Name	Graham Edmond Bartley
Project Title	CoderDojo Zen Projects
Document Type	Functional Specification
Student Number	14541017
Completion Date	21/11/2017

Table of Contents

1. Introduction

1.1 Overview

CoderDojo is a global movement of volunteer-led programming clubs for youths aged 7 to 17 years old. Zen is the CoderDojo open-source community platform which was launched in 2015 by the CoderDojo Foundation (CDF) as a place for the community to communicate, manage their Dojos and events and share their experiences. One of the core features that the CDF want for Zen in 2018 is the ability for youths to be able to share coding projects they have worked on through the platform. This is the focus of my project.

1.2 Business Context

The CoderDojo Foundation exist to support and scale the global CoderDojo movement. They are a team of ten people based in Dublin. One of the main ways they support the movement is through developing new features for Zen. I worked as an intern with the CDF during the Summer of 2017 and towards the end of my time there I was asked if I would like to work on this project for them.

From their point of view, the CDF are always looking to improve Zen and onboard more new people onto the platform. They believe that this project will be a massive improvement to Zen which will see many new users joining the platform and current users having new functionality to enjoy. When I complete this project, it will be deployed by the CDF to production for Zen to be used by the global community.

1.3 Glossary

- **Microservice architecture:** A computer software architecture ideology involving small, independent, modular components which connect to make up a larger system.
- **Version Control System (VCS):** A VCS allows management of changes to a set of information over time such as source code or electronic documents.
- **Frontend:** Code that comprises the part of a system which the users interact with directly, generally called the user interface.
- **Backend:** Code that comprises the part of a system which the users do not directly interact with, but may interact with indirectly through the frontend. Usually contains more complicated logic and functionality.
- **Zen:** The open-source CoderDojo community platform.
- **Dojo:** A programming club founded and run by CoderDojo volunteers. There are thousands of these all over the world.
- **Mentor:** A CoderDojo voluntary tutor who helps youths in a Dojo to learn.
- **Champion:** The founder of a Dojo. Each Dojo must have a champion.

2. General Description

2.1 Product / System Functions

I will create an easily extendable browser runtime environment in JavaScript which will support running different programming languages and integrate this into the existing Zen architecture. The languages I will be supporting runtime for are Python and JavaScript with Scratch as a stretch goal since these are the most popular languages used by youths in Dojos.

I will also be extending the functionality of Zen to allow youths to be able to upload their project code in the first place since this is not something that can currently be done on Zen. When speaking with the Foundation about their requirements for this project, they said that they want uploaded projects to have versioning using an existing version control system. I am going to use GitHub and their version 4 GraphQL API for this purpose.

In addition to this, I will be tracking statistics relating to these projects and making them available to the CDF through an administrator panel. I will also be handling project management by different user types and organizing of projects based on their content, level, language etc. in order to facilitate searching for projects and linking of projects to certain CoderDojo learning resources.

2.2 User Characteristics and Objectives

My project is mainly targeted at youths between the ages of 13 and 17 years old inclusive. These will be the users that will be sharing their projects through the platform and so they will see the most benefit from my project. Youths under 13 cannot have their own profiles on Zen, they can only exist through a parent profile. I am expecting this user group to be familiar with computing devices and interacting with websites since people of this age group generally use modern technology in their everyday lives. It can also be said that this particular group that I will be targeting are likely to be even more familiar with technology than their peers since they are involved in CoderDojo and likely have an interest in programming.

However, while youths are the main target audience for my project I also have to consider the needs of the following user groups who will interact with my project:

- Parents
- Mentors
- Champions
- CDF employees

Expected expertise will vary between the user groups since the age range and background of all of these users is so broad. For this reason, I will focus mainly on the target audience of youths when designing my system since they will be the primary users of it. That being said, I can assume that parents will generally be the least technically-able of the user groups since they will be the least involved in CoderDojo and also some of the oldest users who are less likely to have grown up with computing devices. I can also assume a certain expertise of the CDF employees since I know their skill levels from working with them. I will take these things into account when designing my system also.

The following are requirements I have identified for each of my user groups and feasible solutions to implementing these requirements:

All users (including general public)	
Requirement	Solution
Search projects	Each project will have automatically assigned tags based on its type and more tags may be added by its owner. These tags will facilitate searching for projects based on keywords. Projects will also be filterable by Dojo, country and programming language.
Run projects	Runtimes for supported languages will be implemented so that projects can run in a browser environment.
Find learning resources related to projects	Project owners will have the option to link their project to specific learning resources on the CoderDojo resource platform in order to promote use of these resources. If no link is provided, projects will automatically link to general learning resources based on their type.

