# CS2102

# Group 4 Project Report

# Crowdfunding Website

# *SteFund* ©


**Group Members:**

Leonard Hio (A0127393B)
Huang Feixue (A0129538W)
Chng Hui Yie (A0127357B)
Ng Ee Shuan (A0125255L)
Aylward Tan (A0124274L)

# Table of Contents

# I. Introduction

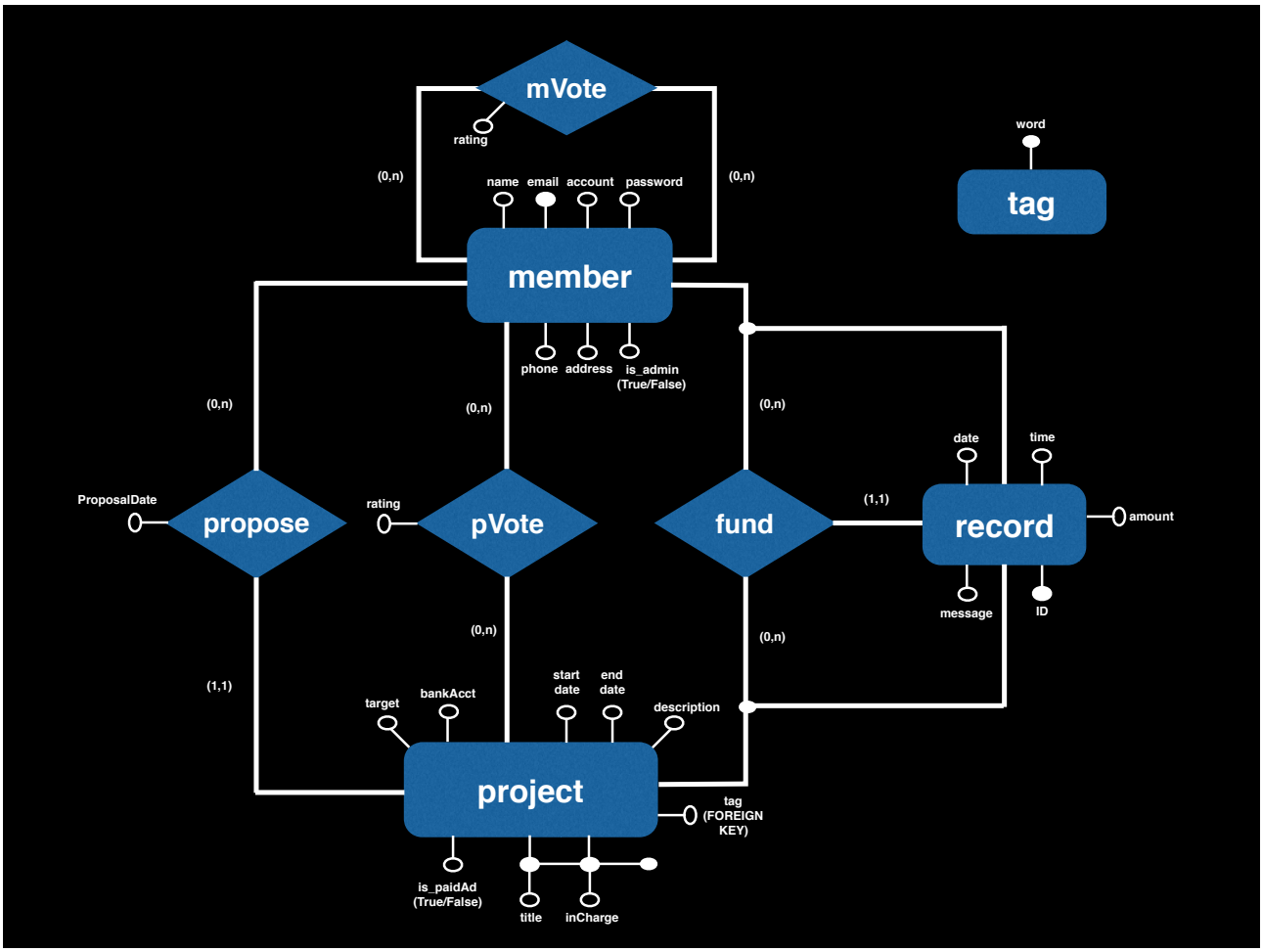
In recent years, Singapore's startup scene has grown at an unprecedented rate. More incubators and accelerators have been popping up across our Little Red Dot, making our nation's startup ecosystem more vibrant than ever.

Therefore, we have chosen to embark on a project to create a crowdfunding website. In doing so, we hope to create a platform for aspiring entrepreneurs to raise money from the masses to turn their ideas into reality. Our website, SteFund, aims to be a one-stop shop for entrepreneurs to propose new projects, individuals or businesses to browse, support and fund others' projects, users to vote on their favourite projects, and so on.

Our group has chosen to use the Zone server and Oracle database management system for our crowdfunding website. In addition, we have chosen to use PHP as our server side language. On the front end, we used a Bootstrap framework together with jQuery in order to create a dynamic and responsive webpage for a more pleasurable experience.

# II. ER diagram

The following is the Entity-Relationship diagram of our website database.



# III. Relational Schema

The member table keeps a record of all users who registers and logs in when using SteFund. This includes regular users who browse and support projects, entrepreneurs who propose projects, and administrators of the website. The various information recorded are the user's email, name, address, log-in password, bank account number for monetary transactions and phone number. An additional attribute is used to indicated whether this user is an admin (with either 1 being YES and 0 being NO). Users are identified by their emails.

```
CREATE TABLE member(
	email VARCHAR(128) PRIMARY KEY,
	name VARCHAR(255) NOT NULL,
	address VARCHAR(255) NOT NULL,
	password VARCHAR(16) NOT NULL,
	is_admin INT NOT NULL CHECK(is_admin=1 OR is_admin=0),
	acct VARCHAR(64) NOT NULL,
	phone VARCHAR(16) DEFAULT 'not given' NOT NULL
);
```

The tag table keeps track of tags (keywords and labels for searching projects) that appear in the tag cloud on our website home page. Tags are simply identified by the word itself.

```
CREATE TABLE tag (
	word VARCHAR(32) PRIMARY KEY
);
```

The proposed_project table keeps track of projects that have been proposed and posted onto the SteFund platform. One of the interesting attributes of this table is the is_paidAd attribute. This allows us to keep track of "sponsored projects", meaning that the individual or company has paid money for SteFund to actively advertise their particular projects on our website to garner more attention from potential donors.

