CS-6083 Project Report

A Database-backed Website for Crowdfunding

Duan Wang

dw2201@nyu.edu

Jintian Zhu

jz2690@nyu.edu

# Outline

This report can be divided into four parts. The requirements part first briefly introduces our understanding of the project and lists the most crucial requirements that this project must realize. The requirements defined in this report are scenario based, they might be implemented by database design, web user interface design or web application function. The next design part introduces in detail how our database and web application are designed. This part is followed by implementation part, states in detail how our database is defined, how the webpages are developed to meet the requirements. In the last part we make up a story to better illustrate how this website is used.

# Requirements

The main task of the project is to design a crowdfunding website that allows user to start a fundraising campaign, or, as a sponsor, pledge money to some project. Similar to other database-backed websites, the website should allow basic manipulation on account, which includes login, logout and register, because in the project only limited information is needed for a user, account setting function is not going to be provided. As a crowdfunding website, this system should also provide interface for user to generate a new fundraising campaign and pledge certain project. When the minimum needed amount of money is raised for a project, all the sponsors are charged. Such projects are regarded as successfully funded. And after a project succeeds, the owner can inform the sponsors of their progress by publishing some description text or videos and photos, project owner can define the version number of current progress. In this project, sponsors will not get anything back as reward for pledging a project, but sponsors can comment on current progress. After a project has been completed, its sponsors can rate it. To provide some social networking features, user can follow other users, and like some projects. In this project we do not need divide user to different types as each user can start fundraising campaigns and pledge other projects.

Also as a web application it shall provide continence to users as much as possible. Here we list some features that could facilitate users to browse projects as other requirements that we aimed to achieve in our project:

-The website could automatically provide information that a user might pay attention to(feed).

-User can track a list of actions by other users related to him or her (e.g. follow relationship, project owner and sponsor).

-User should be able to browse the sites easily.

-User can use keywords or tags to match projects(searching).

-If certain act has relation to a user, say a project the user own is pledged or commented, the user will receive a notification next time he or she log in.

The use-case diagram of this project is shown below.



Figure 1. Use-case diagram

Further, there should be security requirements, such as preventing SQL injection, avoiding repeat sending form, they should also be considered.

# Design

## Database Design

Based on the use-case diagram, the database should contain an entity to store user accounts, an entity to store projects, an entity to store progresses of each project, and also an entity to store comment from each sponsor of a project progress. Further, user entity need have relations to other users to include the following information, and user can own a project, pledge a project, and like a project, so user entity should have different relations to the project entity. Further, a progress is attached to a project, and a comment is related to a user and a progress.

Entities shall include User, Project, Progress and Comment.

-User should have a unique user name (user name for login, could be an email address.) as primary key. Also, a login password is required. When role as a sponsor, the credit information should be required, and in other case, it is optional. When a user pledge, we will check his/her credit information is valid or not.

The relationship between two users is Follower, which is a multi-multi relationship.

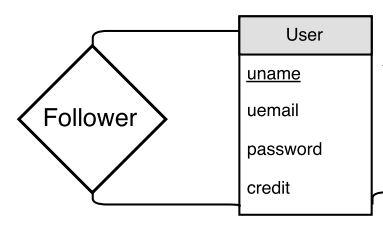


Figure 2. User and Follower

-Project should own a unique project id which is automatically produced by system. A project has one and only one owner: if a project is proposed not by a person but a group, for example, a band,then the band members can use one of their personal accounts or register a new account for their band to be the owner of the project. The project should include essential information for the funding: minimum amount to start the project; maximum raising amount; start date and end date of the fundraising campaign, and current amount sponsors have pledged to this project as an efficiency consideration. Also, information about the project itself includes: description (including txt, videos, images or URL, this part will be further discussed and elaborated), project name, project tag (keywords for searching) and project status.

The relation between project entity and user entity can be owner and sponsor, or liker and being liked. As we discussed above the owner relationship is one-to-many relationship: a project only has one owner. And for the like relationship, a user can like many projects, and a project can be liked by many user, so it is a many-to-many relationship. Similarly, as for sponsor, the relationship is typically many-to-many relationship.

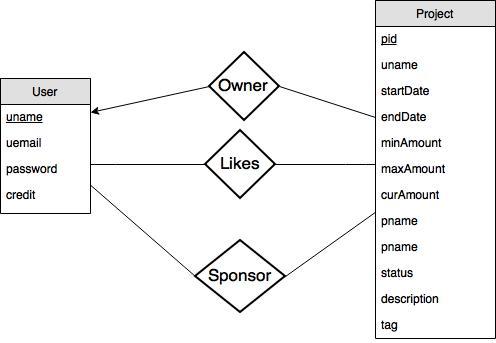


Figure 3. Project and its relationships

The sponsor relationship should contain the following additional information: the amount this sponsor pledges; the rate the sponsor marks (when project finishes), with default value null; a discuss attribute as an optional field.

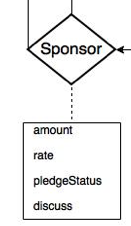


Figure 4. Sponsor

-Progress entity stores updates of projects that are successfully funded. Progresses are ordered by version number. The version number is automatically formed by system. Each progress should contain some description, which can be a multi-media form, submitted by the project owner.

The relationship between progress and project is one-to-many relationship.

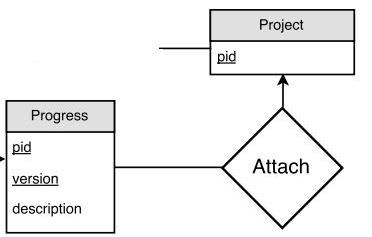


Figure 5. Progress and its relationship

-Comment entity is a weak entity which associates with sponsor and progress. And comment itself has a date information and comment content.

Each comment will associate one sponsor with a version of the project progress.

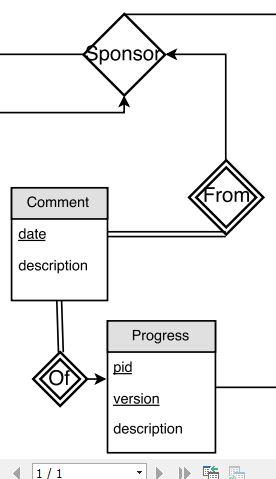


Figure 6. Comment and its relationships

Combining the schema design above, the final ER-Diagram is shown in Figure 7:

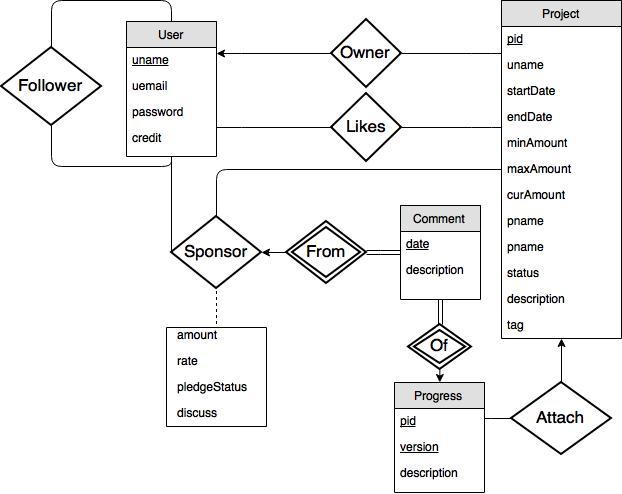


Figure 7. E-R diagram

## Web Design

According to the web-based user interface requirements, our website should have following major pages and the relations and story lines are shown below:



Figure 8. Relations and story lines of major pages

The user can login and register a new account and enter the main site. The site contain access to the Project page where a single project detail will be displayed and several function related to project can be done here, such like the owner of project update his funded project or a sponsor of this project have comment on it. The profile page contains the basic information of a single user. Of cause, a user can change his or her profile information via this page. A user also can create a new project on his profile page and the new project will then exist and can be accessed on main page.