Youths	
Requirement	Solution
Upload their projects	Project code will be uploaded to Zen from the youth's computer and automatically uploaded to GitHub to be stored.
Manage their projects	Projects may be edited by their owners. Project information, code and resources can all be updated and versions can be managed using GitHub versioning which will be abstracted away from the user.
Earn digital badges for their work	Mentors and champions will have the ability to link badges they award to youths to a particular project the youth has worked on and this will be shown on the page for that project.

Parents	
Requirement	Solution
Track progress of their children	Parents will be able to visit a specific page to view projects created by their children so they can see their progress.

Mentors & Champions	
Requirement	Solution
Award badges to youths for their projects	When awarding badges, mentors and champions will be able to link them to specific projects created by the youths.

CDF Employees	
Requirement	Solution
View statistics related to projects	Statistics relating to project usage, impact and other effects will be tracked and presented to CDF employees only through an administrator's panel.
Manage projects	The administrator's panel will allow CDF employees to manage all projects on the platform in case they need to remove or modify them.

A "wish list" of stretch goals for the project is as follows:

- **Scratch 3 & Java support:** Scratch and Java are popular languages among youths in CoderDojo and so I would love to add support for their runtimes. However, they are out of scope for my project due to time constraints and the issue of the unknown release date of Scratch 3 which would be too risky to depend upon.
- **Code analysis and feedback:** Performing analysis on project code in order to give youths feedback on syntax and style among other things to improve their projects would be a great feature to have. However, again due to time constraints it would not be possible for me to implement this within the scope of this project since I will be implementing runtimes.

2.3 Operational Scenarios

The following are use cases for my project:

USE CASE 1	Create new project
Goal in Context	To make a new project on Zen
Scope & Level	Primary task
Preconditions	A youth has logged in to Zen and has code they would like to upload available on their computer in a .zip format
Success End Condition	A new project has been created on Zen
Failed End Condition	The youth was unsuccessful in creating a new project
Primary, Secondary Actors	Primary: Youth Secondary: Zen Frontend
Trigger	Youth clicks the “Add Project” button on their profile page
DESCRIPTION	
Step	**Action**
1.	Youth clicks the “Add Project” button on their profile page
2.	Zen Frontend presents a form to the Youth on a new page
3.	Youth fills in the required information in the presented form
4.	Youth clicks the “Upload Code & Resources” button
5.	Zen Frontend prompts the Youth to select a file on their system
6.	Youth selects a file on their computer with a .zip extension and clicks the “OK” button
7.	Zen Frontend displays the progress of the file upload to the Youth until the upload is complete
8.	Youth clicks the “Save Project” button
9.	Zen Frontend redirects the Youth to the new page for their new project
EXTENSIONS	
Step	**Branching action**
7a.	File upload fails, timeout will occur and the Zen Frontend will prompt the Youth to retry the upload

VARIATIONS	
Step	**Branching action**
8a.	Youth clicks the “Cancel” button, Zen Frontend will redirect them to their profile page and project creation will be cancelled

USE CASE 2	Update project
Goal in Context	To update an existing project on Zen
Scope & Level	Primary task
Preconditions	A youth is logged in and has a project they own on Zen
Success End Condition	The project has been updated
Failed End Condition	The project remains as it is
Primary, Secondary Actors	Primary: Youth Secondary: Zen Frontend
Trigger	Youth clicks the “Edit Project” button on the project page for the specific project they want to edit
DESCRIPTION	
Step	**Action**
1.	Youth clicks the “Edit Project” button on the project page for the specific project they want to edit
2.	Zen Frontend redirects Youth to the edit profile page for their project
3.	Youth enters updated information in the presented form
4.	Youth clicks the “Save Project” button
5.	Zen Frontend redirects Youth to the project page for their project which now contains the updated information
EXTENSIONS	
Step	**Branching action**
VARIATIONS	
Step	**Branching action**

4a.	Youth clicks the “Cancel” button, Zen Frontend will redirect them to their profile page and project will not be updated