Other attributes recorded in this table are the title, in-charge (either an individual or a corporation/company), start date, end state, proposal_date, description, proposer, targeted-amount, tag, and a bank account number for transferring fundings of a project. A project is typically identified by its title and in-charge as a key. This indicates that each individual or company can only propose at most 1 project of the same title.

```
CREATE TABLE proposed_project (
        title VARCHAR(64),
        in_charge VARCHAR(64),
        start_date DATE DEFAULT SYSDATE NOT NULL,
        end_date DATE DEFAULT '30-12-9999' NOT NULL,
        proposal_date DATE NOT NULL,
        description VARCHAR(512),
        proposer VARCHAR(128) REFERENCES member(email) ON DELETE
            CASCADE,
        target NUMBER(*, 2) NOT NULL CHECK(target > 0),
        tag VARCHAR(32) REFERENCES tag(word) ON DELETE CASCADE,
        bank_acct VARCHAR(32) NOT NULL,
        is_paidAd INT NOT NULL CHECK (is_paidAd = 0 OR is_paidAd =
            1),
        PRIMARY KEY (title, in_charge),
        CHECK (proposal_date <= start_date AND end_date >=
            start_date)
);
```

The fund_record table keeps track of all monetary transactions from donors to projects. This ensures that every user's donations is properly captured and the money will be channeled to the correct project. A user is able to make multiple donations to the same project. The information recorded in this table are the transaction data and time, amount donates, transaction ID, donor's identity, project's title and project's in-charge. A record in this table can be identified by the transaction ID.

```
CREATE TABLE fund_record (
        fund_date_time TIMESTAMP DEFAULT LOCALTIMESTAMP NOT NULL,
        amount NUMBER(*,2) NOT NULL,
        message VARCHAR(256),
        id INT PRIMARY KEY,
        donor VARCHAR(128),
        p_title VARCHAR(64),
        p_in_charge VARCHAR(64),
        FOREIGN KEY (donor) REFERENCES member(email) ON DELETE
            CASCADE,
        FOREIGN KEY (p_title, p_in_charge) REFERENCES
            proposed_project (title, in_charge) ON DELETE CASCADE
);
```

The p_vote table keeps track of the review ratings that users award to projects. Users are allowed to submit ratings from 0 to 5 (5 being the best) for each project only once. The votes for each project are collated and the average is displayed on the project information page. The top 10 rated projects are also listed on the homepage leaderboard. Each p_vote record is identified by the combination of voter's email, project title and project in-charge.

```
CREATE TABLE p_vote (
       rating NUMBER(3,2) CHECK (rating >=0.00 AND rating <= 5.00),
       voter VARCHAR(128),
       p_title VARCHAR(64),
       p_in_charge VARCHAR(64),
       PRIMARY KEY (voter, p_title, p_in_charge),
       FOREIGN KEY (voter) REFERENCES member(email) ON DELETE
          CASCADE,
       FOREIGN KEY (p_title, p_in_charge) REFERENCES
          proposed_project (title, in_charge) ON DELETE CASCADE
);
```

The m_vote table keeps track of the ratings that users award to other members. Users are allowed to submit ratings from 0 to 5 (5 being the best) for each member only once. Users are not allowed to vote for themselves. The votes for each member are collated and the average is displayed on the member's profile page. The top 10 SteFund Member (entrepreneurs) are also listed on the homepage leaderboard. Each m_vote record is identified by the voter and voter's emails as a pair.

```
CREATE TABLE m_vote (
rating NUMBER(3,2) CHECK (rating >=0.00 AND rating <= 5.00),
voter VARCHAR(128),
votee VARCHAR(128),
PRIMARY KEY (voter, votee),
FOREIGN KEY (voter) REFERENCES member(email) ON DELETE CASCADE,
FOREIGN KEY (votee) REFERENCES member(email) ON DELETE CASCADE,
CHECK (voter <> votee)
);
```

# IV. Sample SQL code

In our project, the following SQL queries were used:

## 1. Basic Search for Users

a. For basic search, we search the members table for instances where the keyword is in either the email attribute or the name attribute:

```
$username = $_POST["username"];
$sql = "SELECT * FROM member WHERE email LIKE '%".$username."%' OR name LIKE '%".$username."%'";
```

## 2. Advanced Search for Users

a. For advanced search, we allow users to search for members based on their email address, name, min/max rating and whether they want to search for members who have uploaded projects only, and if so, how many projects they've uploaded.

```
if ($minRating >= 1) {
    $sql = "SELECT *
FROM member m
WHERE (m.email LIKE '%" . $email . "%' AND m.name LIKE '%" . $name . "%')
AND EXISTS (
    SELECT v.votee
    FROM m_vote v
    WHERE v.votee = m.email
    GROUP BY v.votee
    HAVING avg(v.rating)>= " . $minRating . " AND avg(v.rating)<= " . $maxRating . ")";
} else {
    $sql = "SELECT *
FROM member m
WHERE (m.email LIKE '%" . $email . "%' AND m.name LIKE '%" . $name . "%')
AND (EXISTS (
    SELECT v.votee
    FROM m_vote v
    WHERE v.votee = m.email
    GROUP BY v.votee
    HAVING avg(v.rating)>= " . $minRating . " AND avg(v.rating)<= " . $maxRating . ")
OR NOT EXISTS (
    SELECT *
    FROM m_vote vo
    WHERE vo.votee = m.email))";
}

if ($hasProject) {
    $sql = $sql.
        "AND EXISTS (
        SELECT p.proposer
        FROM proposed_project p
        WHERE p.proposer = m.email
        GROUP BY p.proposer
        HAVING COUNT(*) =".$numProjects.")";
}
```

b. Since the rating of each member is not stored in the member table, we have to calculate the average rating of each member based on the m_vote table. We then check if the email of a certain member EXISTS in the m_vote table with AVG(rating) above the min rating and below the max rating requested. Since the emails of members of rating 0 will not appear in the m_vote table, we have to make sure to include a NOT EXISTS clause in our SQL query if the user requests for users with a min rating of 0.

c. Lastly, If the user specifies that he wishes to search for members who have uploaded some projects only, we append to the existing SQL query one more EXISTS clause which ensures only members who proposed a certain number of projects are returned.