According to this basic framework, we add addition function on our site base on the requirement and design it as convenient to a user as possible. To assurance the convenient browse of sites, we first design a navigation bar on every site pages exclude login page and registration page. The navigation bar provide assessment to the login user’s profile page and the main page and the logout button. Thus, user can access the most important function of our site whichever page he or she is currently in.

1.JPG

In the main page, we also add a search bar at the top of page. The main page is the only place user can traverse all the project, thus we part our main page into several rows to display different set of projects by different function.

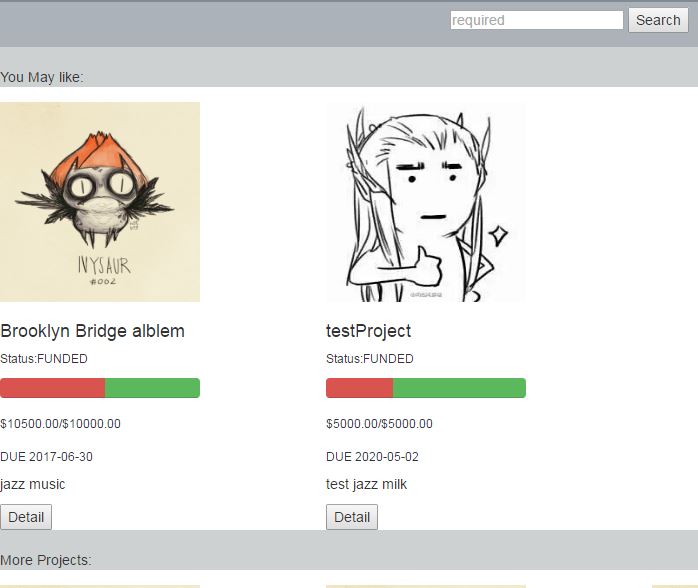


Figure 9. Main page

The project page contains the detail information about a single project as well as sponsor’s comment and rate information. The users’ basic information will be displayed here and we provide the quick access to user’s profile page via user name related to comment or project itself. The basic function like, rate and pledge will be settled here.

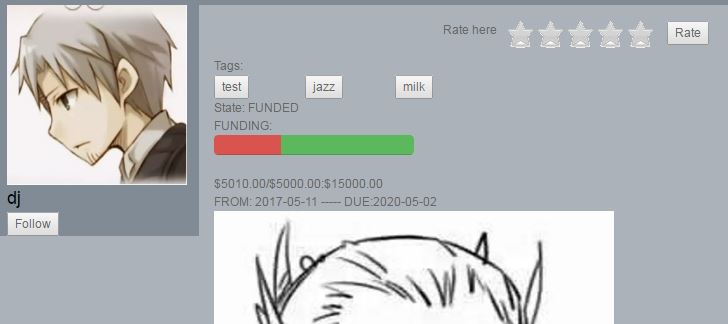


Figure 10. Owner basic information and project basic information



Figure 11. Comment List

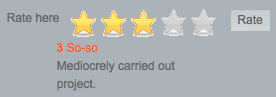


Figure 12. Rating

The progress of a project will be represented as a progress list on project page. So, users can easily compare the difference between versions and get information about other sponsor via different version.

The pledge behavior should happen towards a single project. User can pledge a project on project page. (of course a user can’t pledge to his/her own project) After pledge, user should return to this project page and will able to comment and rate (if project funded or success). A sponsor will be charged on his/her default credit card on a new card where he/her can refilled on the pledge page itself.

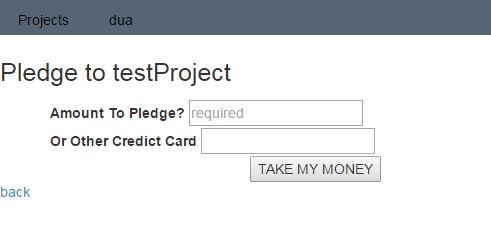


Figure 13. Pledge Page

We also place the notification function on our navigation bar. We integrate some feed and notification function inside this button and an extra notification page. The user can access new action related and notification page will provide quick access to location the behavior happened.



Figure 14. Notifications

The profile page contains the detail information of a user. If user enter his/her own profile page, function assessment of create project and edit new project will be given. Also, we list some related project brief board on this page thus user can easily manage all his necessary behavior here in single page. The project list including user’s own projects, projects that user pledged on, projects user liked.

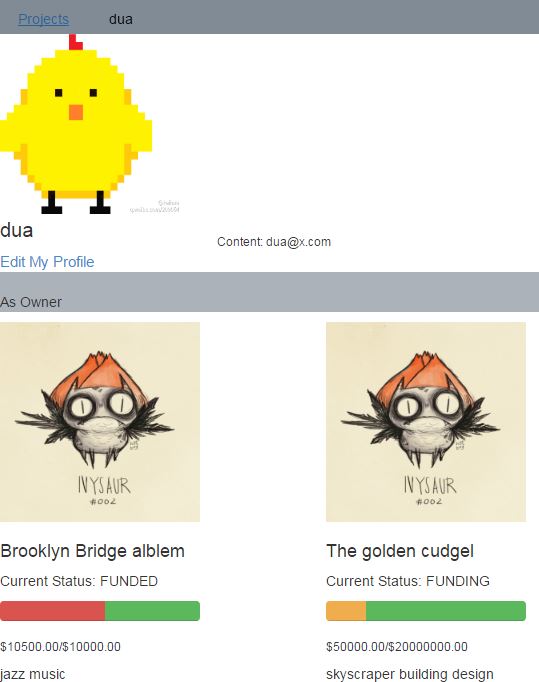
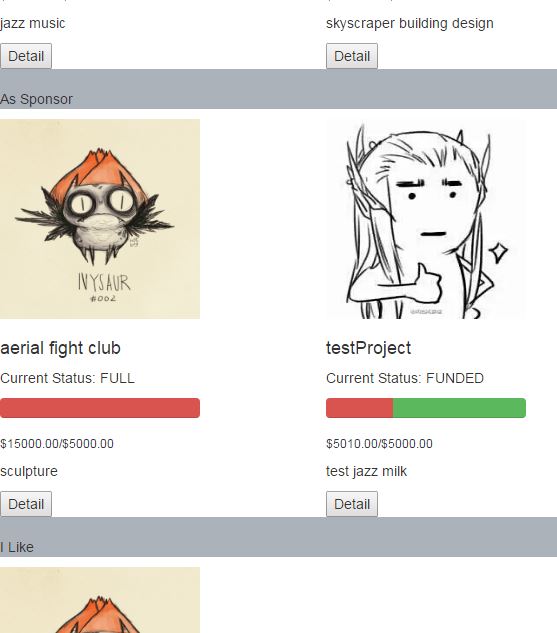
 

Figure 15. Profile page. Right subfigure follows the right subfigure

To create a project, the user should provide the basic information, and optional add additional information like picture and description. Similar with creating a project, editing user profile also requires some compulsory information and optional additional information.