USE CASE 3	Delete project
Goal in Context	To delete an existing project on Zen
Scope & Level	Primary task
Preconditions	A youth is logged in and has a project they own on Zen
Success End Condition	The project is deleted from Zen
Failed End Condition	The project is still on Zen
Primary, Secondary Actors	Primary: Youth Secondary: Zen Frontend
Trigger	Youth clicks the “Edit Project” button on the project page for the specific project they want to delete
DESCRIPTION	
Step	**Action**
1.	Youth clicks the “Edit Project” button on the project page for the specific project they want to delete
2.	Zen Frontend redirects Youth to the edit project page for their project
3.	Youth clicks the “Delete Project” button
4.	Zen Frontend prompts the Youth to say “Are you sure you want to delete this project? It cannot be recovered once deleted.”
5.	Youth clicks “OK” button on the presented prompt
6.	Zen Frontend redirects the Youth to a confirmation page
EXTENSIONS	
Step	**Branching action**
VARIATIONS	
Step	**Branching action**

5a.	Youth clicks the “Cancel” button and the prompt is closed. The project is not deleted and the Youth remains on the edit project page

USE CASE 4	Run project
Goal in Context	To interact with a project on Zen through running it
Scope & Level	Primary task
Preconditions	A user has navigated to a project page on Zen
Success End Condition	The user runs and interacts with the project
Failed End Condition	The user does not run the project
Primary, Secondary Actors	Primary: User Secondary: Zen Frontend, Zen Backend
Trigger	User clicks the “Run the Project” button on the project page
DESCRIPTION	
Step	**Action**
1.	User clicks the “Run the Project” button on the project page
2.	Zen Backend pulls the code for this project
3.	Zen Backend runs the project code
4.	Zen Frontend presents the running project to the User
5.	User interacts with the project through the presented display
EXTENSIONS	
Step	**Branching action**
2a.	The code cannot be retrieved. Zen Frontend displays an error message to the User “Project code cannot be found.”
2b.	Project type is HTML/CSS/JavaScript so instead of pulling the code to run it, Zen Frontend opens the project in a new tab of the User’s browser to interact with as a website
VARIATIONS	
Step	**Branching action**

USE CASE 5	View children's projects
Goal in Context	To view projects created by one's children
Scope & Level	Secondary task
Preconditions	A parent is logged in to Zen
Success End Condition	The parent can view their child's projects
Failed End Condition	The parent cannot view their child's projects
Primary, Secondary Actors	Primary: Parent Secondary: Zen Frontend
Trigger	Parent clicks the "My Projects" button in the profile dropdown
DESCRIPTION	
Step	Action
1.	Parent clicks the "My Projects" button in the profile dropdown
2.	Zen Frontend redirects Parent to a page displaying the most recent projects their children have uploaded
EXTENSIONS	
Step	Branching action
2a.	Parent has no children. Zen frontend redirects Parent to a page displaying "You have no projects!"
VARIATIONS	
Step	Branching action

USE CASE 6	View project statistics
Goal in Context	To view statistics related to projects
Scope & Level	Primary task
Preconditions	A CoderDojo Foundation (CDF) administrator is logged in to Zen and project statistics exist on the platform

Success End Condition	Project statistics are shown to the CDF admin
Failed End Condition	Project statistics cannot be shown
Primary, Secondary Actors	Primary: CDF Admin Secondary: Zen Frontend
Trigger	CDF Admin clicks the “Admin Panel” button in the profile dropdown
DESCRIPTION	
Step	**Action**
1.	CDF Admin clicks the “Admin Panel” button in the profile dropdown
2.	Zen Frontend redirects the CDF Admin to the admin panel
3.	CDF Admin clicks the “Project Statistics” button
4.	Zen Frontend displays project statistics in the form of graphs and charts
EXTENSIONS	
Step	**Branching action**
VARIATIONS	
Step	**Branching action**
3a.	After clicking “Project Statistics”, the CDF Admin clicks “Display Tables”. Zen Frontend displays project statistics in table form

USE CASE 7	Search projects
Goal in Context	To find a project based on search criteria
Scope & Level	Primary task
Preconditions	A user has navigated to the Zen projects page and is looking for a project
Success End Condition	The user finds a relevant project
Failed End Condition	No relevant project is found

Primary, Secondary Actors	Primary: User Secondary: Zen Frontend
Trigger	User clicks the “Search for project” button
DESCRIPTION	
Step	**Action**
1.	User clicks the “Search for project” button
2.	Zen Frontend displays a search bar for the User
3.	User enters a term in the search bar
4.	User clicks the “Search” button
5.	Zen Frontend displays projects matching the search criteria to the User
EXTENSIONS	
Step	**Branching action**
5a.	No projects match the search criteria. Zen Frontend displays message: “No projects could be found matching your search criteria”
VARIATIONS	
Step	**Branching action**