### 3. Basic Search for Projects

a. We only allow users to search based on keywords of project names in the proposed_project table in the basic search:

```
if ($isFromTagCloud) {
    $tag = $_POST['tagFromTagCloud'];
    $sql = "SELECT * FROM proposed_project WHERE tag = '".$tag."'";
} else {
    $sql = "SELECT * FROM proposed_project WHERE title LIKE '%" . $projectTitle . "%'";
}
```

b. The exception is when the user clicks on a tag in the tag cloud on the homepage. In that case, the SQL query returns all the projects with that tag.

## 4. Advanced Search for Projects

a. For the advanced search, users can search for projects based on the title, the organisation/proposer, start/end date, rating, tag, and whether to return unfulfilled projects only.

```
if ($minRating >= 1) {
    $sql = "SELECT *
FROM proposed_project p
WHERE (p.title LIKE '%" . $title . "%' AND (p.in_charge LIKE '%" . $by. "%' OR p.proposer LIKE '%" . $by. "%'))
AND (p.start_date >= TO_DATE('".$sDate."', 'YYYY-MM-DD') AND p.end_date <= TO_DATE('".$eDate."', 'YYYY-MM-DD'))
AND p.tag = '".$tag."'
AND (EXISTS (
    SELECT v.p_title, v.p_in_charge
    FROM p_vote v
    WHERE v.p_title = p.title
    AND v.p_in_charge = p.in_charge
    GROUP BY v.p_title, v.p_in_charge
    HAVING avg(v.rating)>= " . $minRating . " AND avg(v.rating)<= " . $maxRating . "))";
} else {
    $sql = "SELECT *
FROM proposed_project p
WHERE (p.title LIKE '%" . $title . "%' AND (p.in_charge LIKE '%" . $by. "%' OR p.proposer LIKE '%" . $by. "%'))
AND (p.start_date >= TO_DATE('".$sDate."', 'YYYY-MM-DD') AND p.end_date <= TO_DATE('".$eDate."', 'YYYY-MM-DD'))
AND p.tag = '".$tag."'
AND (EXISTS (
    SELECT v.p_title, v.p_in_charge
    FROM p_vote v
    WHERE v.p_title = p.title
    AND v.p_in_charge = p.in_charge
    GROUP BY v.p_title, v.p_in_charge
    HAVING avg(v.rating)>= " . $minRating . " AND avg(v.rating)<= " . $maxRating . ")
OR NOT EXISTS (
    SELECT *
    FROM p_vote vo
    WHERE vo.p_title= p.title
    AND vo.p_in_charge = p.in_charge))";
}

if ($wantsUnfinished) {
    $sql = $sql.
        "AND(
            (EXISTS(
                SELECT *
                FROM fund_record f
                WHERE f.p_title = p.title
                AND f.p_in_charge = p.in_charge)
            AND p.target > (
                SELECT sum(f2.amount)
                FROM fund_record f2
                WHERE f2.p_title = p.title
                AND f2.p_in_charge = p.in_charge)
        OR NOT EXISTS (
        SELECT *
        FROM fund_record fu
        WHERE fu.p_title = p.title
        AND fu.p_in_charge = p.in_charge)))";
}
```

b. Similar to the advanced search for users, we first construct our query based on the min rating specified, making sure that we include projects with rating 0 if the user requested for them. If the user wishes to search for only unfulfilled projects, we append to the SQL query an additional clause searching for projects with raised amount lesser than the target amount as well as for projects with raised amount of $0.

## 5. Get Raised Amount for Projects

a. Getting the raised amount for projects (to be shown in the project page, in the tag cloud, etc.) is a simple SQL query involving one aggregate function:

```
$sql = "SELECT sum(amount) FROM fund_record WHERE p_title = '".$title."' AND p_in_charge = '".$inCharge."'";
```

## 6. Get Ratings for Users/Projects

```
$sql = "SELECT to_char(avg(rating), '0.99'), count(rating) FROM m_vote WHERE votee='".$email."' GROUP BY votee";
```

```
$sql = "SELECT to_char(avg(rating), '0.99'), count(rating) FROM p_vote WHERE p_title ='".$title."' AND p_in_charge='".$inCharge."' GROUP BY p_title, p_in_charge";
```

## 7. Views for Top 10 Lists

a. We constructed a view for two of our top 10 tables located in the home page. This is to allow for dynamic updates of the top 10 tables without excessive recomputation every time we access the homepage.

```
CREATE VIEW top_projects AS
(SELECT v.p_title AS title, v.p_in_charge AS in_charge, avg(v.rating) AS rating
FROM p_vote v
GROUP BY v.p_title, v.p_in_charge)
UNION
(SELECT p.title AS title, p.in_charge AS in_charge, 0.00 AS rating
FROM proposed_project p
WHERE (p.title, p.in_charge) NOT IN (
  SELECT vo.p_title, vo.p_in_charge
  FROM p_vote vo)
);

CREATE VIEW top_users AS
(SELECT v.votee AS email, avg(v.rating) AS rating
FROM m_vote v
GROUP BY v.votee)
UNION
(SELECT m.email AS email, 0.00 AS rating
FROM member m
WHERE m.email NOT IN (
  SELECT vo.votee
  FROM m_vote vo)
);
```

## 8. Registration of Users

a. Registration is a simple INSERT command:

```
$email = $_POST['email'];
$password = $_POST['password'];
$name = $_POST['name'];
$address = $_POST['address'];
$acct = $_POST['acct'];
$phone = $_POST['phone'];
$isAdmin = 0;
$sql = "INSERT INTO member VALUES ('".$email."', '".$name."', '".$address."', '".$password."', ".$isAdmin.", '".$acct."', '".$phone."')";
```