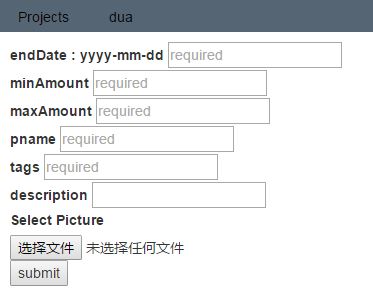


Figure 16. Create project

The project brief board appear times above is designed as basic access against a project. Some important information user may have interest in are displayed on it including the status and pledge progress and some brief tags and brief images. The project brief board will be used on both main page and search result page and profile page.



Figure 17. Project brief board sample

# Implementation

## Database Schema

### Tables

Four entities and two many-to-many relationships form the 7-tables schema:User, Follower, Project, Like, Sponsor, Progress, Comment.

-User: uname, uemail, password, credit

-Follower:uname, funame

-Project: pid, unmae, startDate, endDate, minAmount, maxAmount, curAmount, pname, status, description, tag

-Like: pid, uname

-Sponsor: uname, pid, amount, rate, pledgeStatus, discuss

-Progress: pid, version, description

-Comment: uname, pid, version, date, description

Since the Follower is a many-to-many relationship, two users’ primary keys should be contained. we name one of them as 'funame', meaning the follower's uname. Similarly, Like and Sponsor are many-to-many relationships, so the primary key of user and project should also be contained. The primary key of Sponsor thus is uname and pid.The relationship of Progress and Project is many-to-one, so 'pid' should be included in primary key. Comment's primary key should contain Sponsor's primary key as well as Progress's primary key.

Note that the ‘description’ attribute in Project, Progress, and Comment is just generalized concept. Because description can include text, photos, and videos, we actually use multiple attributes to store description.

### Other constraints

In Project table, the curAmount cannot be larger than maxAmount. Status shall be relevant to curAmount, startDate and endDate, for instance, the status attribute should not be ‘FUNDING’ if current time is later that endDate. Because we can set the inserting query when a fundraising campaign is created, normally unreasonable tuple would not exist.

In Sponsor, the amount attribute cannot be larger than ‘maxAmount - curAmount’ of the corresponding project.

### Analysis

User contains the following functional dependencies: uname→uemail, uname→password, uname→credit. These three functional dependencies form the canonical cover of User table. Because for each dependency the left set (uname) is the super key of User, this schema is in BCNF.

There is no functional dependency in the Follower schema.

Of the Project schema, the super key also contains only one attribute, so for each of its functional dependency, the left set is the super key, thus this schema is in BCNF.

There is no functional dependency in the Like schema.

Sponsor contains the following functional dependencies: uname,pid→amount, uname, pid→rate, uname, pid→pledgeStatus, uname, pid→discuss. For each dependency the left set is the super key of the schema, thus Sponsor is in BCNF.

The canonical cover of Progress is only pid, version→description, and pid, version constitute the super key, so it is also in BCNF.

In the Comment table the only functional dependency of canonical cover is uname, pid, version, date→description, this schema is in BCNF as well.

## Test database with sample data

### Sample Data

The sample data we inserted into the database for testing can be found at the following URL.

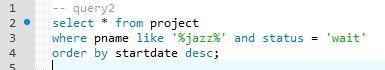
*https://github.com/Dwan9/DatabaseCrowdFunding.git*

### Testing queries

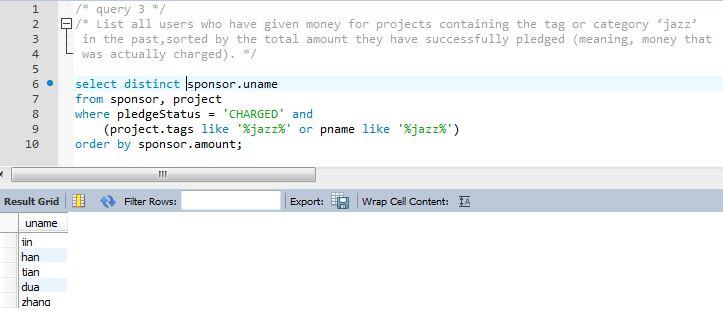
- Create a record for a new user account, with a name, a login name, and a password.



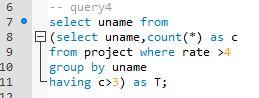
- List all projects that contain the keyword ‘‘jazz’’ and that are currently looking for funds, sorted in descending order by posting time.



- List all users who have given money for projects containing the tag or category ‘‘jazz’’ in the past,  
sorted by the total amount they have successfully pledged (meaning, money that was actually charged).



- List all users who have completed at least 3 projects, and where each of their projects received an average rating of 4 stars or higher from its sponsors.



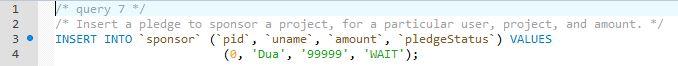
- List all comments by users that are followed by user ‘‘BobInBrooklyn’’.



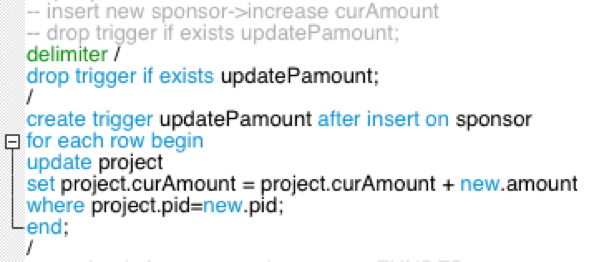
- Insert a new project for a particular user, with a name, description, and other needed info.



- Insert a pledge to sponsor a project, for a particular user, project, and amount.



- Write queries for the end of a funding campaign. E.g., you could use triggers to detect when a campaign is fully funded or time is up; if successfully funded, generate charges to sponsors’ credit cards.



### Other triggers defined for the database

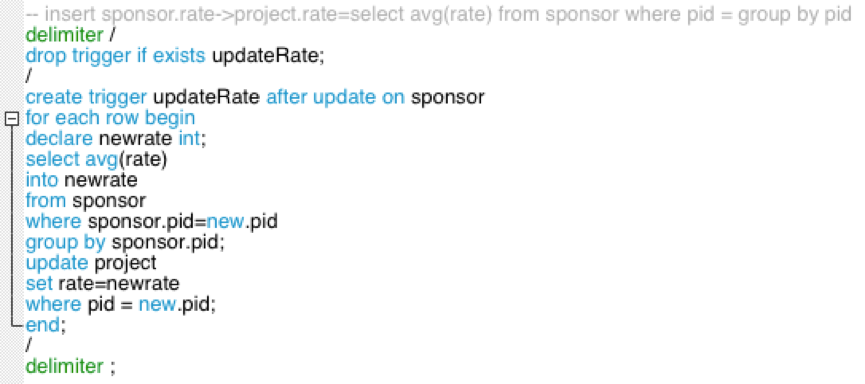
- Automatically alter a project’s status if current funding reaches minimum amount or maximum amount.



- Search every day for fundraising campaigns whose end date arrives, and set project status to ‘FAIL’.



- Every time a sponsor has rated a pledging experience, automatically recalculate average rating of the project and update it.



## Website Implementation

In this subsection we introduce in detail how we build the website and solve the major functional problems to realize the requirements shown above, and include some key codes.

