Phase III: Project Report

**Group:** Null

**Members:**

g4zhange Chaoyu Zhang [davidchaoyu.zhang@mail.utoronto.ca](mailto:davidchaoyu.zhang@mail.utoronto.ca)  
c2robins Thomas Robinson [thomas.robinson@mail.utoronto.ca](mailto:thomas.robinson@mail.utoronto.ca)  
g3lty Tianyang Li [tianyang.li@mail.utoronto.ca](mailto:tianyang.li@mail.utoronto.ca)  
g3mao Guohao Yan [guohao.yan@mail.utoronto.ca](mailto:guohao.yan@mail.utoronto.ca)

**Website URL:** [**http://tomrdelhi.github.io**](http://tomrdelhi.github.io)

**Source code URL:** [**https://github.com/tomrdelhi/CSC309-Website**](https://github.com/tomrdelhi/CSC309-Website)

**Summary of Changes:**

* **Log in required!**

Any user must login first in order to use/view any content of our web page.

* **Privacy is important**

User’s profile has options to become private, and gender is an option to fill.

* **Too much information**

We have removed gender and birth date on registration page for simplicity.

* **Social networking system delayed**

We decided to put part to later phase. Including “Add Friend”, “Private message”, and “Wall” features we introduced in phase 2.

**Feature and Functionality Specification with updates**

**Authentication/User Profiles**

For this project, we had envisioned CommunityFund to be a website for raising funds and awareness on community projects. Upon landing on the website, the user has the option to register or to log in before viewing content pages. During the registration process, the user specifies the email and password they wish to use for authentication, their basic information including their first and last names, gender (optional), birth date, address, and postal code. After registration, the user has the chance to indicate their personal interests, as well as skills. This will help the system better locate projects that would be particularly interesting to the user. Some of the information mentioned above will be featured on the user’s profile, namely, their full name, gender, skills, interests, and the general geographical area that the user is located in. A user has the ability to modify their skills, interests, and address. Another piece of information that is available on a user’s profile is their reputation/rating; which is a score that is used to measure the user’s contribution towards community projects. The score depends on community projects in which the user had initiated, as well as projects that the user had funded. The rating system will be discussed in detail in the sections below. For simplicity, user profiles are public for viewing, this can be changed if privacy issues arise.

**Communities**

Since this crowd funding platform is focused on communities, a user must first belong to a certain community before they can initiate or fund community projects within that community.

After the user profile is completed, the user has the ability to join communities by searching.

Communities could also be automatically suggested based on the user’s geographical location and the interests they had indicated on their profile. A community has a profile that features its members and projects. A geographical community, such as a specific neighborhood, checks the user’s address and postal code, and only accepts a user’s request to join if the user lives within its borders. A community based on interest, however, can have members from around the globe. An example could be, “The People Against Puppy Mills”.

The user has the ability to create communities, if one that pertains to their interests does not exist. Communities must be created with a title and a short description, and the creator of the community becomes its first member. After this process, the newly created community can be searched and joined by other users.

**Initiating a Project**

After a user joins a community, they are free to initiate new projects and/or fund existing projects under that community. To initiate a project, the user would press the “Start a New Project” button on the community’s profile. The project will need a title, a description including the purpose, the method, milestones etc., and a monetary goal. After the user publishes the project, members of the community that the project belongs to will be able to see the posting under the community’s profile. They can learn more about it by clicking the project’s title, which brings them to the project’s page. The initiator can update the project with new goals, progress made and additional information, this can be done on the project’s page by the initiator. The initiator is encouraged to update the project’s page to demonstrate its progression, this will attract more funders and ensure that previous funders’ money is put to good use.

**Funding a Project**

A user can browse through the existing projects on a community’s profile and they can donate to the cause on the project’s page. The funder will specify the amount they wish to donate, and can choose to donate publicly or privately. Donating publicly means that members of the same community can see the donation amount, and by who. Donating privately is for those who wish to donate, but would rather their identities be kept anonymous. After the donation is complete, the goal of the project will be updated, and the donation will be shown on the project page for fellow community members to see. Community members may post questions, concerns and testimonials to the project’s page. Communication between community members is important as potential funders may have questions regarding the project before they decide to donate. Also, community members will be able to give feedback and testimonials on the progression and impact of the project.

