CSC309 Team Project: Phase 2

Harris Chaudhary, g2harris
Raymond Tieu, g4tieura
Mihir Jain, g3mihir
Tri Tran, c2trantr

Feature and Functionality Specification

Users ...

- can register with a unique usernames
  - requires the user to fill in form with their name, location, interests to set up initial profile
  - gets suggested communities based on location and interests
- can log in using the username they registered with
  - an error message will appear if there is an authentication error (e.g. incorrect password)
- can make changes to their profiles
  - able to update personal information, location, interests
- can select which communities to join
- in the same community can chat with each other
  - chat room for all users in the same community
- can be initiators or funders, but cannot be both for the same project

Initiators can ...

- post a project to raise fund
  - requires the user to verify personal information (enter full name, billing address, payment information, etc.)
  - requires the user to provide a project description, milestones, funds required
- update project details
  - update description, milestones, keep community up to date on progress
- specify community for project
  - select a community based on community's location/interests
- view list of funders for each of their projects
- rate funders that have funded their projects

Funders can ...

- fund a project
- navigate through listed projects
  - can view all projects in a community
- give testimonials

  - provide feedback to the initiator about their project if they've funded them
- rate completed projects that they have funded
- rate initiators of projects that they have funded

Admins can ...

- set global variables used in the system
- view total number of projects and how many are being funded
- view average days to reach the funding goal
- delete projects
- delete communities

Use Cases

- user has a project in mind; doesn't have funds to support dream
  - sets up/joins community
  - starts a new project
  - fills out appropriate project information and additional user information
  - gets funded by interested funders
- user wants to fund projects they find interesting
  - they browse within their community
  - they select a project that interests them
  - they click to fund the project
  - they enter necessary payment information, if not already saved in profile
  - user has funded a project

Scenarios

- Initiator
  - an initiator registers on the website
  - they submit a project they want to be funded
  - other users fund they project if they find it interesting
  - the project is funded if it reaches the goal
- Funder
  - a funder registers on the website
  - they navigate through posted projects
  - they fund a project they find interesting
  - they give a rating and/or testimonials on the project once funded

Project Plan

Team Organization

- we will communicate, coordinate, make decisions, resolve problems using Facebook
- we will schedule meetings and discuss about project matters at least once a week
- we will update our progress during these meetings