USE CASE 8	Award badge for project
Goal in Context	To award a digital badge to a youth for a specific project they own
Scope & Level	Secondary task
Preconditions	A champion is logged in to Zen and a youth in their Dojo has a project on the platform
Success End Condition	A badge is awarded to the youth for a specific project
Failed End Condition	No badge is awarded
Primary, Secondary Actors	Primary: Champion Secondary: Zen Frontend, Youth
Trigger	Champion clicks the “My Dojos” button in the profile dropdown

DESCRIPTION	
Step	**Action**
1.	Champion clicks the “My Dojos” button in the profile dropdown
2.	Zen Frontend redirects Champion to the my Dojos page
3.	Champion clicks the Dojo that the Youth is a member of
4.	Champion clicks the “Manage Users” button for that Dojo
5.	Zen Frontend redirects Champion to the manage users page for that Dojo
6.	Champion types the Youth’s name into the search box
7.	Champion clicks the “Filter Users” button
8.	Zen Frontend displays relevant users in the Dojo
9.	Champion clicks the Youth profile
10.	Champion clicks the “Award Badge” button
11.	Champion selects a badge to award from the list of badges
12.	Champion selects the appropriate project from the dropdown list
13.	Champion clicks the “Award Badge” button
EXTENSIONS	
Step	**Branching action**
8a.	No relevant users found. No users are displayed and the Champion must attempt a new search
12a.	The selected Youth has no projects. The Champion can instead fill in a text box of information to provide proof of why the badge is being awarded
VARIATIONS	
Step	**Branching action**
12a.	The selected Youth has projects but none that are relevant to the badge being awarded. The Champion can instead fill in a text box of information to provide proof of why the badge is being awarded

USE CASE 9	Access learning resources related to project

Goal in Context	To access learning resources relevant to a particular project
Scope & Level	Secondary task
Preconditions	A user has accessed Zen and at least one project exists
Success End Condition	The user finds relevant learning resources
Failed End Condition	No relevant learning resources are found
Primary, Secondary Actors	Primary: User Secondary: Zen Frontend, CoderDojo Learning Platform
Trigger	User navigates to the project page for a project
DESCRIPTION	
Step	**Action**
1.	User navigates to the project page for a project
2.	User clicks the “Learning Resources” button
3.	Zen Frontend redirects the User to a relevant CoderDojo Learning Platform page
4.	CoderDojo Learning Platform displays relevant resources to the User
EXTENSIONS	
Step	**Branching action**
VARIATIONS	
Step	**Branching action**

2.4 Constraints

The following are constraints I have identified for my project:

- **Scratch 3 Release:** Since the exact release date for Scratch 3 is still an unknown but is expected to be some time in early 2018, it would be too risky to rely on this to implement Scratch 3 support. For this reason, Scratch 3 support has become a stretch goal of the project. Python will now be focused on as the primary language to support runtime for.

- **Open Source:** Any libraries/packages used must have an MIT open source license to comply with the CoderDojo open source ethos.
- **Version Control System(VCS):** The VCS I use for project versioning must also be used to store the project code and must be free to use. I have chosen GitHub for this purpose.
- **Code Quality:** Code must comply with standards enforced by the CoderDojo Foundation (JavaScript ES6). I will be using a linter for this purpose.
- **Time:** Time is a constraint in any project and this is no different in the case of mine. I have until May to complete this project and have exams and assignments to work on in the meantime. Items on the “wish list” I provided in section 2.2 have been deemed out of scope based on this.

3. Functional Requirements

3.1 Browser Runtimes

- **Description:** I must implement runtime support for Python, HTML/CSS/JavaScript and JavaScript-only projects within the Zen environment so that project may be interacted with by users through Zen.
- **Criticality:** This is the most critical part of my project. Without runtimes, projects would display in a much simpler manner through screenshots but could not be interacted with.
- **Technical issues:** Running Python projects in a browser environment within an existing architecture is technically very challenging and there is a certain risk factor here. I will be implementing this requirement early on in my project schedule in order to mitigate this risk factor by allowing myself more time in case of unforeseen difficulty. HTML/CSS/JavaScript and JavaScript-only projects should not cause as much risk since they are all technologies which are native to browser environments.
- **Dependencies with other requirements:** This requirement only depends on my completed research to decide how I will implement the runtimes.

3.2 Version Control

- **Description:** I must implement version control for all projects uploaded to Zen using an existing version control system. I have chosen GitHub for this purpose. Versioning will allow projects to be returned to previous versions if need be and hosting files on GitHub will allow for project code storage.
- **Criticality:** This is a very critical requirement since no projects can exist on Zen without storage for the project code and resources.
- **Technical issues:** I will be communicating with GitHub via their v4 GraphQL API which will prove to be challenging for me given that I have no experience with GraphQL nor have I ever communicated with a version control system through its API before.
- **Dependencies with other requirements:** This requirement only depends on my completed research into how to use the GitHub API.