## 9. Submission of Projects

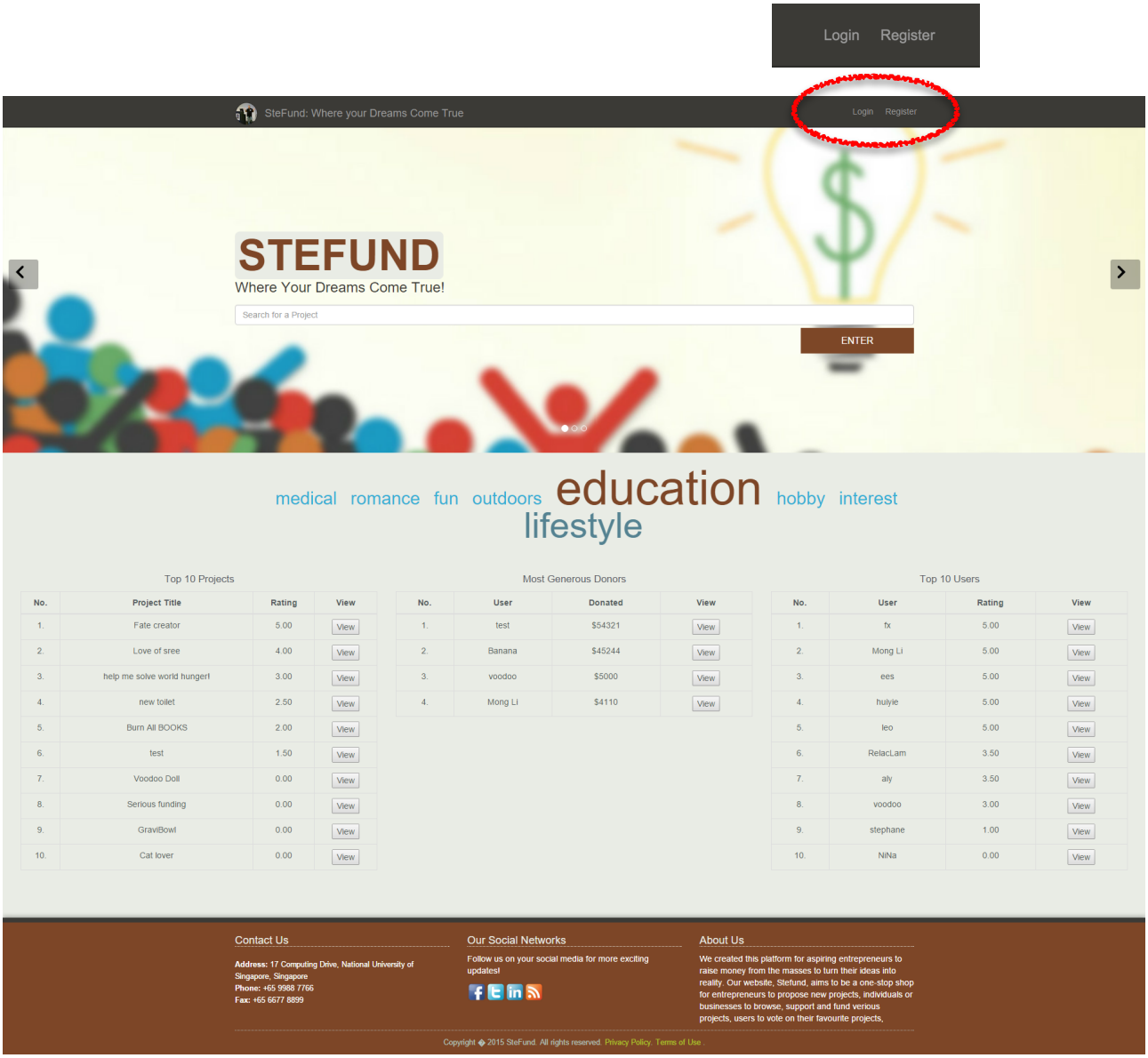
a. Submission is also a simple INSERT command:

```
$title = $_POST['title'];
$inCharge = $_POST['inCharge'];
$startDate = $_POST['startDate'];
$endDate = $_POST['endDate'];
$target = $_POST['target'];
$bankAcct = $_POST['bankAcct'];
$description = $_POST['description'];
$tag = $_POST['tag'];
$proposer = $_POST['proposer'];
$propDate = $_POST['propDate'];
$sql = "INSERT INTO proposed_project VALUES ('".$title."', '".$inCharge."', TO_DATE('".$startDate."', 'YYYY-MM-DD'), TO_DATE('".$endDate."', 'YYYY-MM-DD'),
        TO_DATE('".$propDate."', 'DD/MON/YYYY'), '".$description."', '".$proposer."', ".$target.", '".$tag."', '".$bankAcct."', 0)";
```