### Login

Login function require user’s username and password to login. Once success match to our database, the session will start with user’s login username as ‘username’.

$\_SESSION["username"] = $loginname;

### Logout

To logout function is easy to implement, we provide a button in the navigation bar so the user can logout in nearly every page. When logout button is clicked, the server jumps to a page that destroys the current session, and goes back to the login page. Because we define each page to check session first, if the browser did not login but call the URL directly, she will also be sent back to the login page. In this method the logout act is realized.

### Follow/Unfollow someone and like/unlike some project.

Implementation of these two use cases are put here to be introduced together because our method to realize them are quite similar. The two functions connect to the database, and there are buttons in the profile and project pages that help the user to choose to either follow someone or like some project. We first test if the user is owner of the project, otherwise she cannot like her own project, similarly the “Follow” button is not available to follow herself. Then the page sends SQL query to check if there exists tuple of the relation that the button is about to create, if so, the button shows “Unlike” or “Unfollow”, and clicking the button will lead to deleting an existing tuple, if not, the button shows “Like” or “Follow”, and clicking the button will lead to inserting a tuple. For example, codes to test if the user is already a follower of the project owner are shown below, and the codes to test if the user has liked a project is very similar to these.

if($loginname != $owner){

$isFollower = mysqli\_query($db, "select \* from follower where uname = '$owner' and funame = '$loginname'");

if (mysqli\_num\_rows($isFollower)==0) {

echo "<button onclick=\"location.href='Follow.php?uname=$owner&f=0&pid=$pid'\">Follow</button>";}

else{

echo "<button onclick=\"location.href='[http://127.0.0.1/Main/Follow.php?uname=$owner&f=1&pid=$pid'\](http://127.0.0.1/Main/Follow.php?uname=$owner&f=1&pid=$pid'%5C)">Unfollow</button>";} }

With different parameters, the page jumped to either inserts a tuple in “Follower” table, or deletes a tuple in the “Follower” table, then the user is leaded back to the project or profile page. For example, the codes related to inserting a tuple into the Likes table are shown below.

$like = mysqli\_query($db, "insert into likes values($pid, '$loginname');");

if ($like){

echo "You've liked this project."; echo "<meta http-equiv=\"refresh\" content=\"3; URL=http://127.0.0.1/Main/project.php?pid=$pid\">";}

else echo "Error".$follow."<br>".mysqli\_error($db);

### Sponsor a project

In each project page, only if this project is currently in funding (including not yet reaching minimal amount and already reaches minimum amount but has not reached maximum amount) and the user is not browsing the project she owns the pledge button is shown and able to be clicked by the user. When the pledge button is clicked, the browser jumps to another page that receives pledge amount and credit card information. PHP code for this part is shown below.

if(($loginname != $owner)&&($thisStatus != "FULL")&&($thisStatus != "FAIL")){

echo "<button onclick=\"location.href='http://127.0.0.1/Main/Pledge.php?pid=$pid'\">pledge</button>";}

In the sponsor information getting page, there is a form with two text fields for user to type amount to pledge, and if she wants to use another credit card. Besides SQL injection detection, these fields only allows numeric characters. Further, the user is also not allowed to sponsor more than the project needs. We use the following PHP that gets and stores the currently needed funding amount, if the number the user types exceeds it, this will also not be allowed.

$getMaxAmount = mysqli\_query($db,"select maxAmount-curAmount as T from project where pid=$pid");

$maxAmount = 0;

while ($pro = mysqli\_fetch\_array($getMaxAmount)) {

$maxAmount = $pro["T"];}

echo "<script>var maxAmount = $maxAmount; </script>";

We use the following JavaScript codes to check the input string:

var len = v.length;

for (var i=0;i<len;i++){

if(v.charAt(i)>"9"|| v.charAt(i)<"0"){

alert("Please enter a number!");

oField.value = "";

oField.focus();

oField.className="errInfo";

return false;} }

if (v>maxValue){

alert("Please enter a number within "+maxValue+"!");

oField.value = "";

oField.focus();

oField.className="errInfo";

return false;}

If the user does not provide valid amount, an alert window is jumped out with related cause, and the text field is emptied and focused.

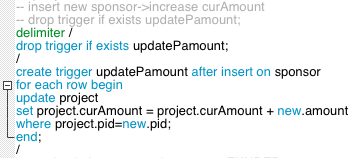
Once the form is submitted, our database will be changed, and there are two different cases resulting in doing different SQL queries. If the user has not pledged this project before, then a new tuple is inserted into the ‘Sponsor’ table, if the user is pledging the same project for a second time or even more, we do not insert another tuple but update the current tuple on the ‘amount’ attribute. Main PHP codes are:

$ifSponsored = mysqli\_query($db,"select \* from sponsor where pid = $pid and uname = '$loginname'");

if (mysqli\_num\_rows($ifSponsored)>0) $query2 = mysqli\_query($db,"update project set curAmount = curAmount + $amount where pid = $pid");

else $getStatus = mysqli\_query($db,"select status from project where pid = $pid");

Because money is added for the funding, this will cause the updating of ‘curAmount’ attribute in ‘Project’ table. The trigger ‘updatePamount’ can automatically update ‘curAmount’ in ‘Project’ table if a new tuple is inserted into ‘Sponsor’.



If instead a tuple of ‘Sponsor’ is updated, a trigger with similar function as ‘updatePamount’ would cause trouble because we can only increases ‘curAmount’ by the current ‘project.amount’, which is larger than the newly sponsored value. So for update on ‘Sponsor’, we just make a query:

$query2 = mysqli\_query($db,"update project set curAmount = curAmount + $amount where pid = $pid");

Also if its new ‘curAmount’ has reached ‘minAmount’, the status will be modified from ‘FUNDING’ to ‘FUNDED’, if it reaches ‘maxAmount’, the status will become ‘FULL’. We also defined a trigger to deal with this part:



Meanwhile when ‘curAmount’ of a project has reached ‘minAmount’, all sponsors are charged, which means all the tuples in ‘Sponsor’ table with ‘pid’ equal to the newly sponsored project have to be updated too, specifically, their ‘pledgeStatus’ attribute is changed from ‘WAIT’ to ‘CHARGED’. However, because we could not actually deal with credit card transaction in this project, changing ‘pledgeStatus’ is thus useless in this context, though we defined it.

In addition to the jobs above, a notification is created, and an existing notification will cause web application to show a message in the main page next time the owner log in.

mysqli\_query($db,"update project set curAmount = curAmount + $amount where pid = $pid");

Finally, the user is leaded back to the project page. And a pledge act is fully dealt with.

echo "<meta http-equiv=\"refresh\" content=\"3; URL=http://127.0.0.1/Main/project.php?pid=$pid\">";

### Comment on progress

Comment can be seen by every user, only leaving comment is only available to sponsors, and because project owners can generate several progresses, a sponsor is allowed to make a comment for every progress. The editable comment area is shown in the bottom of project page, if the user is a sponsor and hasn’t commented on the latest progress. PHP codes for it are:

$checkIfPledge = mysqli\_query($db,"select \* from sponsor where pid=$pid and uname = '$loginname'") or die(mysqli\_error());

$checkIfComment = mysqli\_query($db,"select \* from comment where pid=$pid and uname = '$loginname' and version = $version") or die(mysqli\_error());

if ((mysqli\_num\_rows($checkIfPledge)>0) && (mysqli\_num\_rows($checkIfComment)==0)){

echo "<div><form action='sendComment.php?version=$version&pid=$pid' method='POST' id = 'commentForm'><textarea style='margin-left:50px' name='description' rows=4' cols='50' placeholder='Leave a Command here:' form='commentForm' required></textarea><input style=\"margin-left:50px\" type=\"submit\" name=\"submit\" value=\"Submit\"></form></div>";}

While previous comments of all sponsors are shown above the text area and can be seen by everyone. We use the following PHP codes to fetch each comment, and show the reviewer information such as username, portrait. Then use simple HTML tags to list them and we will not put this part here.