3.3 Project Creation & Management

- **Description:** I must implement the ability to create, update and delete projects within Zen.
- **Criticality:** This is a very critical requirement since the ability to run projects within Zen is pointless if projects cannot be created or managed.
- **Technical issues:** I foresee very few technical issues here, the main difficulty will be integrating this functionality into what currently exists on Zen so that it all works together.
- **Dependencies with other requirements:** This requirement depends on (3.1) Browser Runtimes and (3.2) Version Control.

3.4 Project Discovery & Promotion

- **Description:** I must implement searching of projects based on tags and other queries and display of relevant projects on relevant pages (profile page, dojo page, projects page).
- **Criticality:** This is a very critical requirement since projects would not be able to be found by anyone without it.
- **Technical issues:** I foresee very few technical issues here.
- **Dependencies with other requirements:** This requirement depends on (3.3) Project Creation & Management.

3.5 Project Statistics

- **Description:** I must implement tracking of statistics related to projects such as level of engagement from particular user groups, number of projects uploaded, number of times projects were run, popularity of project types etc.
- **Criticality:** This requirement is not as critical relative to others since it only affects a small group of users of Zen in the CoderDojo Foundation. However, it is something that is really important for them and would be ideal to have in the project for first release since that's the best time to start tracking statistics.
- **Technical issues:** I don't foresee too many technical issues here but I have never attempted statistics tracking before so I cannot say for sure. I think the main difficulty will be displaying the statistics in nice formats such as graphs and charts.
- **Dependencies with other requirements:** This requirement depends on (3.4) Project Discovery & Promotion.

3.6 Project Extras

- **Description -** This requirement involves implementing project extras such as linking projects to relevant CoderDojo learning resources and digital badges.
- **Criticality -** This requirement is the least critical to the overall project since it would not affect any other requirements if it was missing but is ideal to have.
- **Technical issues -** I don't foresee many technical difficulties here. If any do arise, it will likely be to do with working with the existing implementation of digital badges on Zen.
- **Dependencies with other requirements -** This requirement depends on (3.4) Project Discovery & Promotion.

4. System Architecture

Zen is built using a microservice architecture with the backend being written in NodeJS and the frontend in Angular 1 with a new frontend currently in development using the VueJS framework. A PostgreSQL database is used for data storage. I will be modifying existing frontend and backend microservices and also creating a new microservice named cp-projects-service which will handle runtimes and version control integration. The system architecture diagram below (Fig. 4.1) illustrates this.