# V. Webpage Screenshots

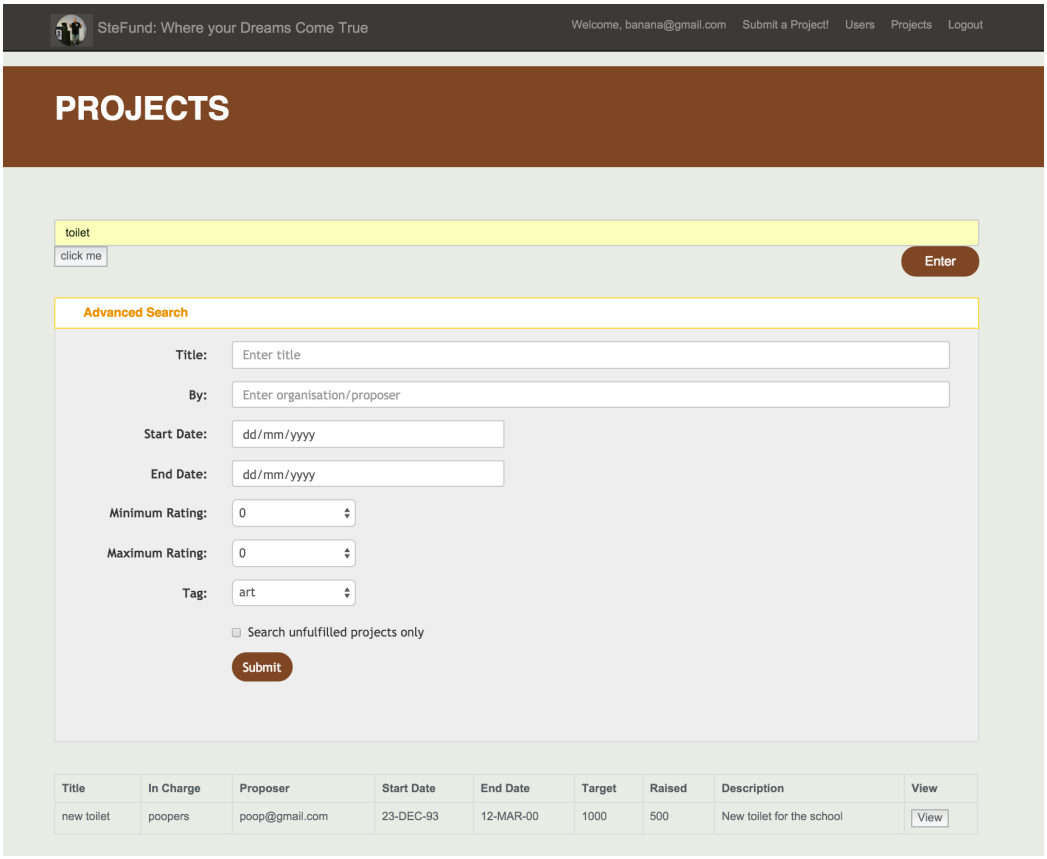
## 1) Home Page (Quick-Search, Tag Cloud, Leaderboards)

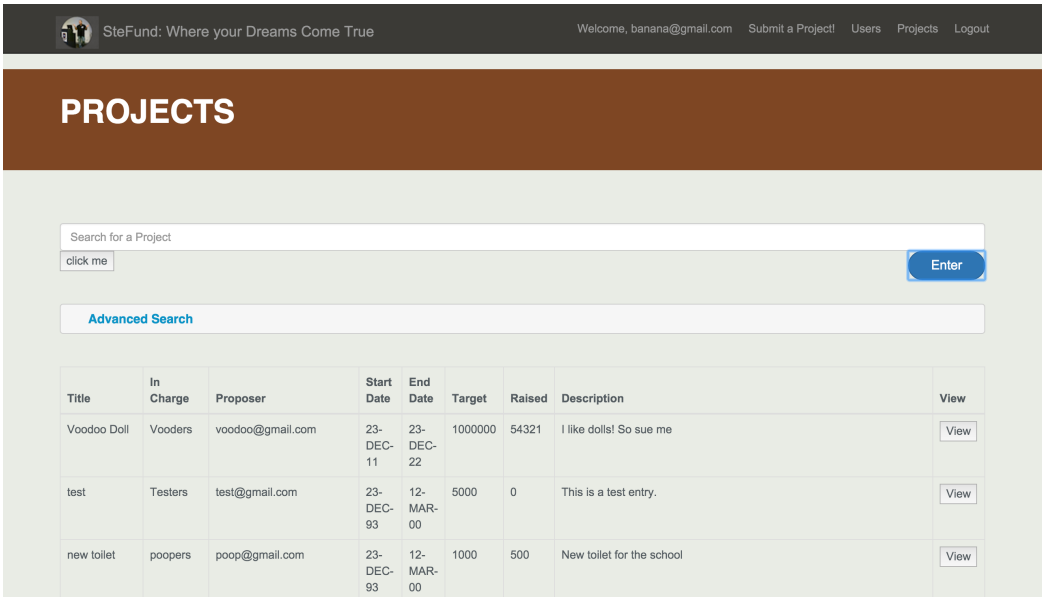
The following image showcases the homepage of our website, which features a Quick-Search bar for projects, Tag Cloud for project categories and 3 Leaderboards ranking the top 10 projects, top 10 entrepreneurs (users) and top 10 donors.



## 2) Project List

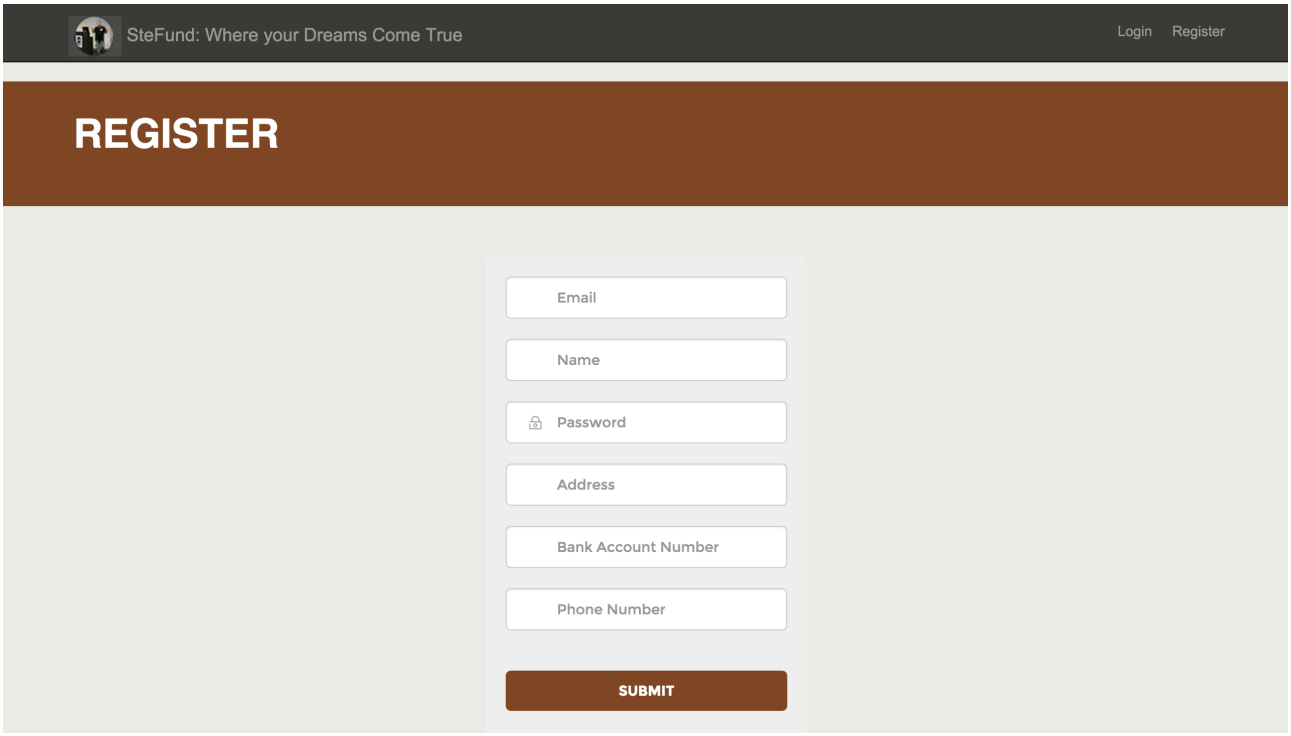
Keying in a keyword and clicking ‘ENTER’ allows the website to generate a list of projects relevant to the context. Users may also make use of the ‘Advanced Search’ feature to filter the listed results.