**Social Network**

Friends, walls, private message features have been removed from this phase because it consist large amount of work with a lower priority. They might still come back in the next phase.

**Reputation System**

A reputation system is in place in order to assess a user’s contribution to their community, and the quality and impact of a project. We have decided to implement a five star system rating as it is simple and effective. Every time a funder interacts with an initiator through a donation, each will be able to rate the other on their interaction. Members of a community have the chance to rate projects under that community without having to donate to that project. There are many reasons as to why a community member might want to rate a project.

A community member may have noticed a positive change in their geographical community as a result of a project and would like to show their support. On the contrary, a community member may have noticed that nothing is being done in their community despite numerous donations for a project, and they would like to express their concerns through a rating and a question on the project’s page. Also, members may wish to rate a project because they support the cause, since funders are definitely more likely to donate to a project with a high rating.

**Administrative View**

Administrator accounts cannot be created through the typical registration process, they are created by the developers. As an administrator, one has the ability to review aggregated information of the system. Data could be aggregated on a single community, a selection of communities, or of the entire database. Some interesting analytics include:

● Total number of projects

● Total number of community members

● Average days to reach a fund goal

● Average goal amount

● Average rating of projects

More statistics can be included in administrative view if necessary.

**Project Plan**

Our team will be using Facebook group for general task assignment and announcements, we also set up Facebook chat room and Skype for meetings and quick notes purposes. We’ve all agreed to use GitHub to make sure code consistency.

We are team of four self-motivated students so there is no “project manager” for this team, we like the idea to self-assign tasks that are available that’s usually posted on Facebook group or GitHub issue management system. In case of encountering problem/difficulties, other team members are always there and willing to help.

We’ve reached our milestone one proposed in phase 2, which is to create a front-end webpage and database that will support all of our proposed features. In the next milestone, we aim to finish the back-end code so that everything will function properly.

**Software Architecture**

**Tools**

As this project is multifaceted we will require a variety of tools to complete it, these include

(More or less in order of frontend to backend):

**AJAX:**

The front end of our web app will be written in a variety of languages and using a variety of tools that collectively fall under the AJAX header. These include HTML and CSS for aesthetic elements and the general “look” of the website, XML and XHL for actions that require the dynamic modification of the webpage but require no or limited interaction with the server and don’t require an entirely new page or the reloading of the current one, and

JavaScript to act as a connector between the front and back ends and between the various front end technologies themselves.

**Django:**

Acting as an interim between the front end and the database, most of the functionality of the app will come from this tool. The functions and classes that define the users and the actions associated with them will all be written within this framework, it will draw data from the MySQL database (see below) and feed it forward to be represented by the AJAX suite of tools (see above).

**MySQL:**

This is the database manager we have chosen to use for our application. There are a number of similar tools out there but at least some members of our group have some familiarity with this one in particular. Here we will store information about our users, communities, and initiatives, to be accessed and displayed.

**GitHub:**

The repository are using to manage and keep our code consistent and organized.

**High Level Structure and Module Descriptions**

Our front end is divided into a number of major sections representing different roles a single individual can play in the community:

First, we have our login and registration section, a page for those who haven’t yet entered the community. This is relatively straightforward consisting of probably only two pages, an initial greeting page and a registration page. The responsibilities of these pages should be relatively self-explanatory, the greeting page takes in credentials (a username and a password) checks them against the database and responds appropriately, the registration page takes in information and passes it back to a handler, which in turn updates the database.

Second, we have the suite of pages used by Funders. These include a browser of all the currently available fundable projects, organisable based on any number of desired criteria.

This section also includes a backing page for when an initiative has been selected; a social media hub page where you can see you’re “interest communities” as well as nearby “geographic communities”. The social media hub also allows you to communicate with Initiators, rate them and leave testimonials on initiatives.

We have our final public pages in the form of the Initiator suite. Here an initiator can find a page for each initiative he initiates. From these pages he can communicate with his funders, add additional funding goals to his project, post updates on its status and so on. He can also create a new initiative from one of these pages, specifying financial goals, purpose, and associated community. Finally an initiator can create a new community, this doesn’t require an associated initiative, merely a description and a specification as to whether the community is geographic or interest based. The creator of a community begins as the sole moderator of that community, though he can promote others to that level. As moderator he or she can remove users from the community, flag troubling initiatives and so on.