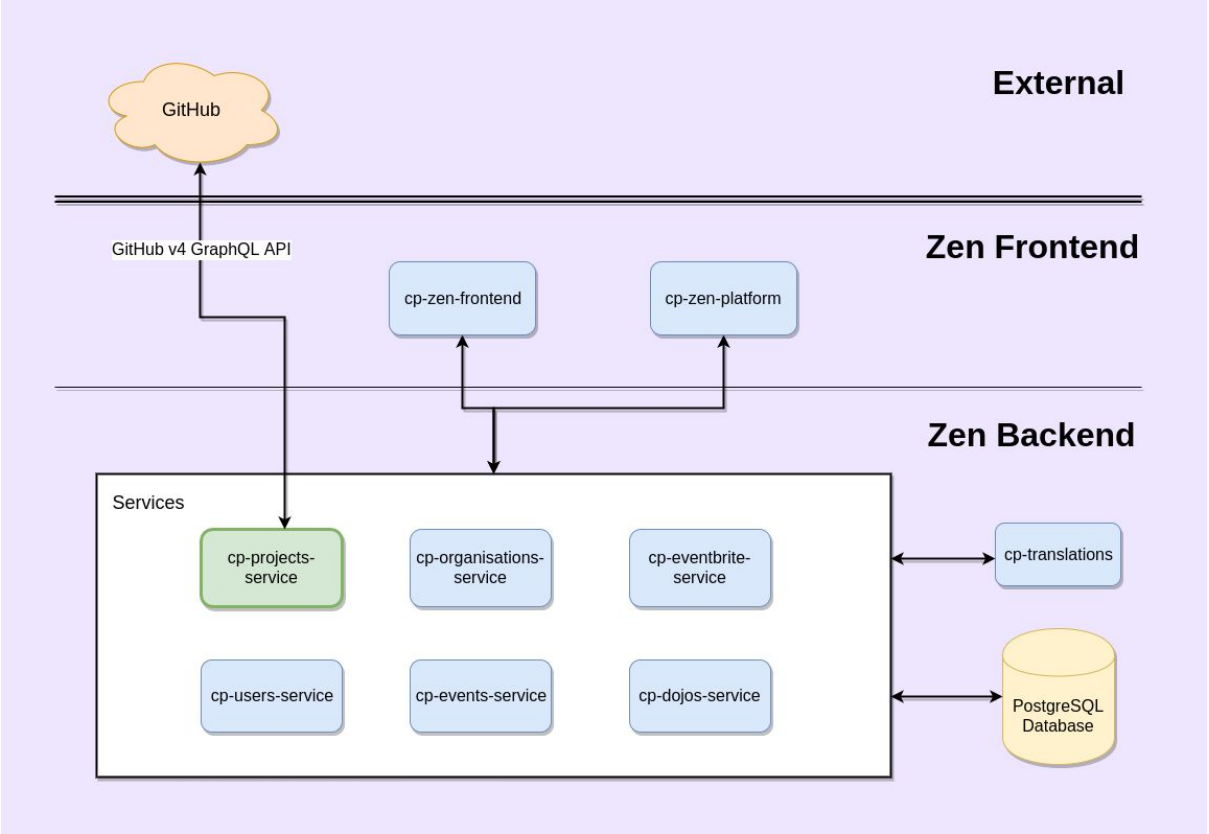


Fig. 4.1

4.1 External

- **GitHub**: Popular VCS which I will be using to support versioning for youth's project code. I will communicate with GitHub via their version 4 GraphQL API.

4.2 Zen Frontend

- **cp-zen-frontend:** Frontend user interface for Zen. Written using the VueJS framework. Existing third party component which I will be modifying.
- **cp-zen-platform:** Legacy frontend user interface for Zen. Written using the Angular 1 framework. Existing third party component which I will likely be modifying.

4.3 Zen Backend

- **cp-projects-service:** I will create this new microservice to handle runtimes for Python and JavaScript with the possibility of also supporting Scratch. This will be an independent microservice which will be abstracted enough that it can be easily expanded upon in future to add support for more languages. Version control for projects with GitHub will also be handled here.
- **cp-organisations-service:** Existing third party component used to run Zen.
- **cp-eventbrite-service:** Existing third party component used to run Zen.
- **cp-users-service:** Handles functionality relating to users of Zen. Existing third party component which I will likely be modifying.
- **cp-events-service:** Existing third party component used to run Zen.
- **cp-dojos-service:** Handles functionality relating to Dojos. Existing third party component which I will likely be modifying.
- **cp-translations:** Contains all strings used in the frontend of Zen and their translations in various different languages. Zen crowdsources its translations using Crowdin - a localization management platform. Existing third party component which I will be modifying.
- **PostgreSQL Database:** Existing third party database which I will be using for data storage relating to projects.