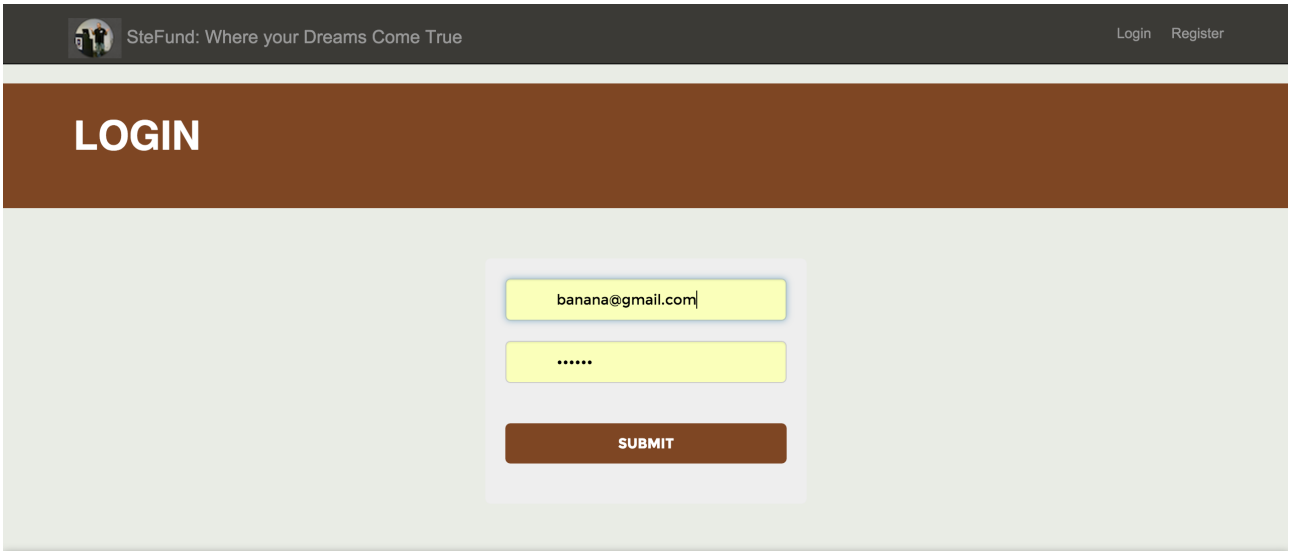


### 3) Login Page, Registration Page

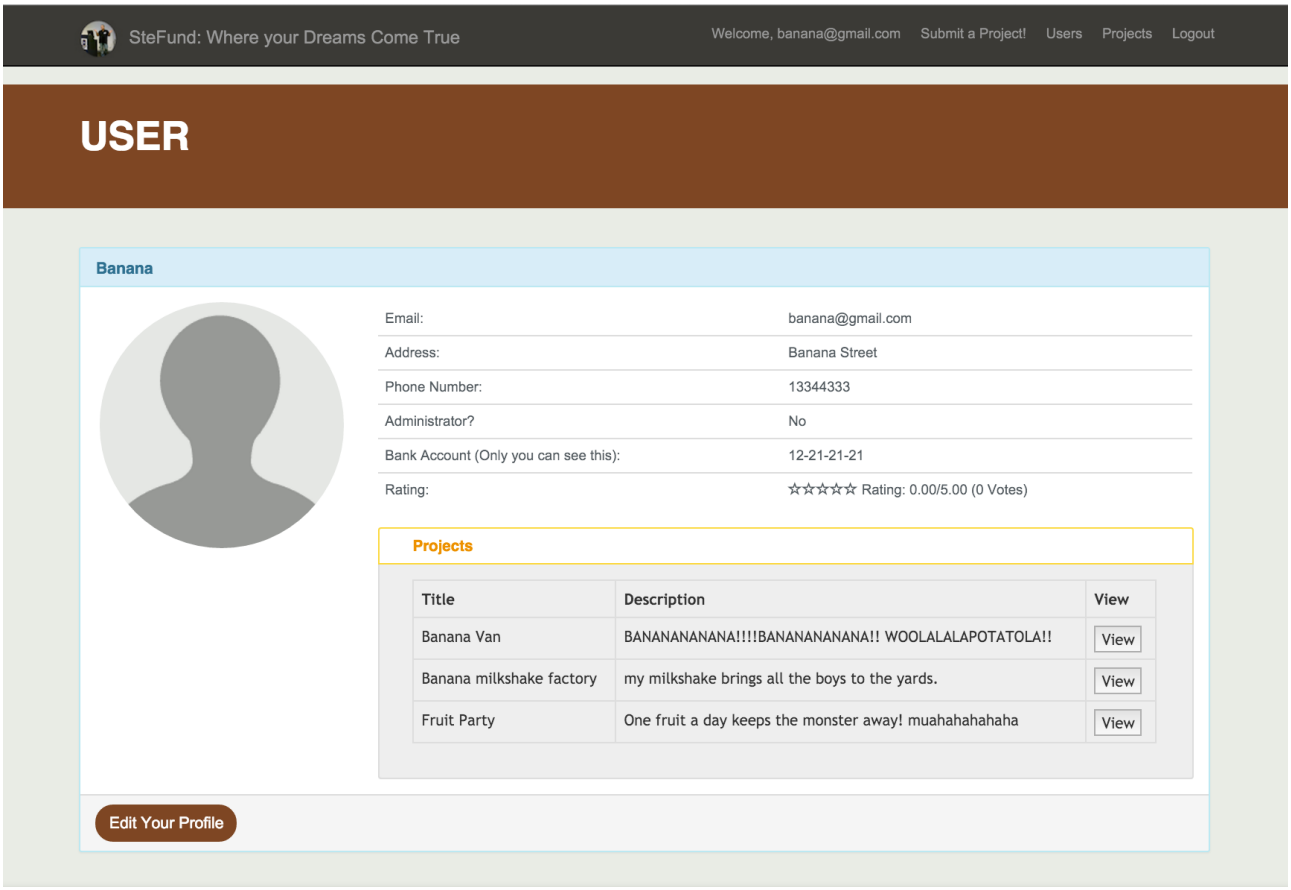
Users are able to register for an account on SteFund and login to access member-only content and services.