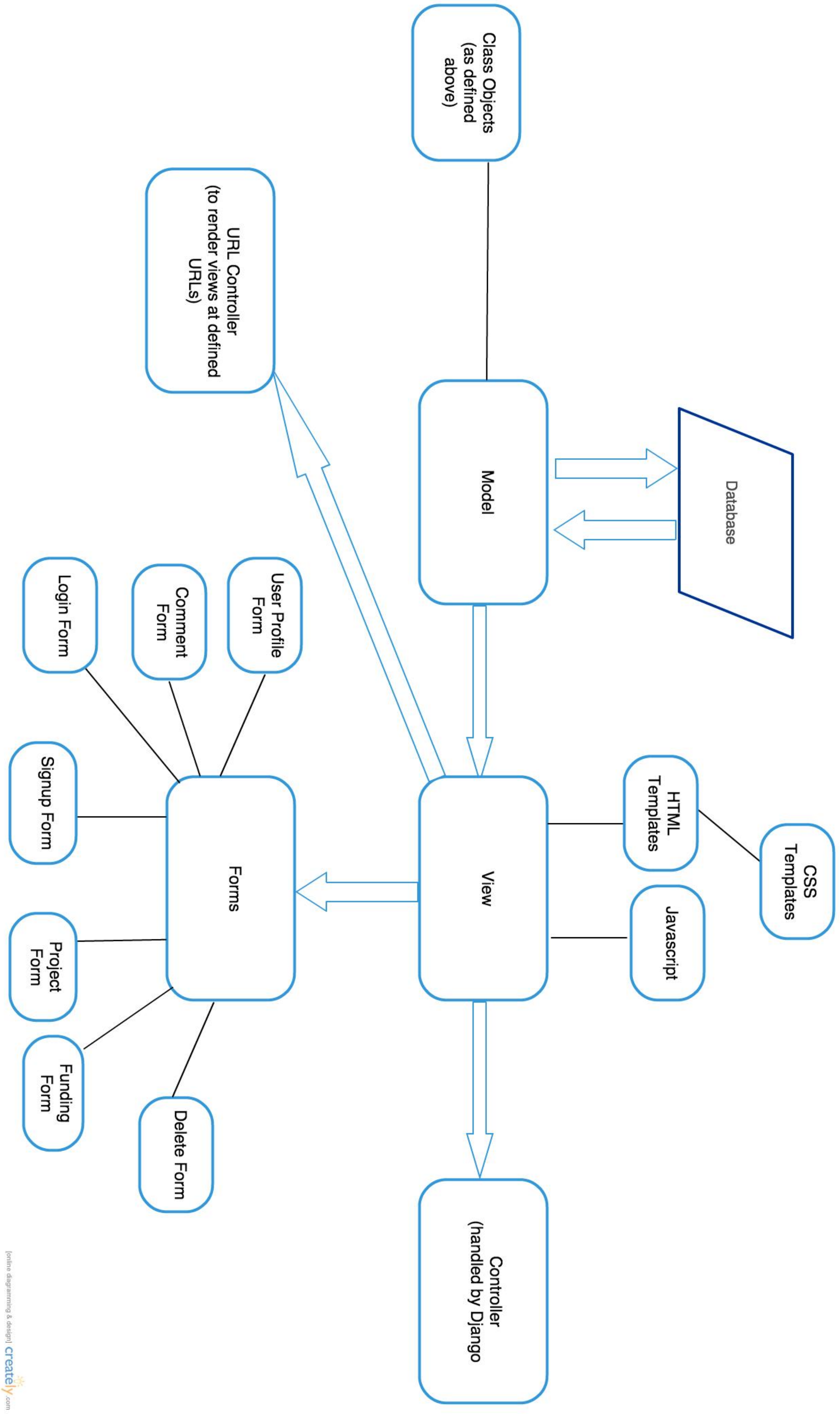
Project Milestones

1. set up the database models in accordance with our information representation and software architecture plans
2. implement base functionality/class variables for each class
   (a) user class will be first (setup login/logout, passwords, etc)
   (b) community class (message board, descriptions, etc)
   (c) user derived classes (see below)
   (d) reputation system
3. after finishing base functionality, implement HTML formatting
   (a) construct a reasonable login form
   (b) Javacript functions to provide appropriate functionality
   (c) construct appropriate DRY forms
      i. form to join community
      ii. form to enter payment info
      iii. form to update user profile
      iv. form to submit messages to message board
      v. form to enter payment info when funders fund projects (if payment info not saved in profile already)
   (d) use AJAX to submit these forms in background
   (e) use Celery to calculate stuff in background
4. CSS to format elements
   (a) make everything look pretty and unified (base CSS template)
   (b) HCI/web design principles
   (c) simple, functional, and easy to use