$getComment = mysqli\_query($db,"select \* from comment where pid=$pid order by version DESC") or die(mysqli\_error());

while ($row = mysqli\_fetch\_array($getComment)) {

$commentuname = $row['uname'];

$userResult = mysqli\_query($db,"select uemail, protraitPath from user where uname = '$commentuname'") or die(mysqli\_error());

while ($e = mysqli\_fetch\_array($userResult)) {

$emailValue = $e["uemail"];

$protraitName = $e["protraitPath"];}

if($protraitName == ""){

$protraitName = "bg1.jpg";}

$protraitPath = "http://127.0.0.1/Images/".$protraitName;

When a comment is submitted, a new ‘Comment’ tuple is inserted. Meanwhile a notification is also created and inserted into the database, the process is similar to making a sponsor so we are not restating here.

### Prevent SQL injection

Guard against SQL injection is applied for every tag that supports inputting text, such as <textbox>, <textarea>, <input <text>>. We defined a JavaScript function to examine input stream, if it contains part of a SQL query, then it clears text field and pops out an alert. Key codes are:

re= /select|update|delete|exec|count|'|"|=|;|>|<|%/i;

if ( re.test(oField.value) ){

alert("Please do not type special characters or SQL query!");

oField.value = "";

oField.className="errInfo";

oField.focus();

return false;

### Create a project, create progress and registration

The function of creating a project or progress and registration are similar. We use <form> to upload necessary data and try to insert into our database.

<?php echo "<formenctype= \"multipart/form-data\"action= \"CreateProgressUpload.php?pid =$pid\" method= \"post\">";?>

<ul>

<li><label for="version">version</label>

<input type="text" name="version" placeholder="required" required></li>

<li><label for="description">description</label>

<input type="text" name="description"></li>

<li><label for="uploadedfile">Select Picture</label>

<input type="file" size="35" name="uploadedfile" /></li>

<li><input type="submit" name="submit" value="submit" /></li>

</ul></form>

$insertQ = "INSERT INTO `project` (`uname`, `startDate`, `endDate`, `minAmount`, `maxAmount`, `curAmount`, `pname`, `status`, `tags`, `imagePath`, `description`) VALUES

('$loginname', '$startDate', '$endDate', $minAmount, $maxAmount, $curAmount, '$pname', '$status', '$tags', '$protrait\_name', '$description');";

$insertNewProject = mysqli\_query($db,$insertQ) or die(mysqli\_error());

### Edit profile

This function used the data post by <form> and update our database. Key codes are:

<form enctype="multipart/form-data" action="EditProfileUpload.php" method="post">

<ul>

<li><label for="uploadedfile">Select Protrait</label>

<input type="file" size="35" name="uploadedfile" /></li>

<li><label for="email">Email</label>

<input type="text" name="email"></li>

<li><input type="submit" name="submit" value="submit" /></li></ul></form>

$updatePath = "update user set protraitPath = '$protrait\_name' where uname = '$loginname';";

$updateResult = mysqli\_query($db,$updatePath) or die(mysqli\_error());

### Search project

We settle the search function in main page. The search keywords will be matched with projects’ name or tags or descriptions. The result will be listed on search result page. Also we have the function that clicking tags buttons on a project page lead to the same result. Thus, we use two methods to pass the parameters to PHP file. The first one is using <form> submit keywords and second one pass keyword by URLs.

The <form> case: the URL remain as ‘../Main/SearchMain.php’. The tag button case: URL will looks like ‘../Main/SearchMain.php?keys=jazz’. Codes to achieve the search function are:

$searchKey = $\_POST['keys'];

$searchKey\_temp = $\_GET['keys'];

if($searchKey\_temp != ""){

$searchKey = $searchKey\_temp;}

$searchProject = "select project.pid, project.pname, status, tags, curAmount, minAmount, maxAmount, endDate , imagePath

from project

where project.tags like '%$searchKey%' or pname like '%$searchKey%'";

$allProject = mysqli\_query($db,$searchProject) or die(mysqli\_error());

<form name="search" action="SearchMain.php" method="post">

<input type="text" name="keys" onblur="AntiSqlInject(this)"

style="height: 20px;" placeholder="required" required>

<input type="submit" name="submit" value="Search"></form>

$tagList = explode(" ", $thisTags);

echo "<p>Tags: </p>

<ul class=\"row\" style=\"list-style-type:none\"><!--tag-->";

foreach ($tagList as $tag){

echo"<li class = \"col-md-1\" > < buttononclick = \"location.href = 'http://127.0.0.1/Main/SearchMain.php?keys = $tag' \ "> $tag</button></li>";}

echo "</ul>";

### Progress Bar

For the Progress Bar css we used from bootstrap:

<link rel= "stylesheet" href= "http://netdna.bootstrapcdn.com/bootstrap/3.1.0/css/bootstrap.min.css">

Progress Bar can provide a directive view about projects’ pledge information. We have three value can present pledge progress: current money gets from sponsor, minimum amount to start a project, maximum amount to stop pledge function. Thus, the progress will contain three part:

curAmount/MaxAmount; minAmount/maxAmount-curAmoun/maxAmount and remain.

$ratecur = (float)$curAmount/(float)$maxAmount\*100;

$ratemin = (float)$minAmount/(float)$maxAmount\*100 - $ratecur;

if($ratemin < 0){

$ratemin = 0;}

$ratemax = 100-$ratemin-$ratecur;

<div>

<div class=\"progress\" style=\"width:200px\">

<div class=\"progress-bar progress-bar-danger\" role=\"progressbar\" style = \" width: $ratecur % \"> </div>

<div class=\"progress-bar progress-bar-warning\" role=\"progressbar\" style = \" width : $ratemin%\"></div>

<div class=\"progress-bar progress-bar-success\" role=\"progressbar\" style =\"width: $ratemax%\"></div></div>

### Upload files

We have function that allow users customize their portrait by upload a local image to the server. We also allow a project owner upload files to server as description of the project. We use <form enctype="multipart/form-data"> to submit the basic information and upload the files selected. This function is used in Edit Profile and Create Project.

<form enctype="multipart/form-data" action="EditProfileUpload.php"

method="post"><ul>

<li><label for="uploadedfile">Select Protrait</label>

<input type="file" size="35" name="uploadedfile" /></li>

<li><label for="email">Email</label>

<input type="text" name="email"></li>

<li><input type="submit" name="submit" value="submit" /></li>

</ul></form>

$target\_path = "..\htdocs\Images\\";

$target\_path = $target\_path . basename( $\_FILES['uploadedfile']['name']);

$protrait\_name = $\_FILES['uploadedfile']['name'];

if(move\_uploaded\_file($\_FILES['uploadedfile']['tmp\_name'], $target\_path)) {

echo "The file ". basename( $\_FILES['uploadedfile']['name']).