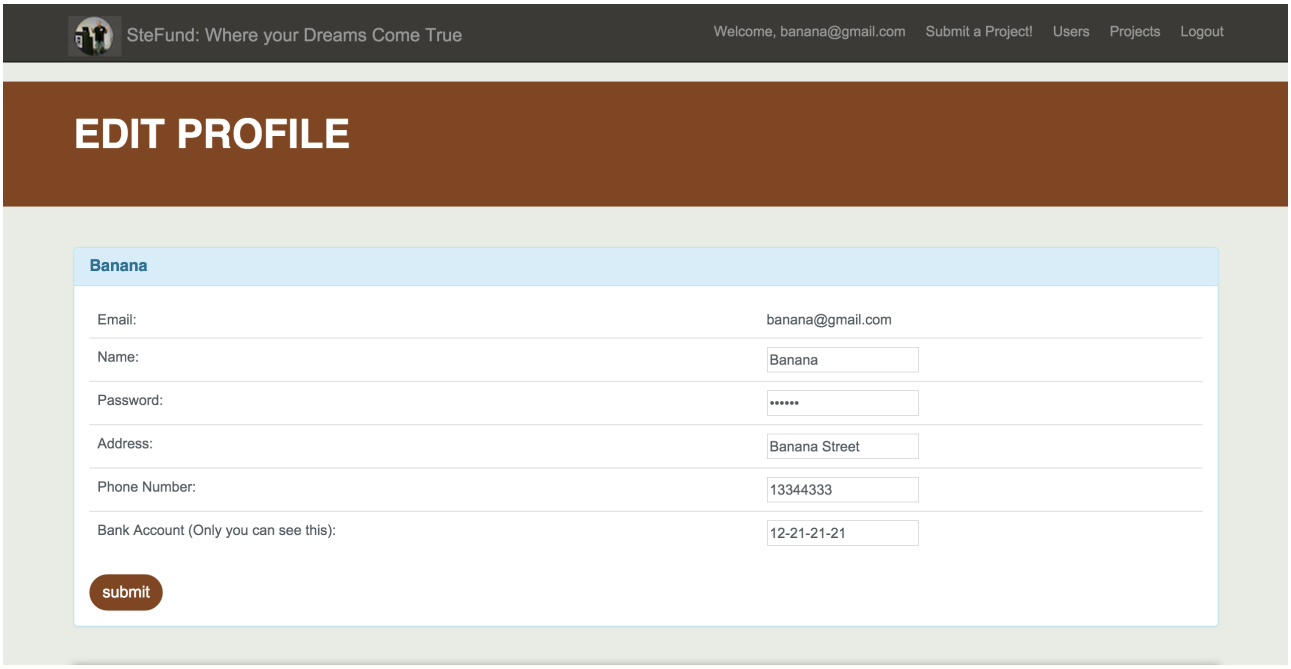


## 4) Profile, Edit Profile

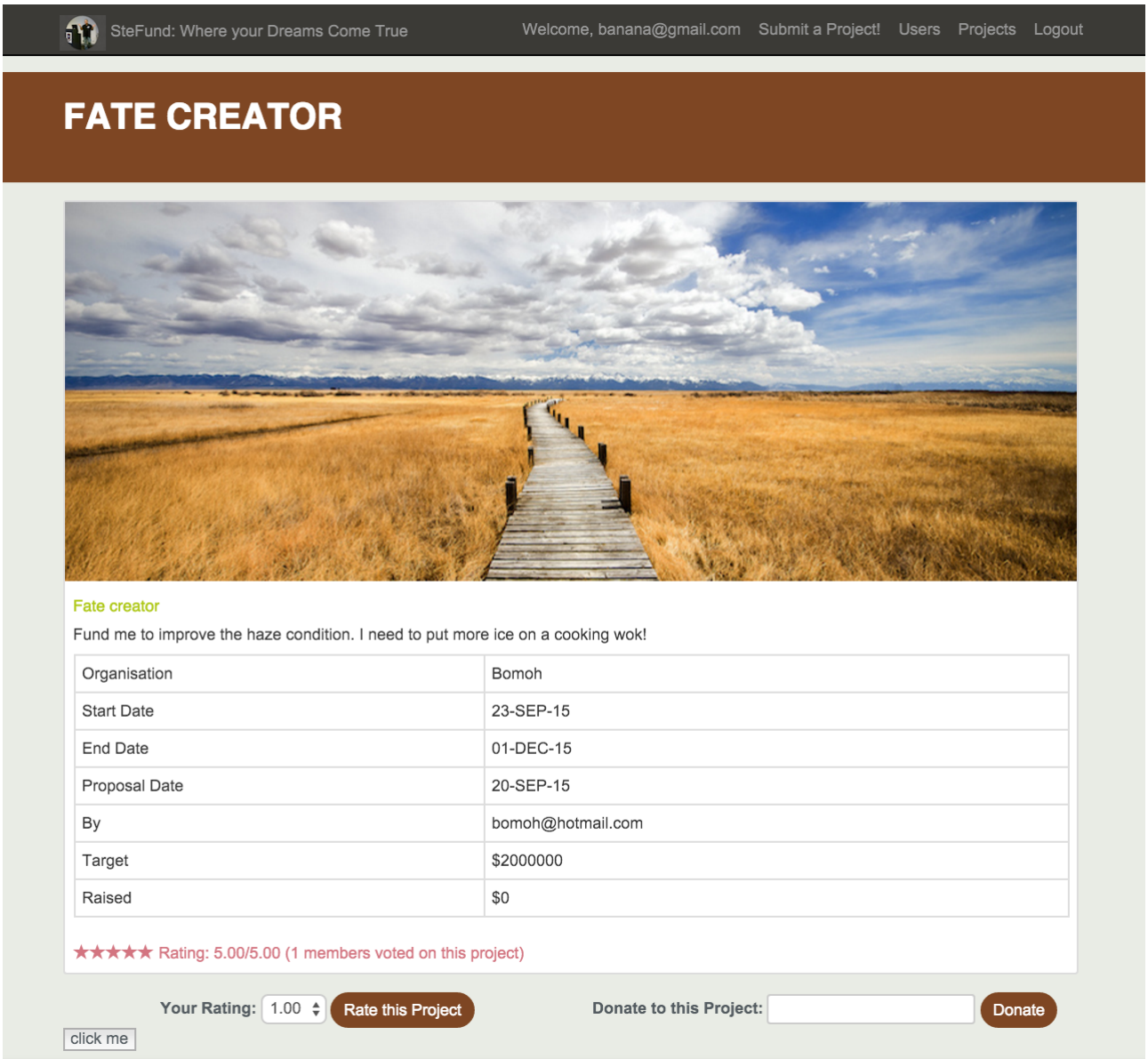
Each member will have a profile page displaying their individual information. They may update and edit the information on their profile. Content on the profile pages can be viewed by any other SteFund members.