Software Architecture and High-level Design

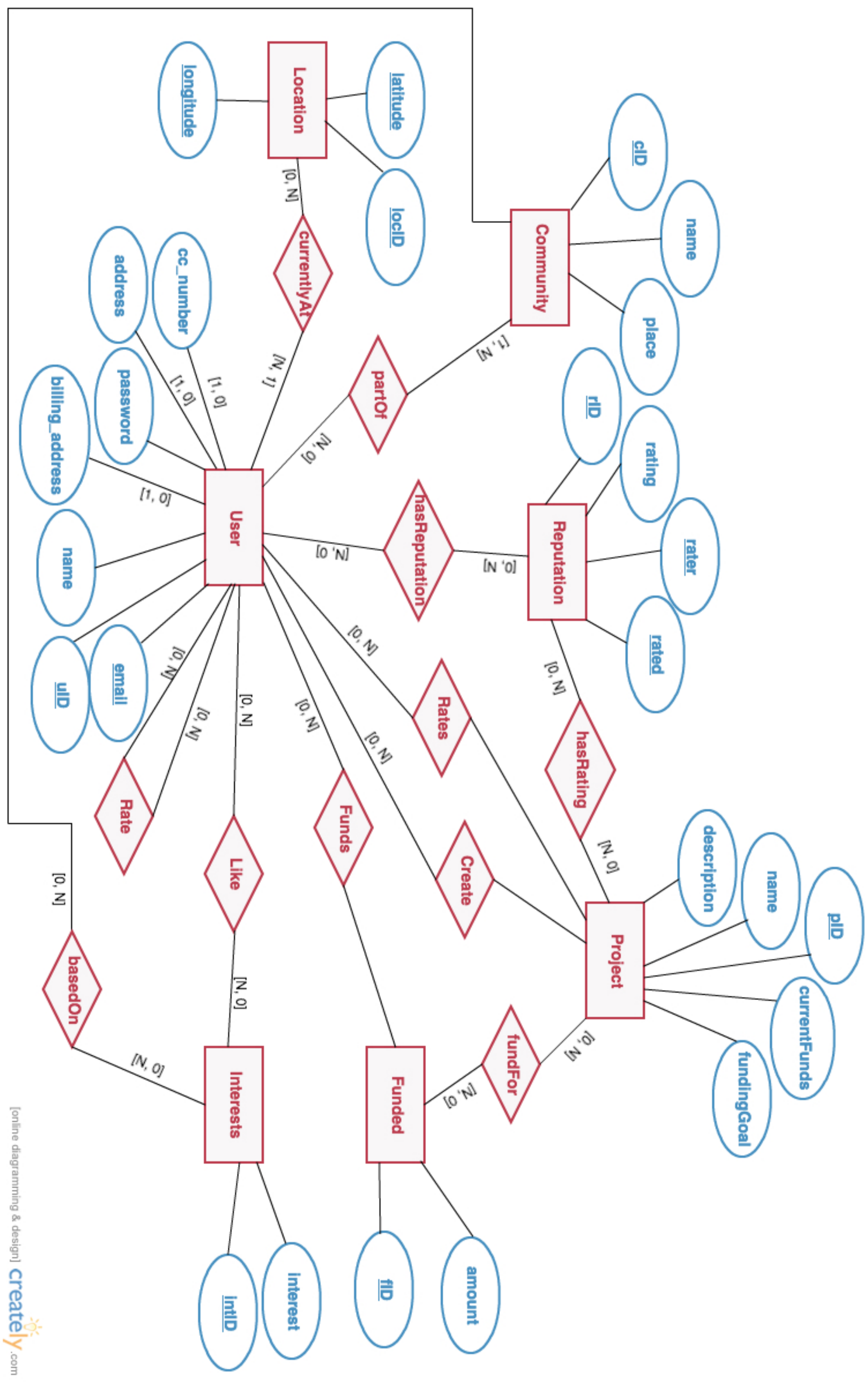
- User class (django derived)
- Community class (consists of users, projects, description, name)
- Project class (associated with one user (the initiator), name, description, funding needed, current funding)
- Reputation class (rated, rater, up/down)
- Funded class (user, project funded, amount)
- All users are authenticated
- Users have profiles that can be updated and can store payment info
- Users can fund or start projects
- Project initiators can not fund their own projects
- Users, communities, funders, and initiators all have reputation
- Communities:
  - central message board
  - description, project summary, funding summary (funding goal vs funding achieved)
- Users signup with basic information (email, username, and password)
- When a user wants to start a project, they must enter additional info
  - Additional info in user profile (address)
  - Additional info related to specific project (company name, company info, project description, project name, project funding goal)
- When a user wants to fund a project, they must enter additional info
  - Credit card information and billing address
  - Optionally, user may choose to store this information in their profile so they don't have to enter it each time
- Users must signup to view projects. They are automatically placed in a community according to their location. They can remove themselves from this community later.