" has been uploaded";

//update in data base

$updatePath = "update user set protraitPath = '$protrait\_name' where uname = '$loginname';";

$updateResult = mysqli\_query($db,$updatePath) or die(mysqli\_error());

if($newEmail != ""){

$updateEmail = "update user set uemail = '$newEmail' where uname = '$loginname';";

$updateResult = mysqli\_query($db,$updateEmail) or die(mysqli\_error());}

### Notification

The notification system used to record behavior of other users which behavior have effect related to one user. When this user login to Main page, these behaviors happened when user offline will be post to client thus user can know what happened related to him/her when he/she leaving.

The notification displays as a list and provide a button where can quick match to a detail page. We here listen the behavior comment and pledge and project status change. We create a new table `notification` on our database with no primary key. And once user has read these notification, we will drop the record.

require\_once("../connect.php");

$searchNoti = mysqli\_query($db,"select pid, description from notification where uname = '$loginname'") or die(mysqli\_error());

<?php

while ($row = mysqli\_fetch\_array($searchNoti)) {

$pid = $row["pid"];

$description = $row["description"];

$finePName = mysqli\_query($db, "select pname from project where pid = $pid") or die (mysqli\_error);

$pname = "project";

while ($name = mysqli\_fetch\_array($finePName)){

$pname = $name["pname"];}

echo "<div style=\"margin-top:10px\" class=\"row\">";

echo "<p class=\"col-md-5\">$description On $pname</p>";

//button to the project:

echo "<button class = \" col-md-2 \" onclick = \" location.href = 'http://127.0.0.1/Main/Project.php?pid=$pid' \">More</button>";

echo "</div>"; }

?>

# Use example

Here is a simple example of using this website. Someone visited the website, and saw the following login interface.

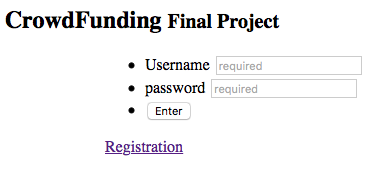


Figure 18. Login

She has no account, so she clicks the Registration link. Then she saw the following interface.

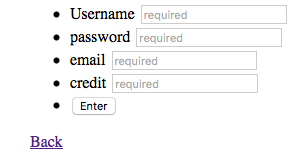


Figure 19. Registration

After typing basic information, she has created an account. Then she can see the main page with a bar in the front including ‘Projects’, her user name, and ‘Log out’, a search bar right below it, then several projects with photo (what shown in the picture is the default photo), name, current funding with a progress bar, status, type, due date.

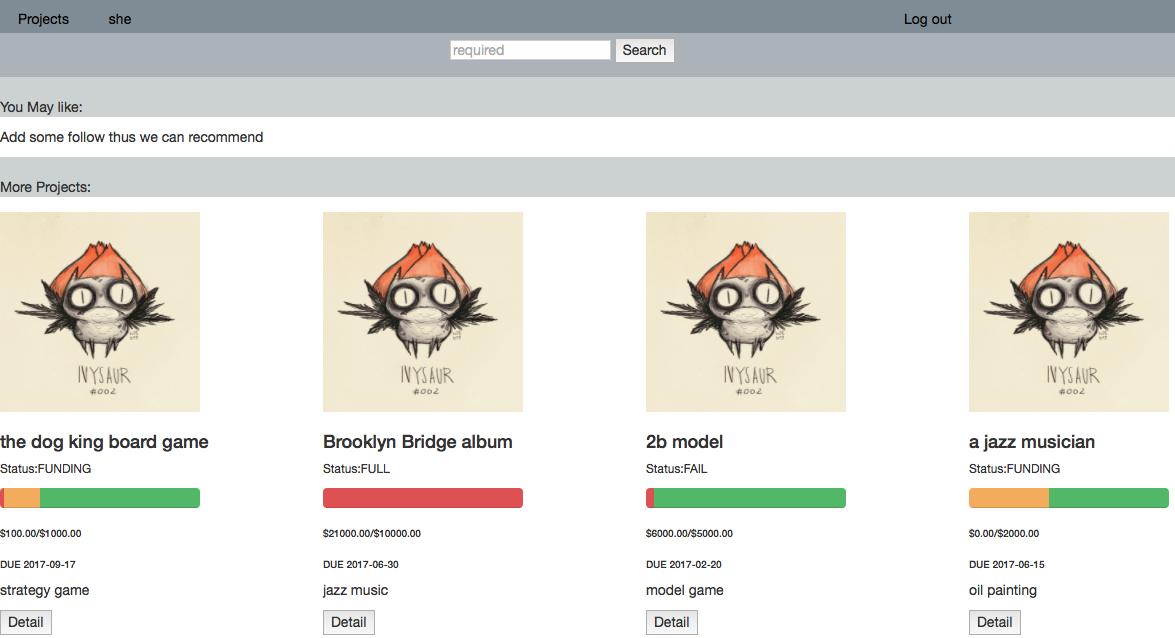


Figure 20. Main page

She was interested in the project “The golden cudgel”, so she clicked ‘Detail’ and entered the project page. This project is current in funding, she decided to follow the project owner ‘dua’, and like the project. After clicking responding buttons, the text on them has changed, meaning she has followed ‘dua’, and liked the project.

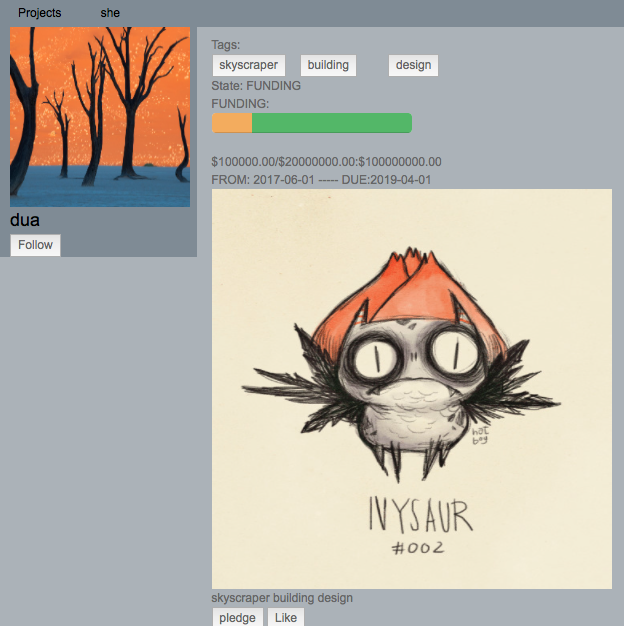
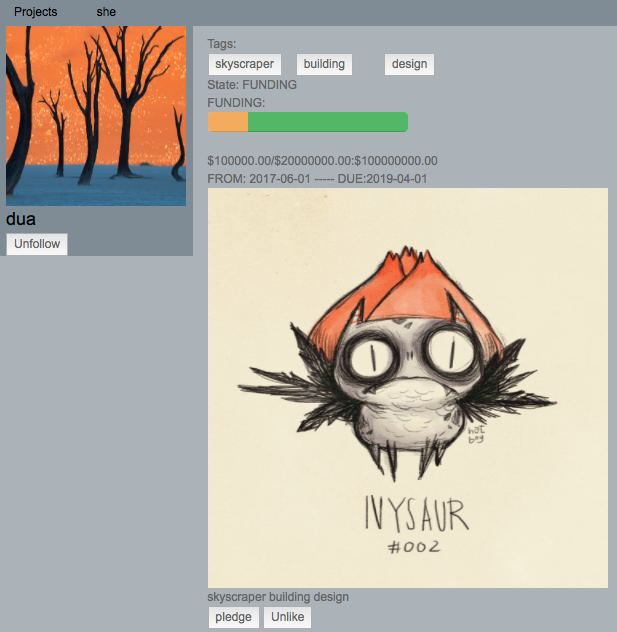
 

Figure 21. Project page and button change

Then she decided to sponsor the project and clicked the ‘pledge’ button. Now she saw the following interface. She decided to use the credit card submitted during registration, but mistakenly input 100,000,000. Without doubt, a messages shows and the value she inputted disappeared.