5. High-Level Design

5.1 Context Diagram

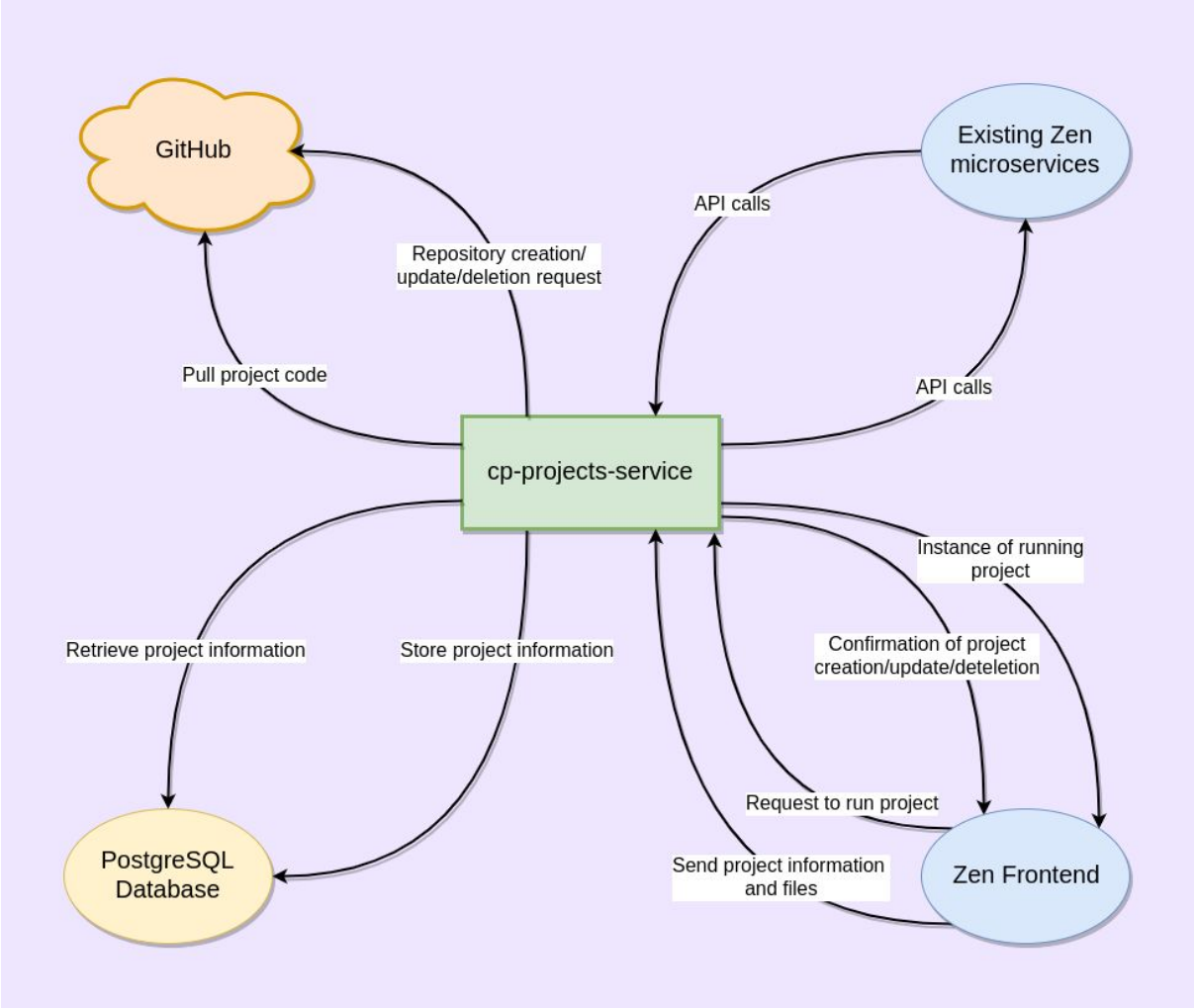


Fig. 5.1

This context diagram (Fig. 5.1) describes the interactions that will take place between cp-projects-service and external entities to itself which may exist within Zen or outside of it (represented by the cloud).