Finally, we have the private administrative page, visible only to those users identified as administrators. From here an administrator can look at metadata from across the site, this gives him access to pretty much all information from the site database (separate from the password database, see below). This information is contained in a single page, where the administrator types in a query and receives information.

All of the above will function based of the basic MVC model. Next we will move towards the backend and see what major classes and functions will create the underlying structure for the app:

We first and foremost have a user class; this acts a connecting feature between the user as an initiator and as a funder. The user class itself only needs to model functionality shared across all user types. Information about the user’s communities and basic information (name, location, etc.) can be stored at this level. Furthermore two objects, one of the initiator type, and one of the funder type are also both held here, this allows easy switch over from one role to another. The user class’ other main functionality is drawing information from the database to hold in the object as it’s being acted upon and write it back to the database once this is no longer the case. In each user object we have an object of the funder type. This represents the user as a funder and as such has a series of context specific functions. For example the ability to back an initiative, the ability to chat and the ability to add communities all fall under this class’ responsibilities, furthermore reviews, testimonials and rating functions are inside this class’ scope.

On the other hand, each User object also includes an Initiator object. The initiator class is responsible for all initiator exclusive functionality, this includes such things as creating or modifying an initiative, creating or modifying a community and moderating a community. It includes information such as list of initiatives and current rating.

Administrators will not get their own object, but rather be a characteristic of a user object.

Each user object will have a Boolean set to false by default that determines whether they have access to the set of administrator methods. These include accessing the administrative pages and absolute control over community pages. This gives us control over the types of communities that can be created, making sure there are no “interest community’s for things like racism.

We can move away from users now and focus on the other aspects of the site. Community objects allow us to manage everything from creator and other basic information, to a list of group members, to a list of relevant initiatives. Access to most community object methods is restricted to moderators of a given community.

Initiatives objects represent a given initiative. Information in a given initiative object includes backers, initiator, associated community, pledged amount, description and so on.

Initiative function access is restricted to the initiator.

Finally, we have controller objects for each of the front end pages, and connective functions to associate the python framework with the database. These are required on a simply functional level.

To get a better idea of the interactions between these parts, see the following diagram:

**Test Strategy and Test Plan**

We plan to implement most of our backend features and functions with test driving methods.

For the frontend applications we will conduct unit testing from Python’s standard Django testing module which will acts like a dummy Web browser and allow view and interaction with our Django application. The frontend testing should be done by one or two of our team members and double checked, verified by others.

**Information Representation**

**ER model**

There are three main entities: User, Project and Community.

**User:** There is an unique id (UId) for each user. Email and Pwd (password) are for user authentication. Each

user has reputations for initiating and funding a project (IRep, FRep). UClass is used to identify whether a

user is an administrator or not. For instance, 0 stands for a normal user and 1 stands for an administrator.

There is also personal data for each user. One or more users can either start or fund a project. A user can

belong to one or more communities.

**Project:** A unique project id (PId) for each project. Each project has a description (Desp) and a reputation

(Prep). A project belongs to a community according to ComType and ComId. Fund required by a project is

indicate by AmountReq. And AmountGot shows how much fund is now raised.

**Community:** A community has one unique CType and Cid combination. Because communities are differ

from types (location or interest). Id is unique for each community type.

\*Over time, this data model will change slightly during project development.

**Schema**

There are three tables for each main entity. Primary key is in bold. More than one attribute in one box

indicates a superkey.

**UserSP** records users start a project with a date. **UserFP** records users fund a project with a date.

**ComPost** stores users’ posts within a community. Similarly, **ComPostRe** is a table to store users’ reply to a

post. **ProjectRev** is used to store reviews about a projects from users.

\*Over time, this schema will change slightly during project development.

UserSP(UId) ⊆ User(UId)

UserSP(PId) ⊆ Project(PId)

UserFP(UId) ⊆ User(UId)

UserFP(PId) ⊆ Project(PId)

ComPost(CId, CType) ⊆ Community(CId, CType)

ComPostRe(PoId) ⊆ ComPost(PoId)

ProjectRev(PId) ⊆ Project(PId)