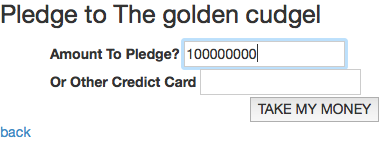
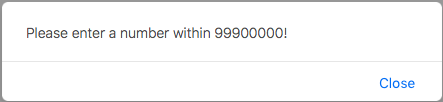
 

Figure 22. Pledge page and wrong input

Now she typed the correct number and the website jumped back to the project page. She found now she could make comment. She wrote something and clicked submit. Then she saw her comment.

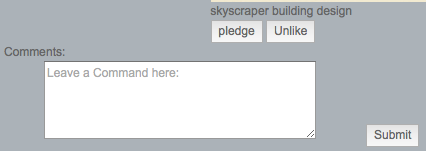
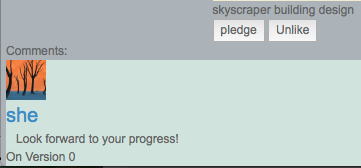
 

Figure 23. Comment

She clicked her user name in the front bar and went to her profile page, in this page she saw the project she liked and she sponsored.

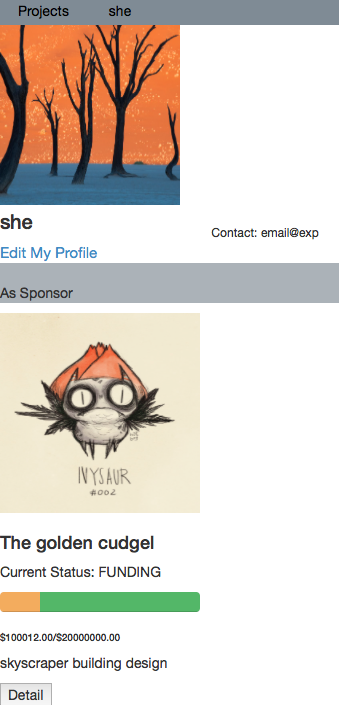
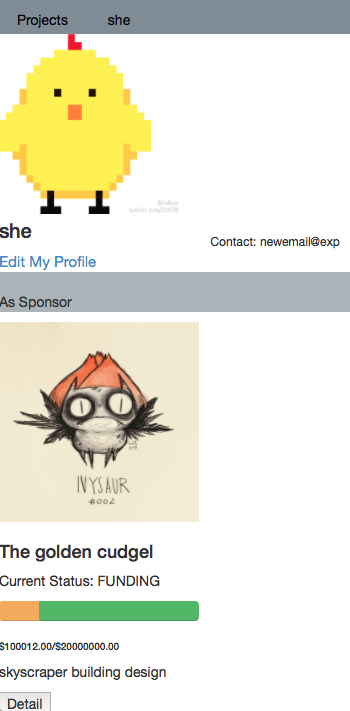
 

Figure 24. Profile before and after modification

She wanted to update her portrait, so she clicked ‘Edit My Profile’, and saw the following interface. She uploaded new portrait image and new address. Now her profile looks as the right picture above shows.

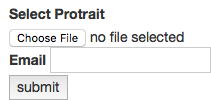


Figure 25. Profile before and after modification

She is done with the website today, so she clicked the log out link or simply closed the web browser.

The owner of project “The golden cudgel” dua logged in. He saw red “New Notification” text on the top bar.

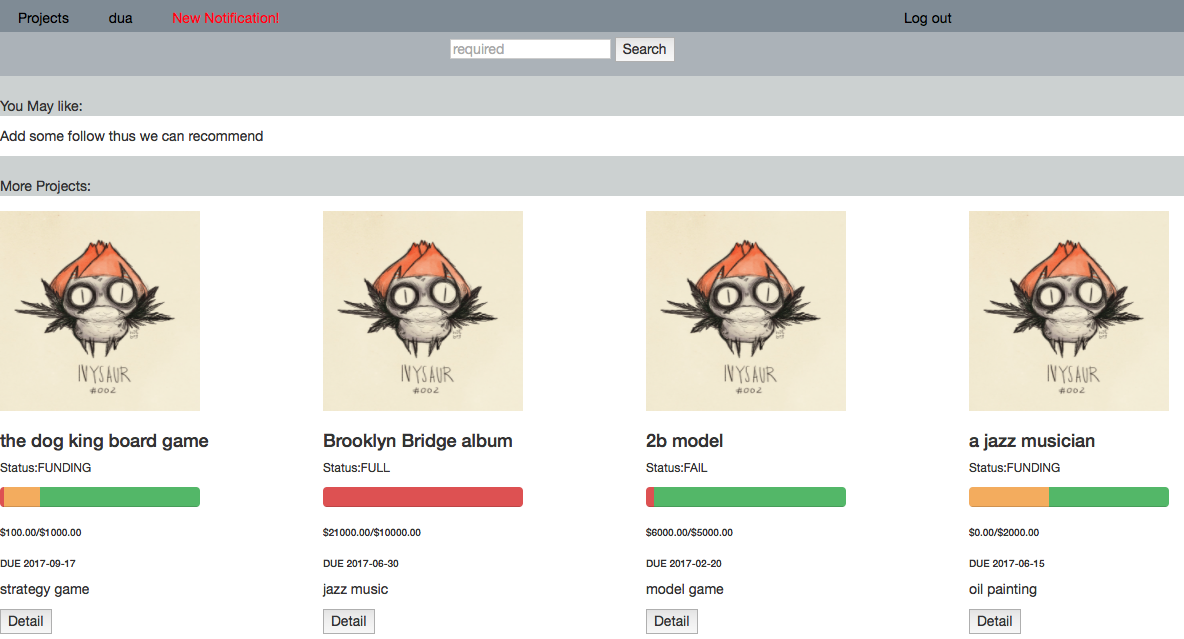


Figure 26. Main page with notification

Clicking the text and he went to the following page, showing all the actions related to his project. He was happy to see the user ‘she’ has sponsored his project “The golden cudgel”. Clicking the ‘More’ button and he went to the project page, and found project was still in funding.



Figure 27. Notification message

Then dua went to his another project “Brooklyn Bridge album” that is fully funded because he was going to publish a new progress. In the project page he could see a ‘update a progress’ tag, and he clicked it.

