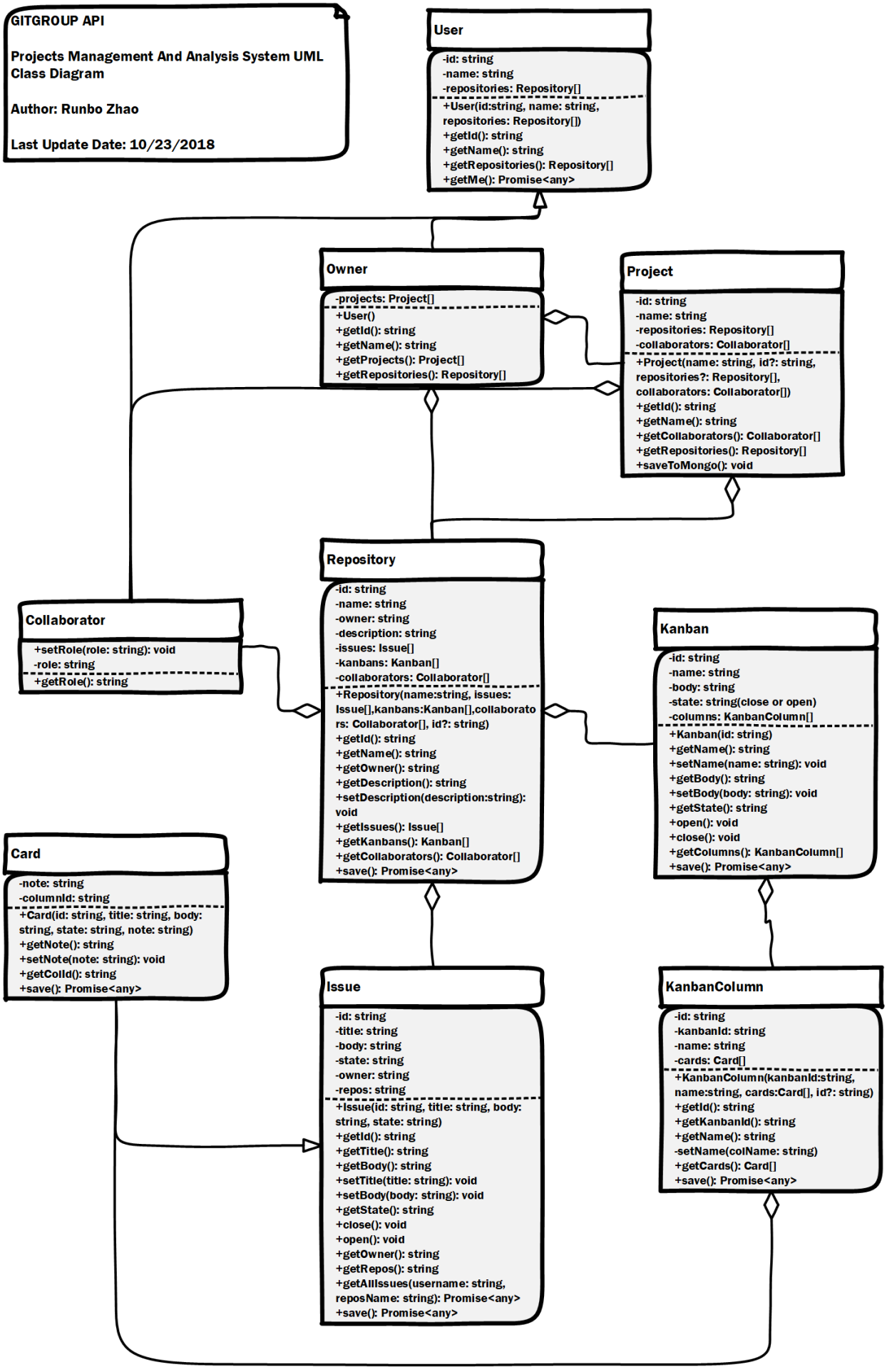

Figure A.8: **DETAIL** UML Class Diagram

List of Figures

List of Tables

References

Daniel Jonathan Kemp (Team leader), Jinwoo Song, Pratheek Basavana Gowda, Rajesh Ramesh, Rahul Vijaydev, Theerut Foongkiatcharoen, Vibhu A Bharadwaj. (2018). *SRS: Parking Marketplace* (Tech. Rep.).

GitHub Inc. (n.d.). *Github api v3 — github developer guide.*

Wikipedia contributors. (2018a). *Express.js — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=Express.js&oldid=861000326` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018b). *Font awesome — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=Font_Awesome&oldid=862582036` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018c). *Heroku — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=Heroku&oldid=855032748` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018d). *Mlab — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=MLab&oldid=863362348` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018e). *Mongodb — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=MongoDB&oldid=864580817` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018f). *Node.js — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=Node.js&oldid=865979769` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018g). *React (javascript library) — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=React_(JavaScript_library)&oldid=865980729` ([Online; accessed 28-October-2018])

Wikipedia contributors. (2018h). *Typescript — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/w/index.php?title=TypeScript&oldid=864413390` ([Online; accessed 28-October-2018])

Zube team. (n.d.). *Zube documentation.*