5.2 Logical Data Model

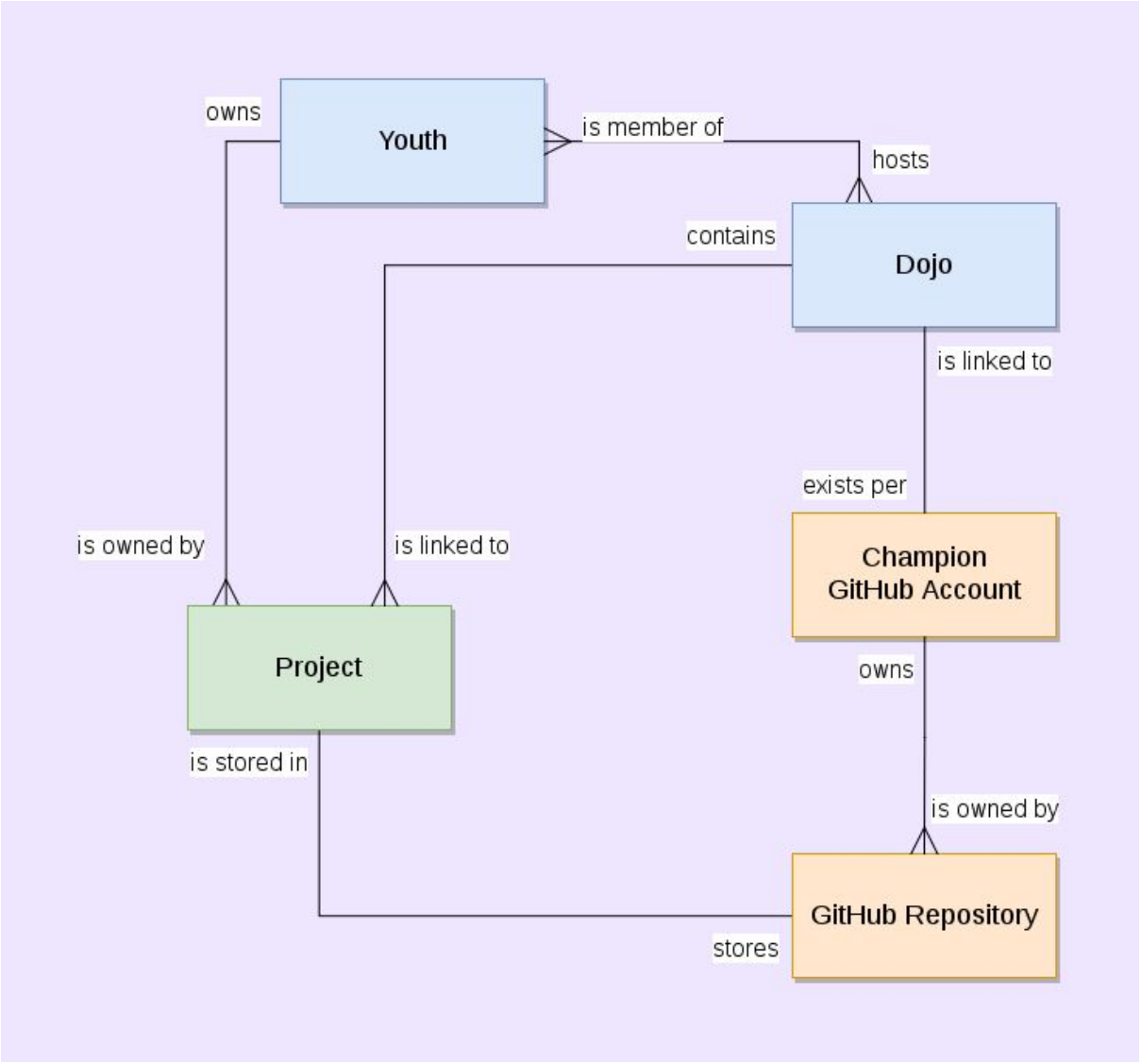


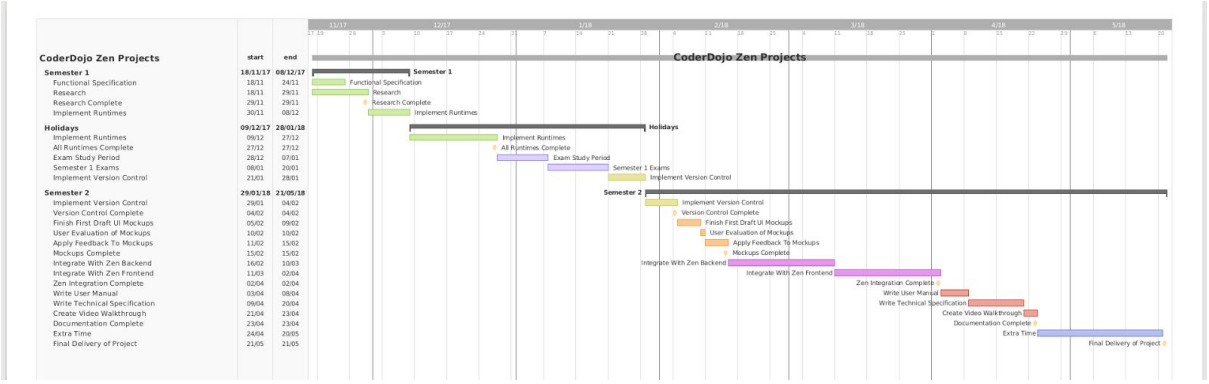
Fig. 5.2

This Logical Data Model (LDM) (Fig. 5.2) highlights the logical relationships which will exist between various entities relevant to my project. The labels describe each relationship from the point of view of the entity they are closest to. A line which expands into three at the end defines the arity of the relationship it represents as “many” whereas a regular line defines the arity of the relationship it represents as “one”. In this way the relationships I have defined are shown in more detail.

6. Preliminary Schedule

6.1 Gantt Chart

The following Gantt Chart shows the breakdown of tasks to be completed as part of my project and the timescale of each task over semester 1, the holiday period and semester 2 of fourth year.



The task labelled “Extra Time” is time at the end of the project schedule which I have allocated for use to complete anything which may have taken longer than originally expected or was delayed due to any unforeseen circumstances. If no such task exists, I will then look to achieving some of my stretch goals during this time such as support for Scratch 3 runtimes.

7. Appendices

CoderDojo Zen platform: http://zen.coderdojo.com
CoderDojo GitHub: http://www.github.com/CoderDojo
My Project Tracker: https://www.pivotaltracker.com/n/projects/2122879