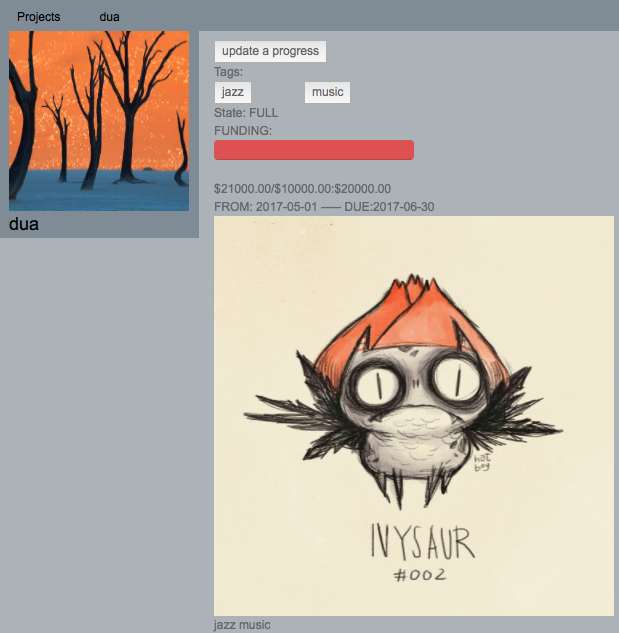


Figure 28. Project page with ‘update a progress’ button

In the interface below, he typed the new version number, description and uploaded new photos. Then he clicked ‘submit’. Now the new version is shown in the project page.

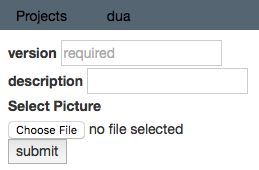
 

Figure 29. Upload progress and it is published in the project page

Also dua wanted to check the current status of the project “the dog king board game” he sponsored before, without going to his profile, he wanted to use the search bar. However, he mistakenly copied ‘delete’, then the following message popped out and the text field is cleared.

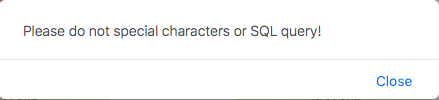


Figure 30. SQL injection alert

This time he typed ‘dog’, and the only result shown is exactly “the dog king board game”. He found its status had become ‘FUNDED’, meaning he could leave a rating.

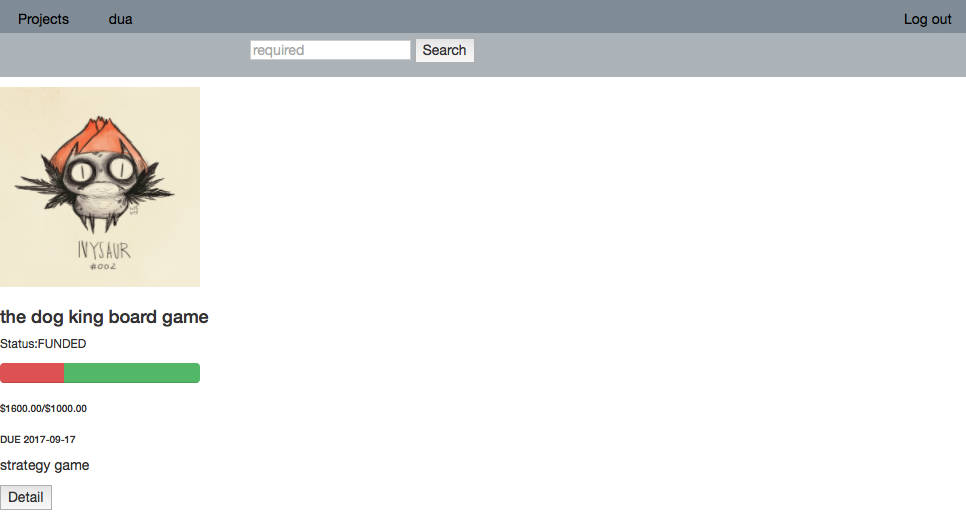


Figure 31. Search result

In the project page he saw a star rating label. By moving mouse, the number of lightened star changes.

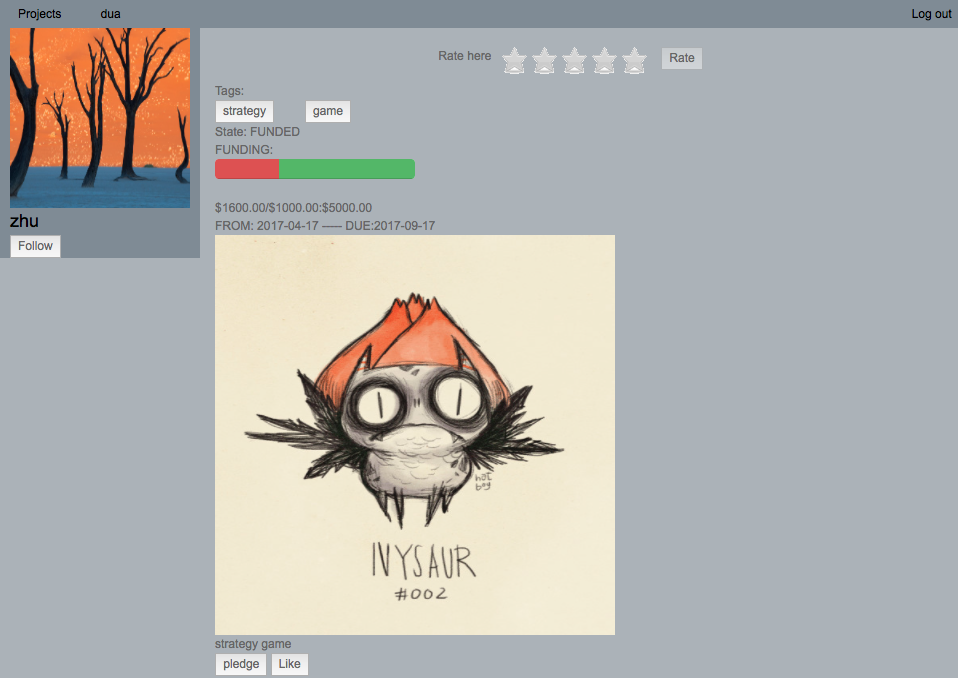
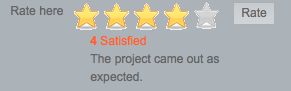
 

Figure 32. Project page with rating

After viewing the current progress, he chose 4 stars and clicked ‘Rate’. Then the overall rating of this project has been modified too.

Now dua found a new project “a jazz musician” that hasn’t received any fund, he was interested in it and decided to pledge 100$. The following shows the change of project funding progress bar before and after the pledge.

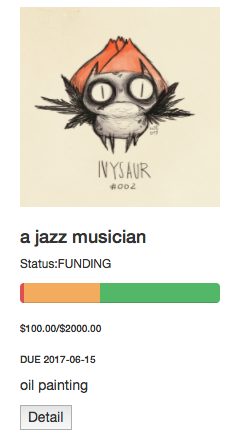
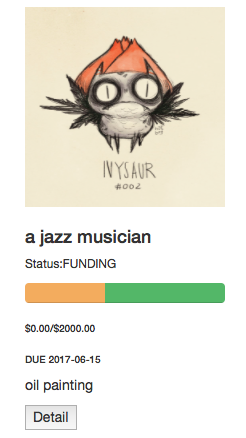


Figure 33. Change of money and funding progress bar after pledge

# Source

We uploaded the source code to develop the website, SQL queries to build the database, report and other resources in GitHub. The URL is:

*https://github.com/Dwan9/DatabaseCrowdFunding.git*