Software Architecture and High-level Design


Class Objects (as defined above)
URL Controller (to render views at defined URLs)
Model
Database
View
HTML Templates
CSS Templates
Javascript
Forms
User Profile Form
Comment Form
Login Form
Signup Form
Project Form
Funding Form
Delete Form
Controller (handled by Django)
[online diagramming & design] creately.com

Information Representation

ER Diagram



Schema

- User(uID, email, password, name, address, billing_address, cc_number)
  A tuple in this relation represents a user registered in the system. The email is the username used to log in. The addresses and credit card number are optional during registration.
- Project(pID, name, description, fundingGoal, currentFunds)
  A tuple in this relation represents a posted project in the system. The pID is the id used to identify the project. Each project includes a name, description and specifies the funding needed to be completed.
- Funded(fID, amount)
  A tuple in this relation represents a donation a user made to help fund a project.
- Community(cID, name, place)
  A tuple in this relation represents a registered community. Each community has a cID to identify them, a name and a place it is based on.
- Interests(intID, interest)
  A tuple in this relation represents an interest.
- Reputation(rID, rated, rater, rating)
  A tuple in this relation represents the rating given to either a project or another user by a user.
- Location(locID, longitude, latitude)
  A tuple in this relation represents a location.
- Create(pID, uID)
  A tuple in this relation represents that the user created a project.
- hasRating(rID, pID)
  A tuple in this relation represents the rating of a project given by a user.
- hasReputation(rID, uID)
  A tuple in this relation represents the rating of a user given by another user.
- partOf(cID, uID)
  A tuple in this relation represents which community a user is a part of.
- fundFor(fID, pID)
  A tuple in this relation represents a fund given to a project.
- Funds(fID, uID)
  A tuple in this relation represents a fund given by a user.
- Like(uID, intID)
  A tuple in this relation represents an interest a user has.
- Rate(uID, uID)
  A tuple in this relation represents a rating by a user to another user, or a rating by a user to a project where the uID of the rater cannot be the same as the uID of the user rated.
- Rates(uID, pID)
  A tuple in this relation represents a rating by a user to a project.
- currentlyAt(uID, longitude, latitude)
  A tuple in this relation represents the location of a user.
- basedOn(cID, intID)
  A tuple in this relation represents the interest that a community is based on.

Integrity Constraints

- Π_{rating} Reputation $\subseteq$ {0,1,2,3,4,5}
- Create[uID] $\subseteq$ User[uID]
- Create[pID] $\subseteq$ Project[pID]
- hasRating[rID] $\subseteq$ Reputation[rID]
- hasRating[uID] $\subseteq$ User[uID]
- hasReputation[rID] $\subseteq$ Reputation[rID]
- hasReputation[uID] $\subseteq$ User[uID]
- partOf[cID] $\subseteq$ Community[cID]
- partOf[uID] $\subseteq$ User[uID]
- fundFor[fID] $\subseteq$ Funded[fID]
- fundFor[pID] $\subseteq$ Project[pID]
- Funds[fID] $\subseteq$ Funded[fID]
- Funds[uID] $\subseteq$ User[uID]
- Like[uID] $\subseteq$ User[uID]
- Like[intID] $\subseteq$ Interests[intID]
- Rate[uID] $\subseteq$ User[uID]
- Rates[uID] $\subseteq$ User[uID]
- Rates[pID] $\subseteq$ Project[pID]
- currentlyAt[uID] $\subseteq$ User[uID]
- currentlyAt[longtitude] $\subseteq$ Location[longtitude]
- currentlyAt[latitude] $\subseteq$ Location[latitude]
- basedOn[cID] $\subseteq$ Community[cID]
- basedOn[intID] $\subseteq$ Interests[intID]

Test Strategy and Test Plan

Login and credentials

- testing we can login with correct username and password
- testing if we get errors for attempting to login with wrong credentials
- testing if username and password is correctly recorded in database

Profiles

- testing a funder can fund
- testing an initiator can initiate a project
- testing if a funder decides to be an initiator
- testing if a initiator decides to be a funder
- testing if a funder/initiator can not fund his/her own project

Project

- testing if initiator creates a fund correctly
- testing if funds are recorded in database
- testing if fundees are recorded in database
- testing for fund accumulator
- testing if project details are true
- testing if project can be updated

Community

- testing if we are part of a community
- testing if we are friends with another person in the community
- testing if posting on community wall works
- testing if users are grouped by interests

Reputation

- testing if a rating is recorded
- testing if initiator reputation is the average of all its respective ratings
- testing if project reputation is the average of all its respective ratings

The tests above will be done with unit tests. Integrity testing will be done by using the website manually.