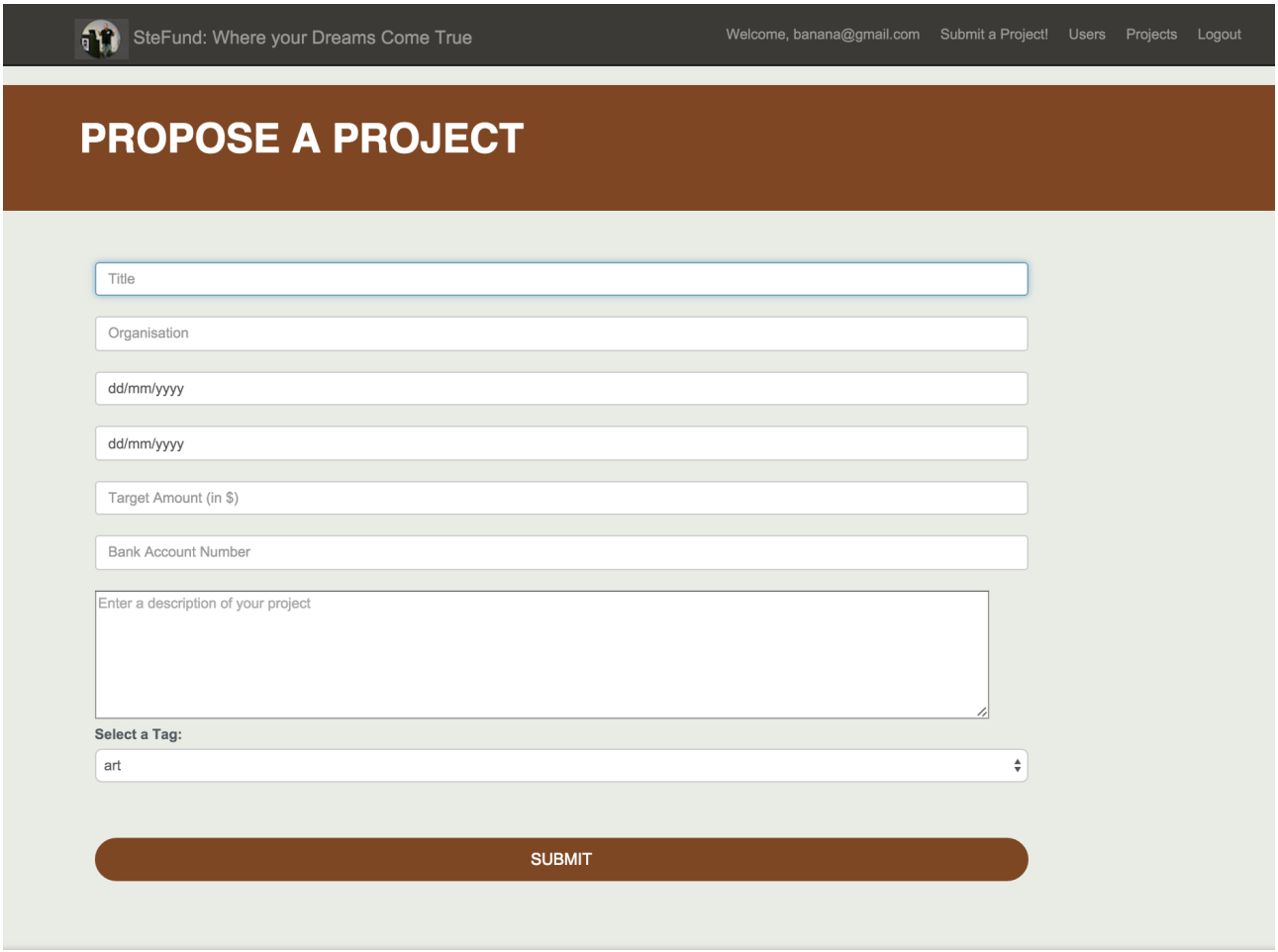
## 5) Project Information Page

Just as how each member has their own profile page, each project also has a information page, where essential information such as project description, target amount, ratings etc. are shown.



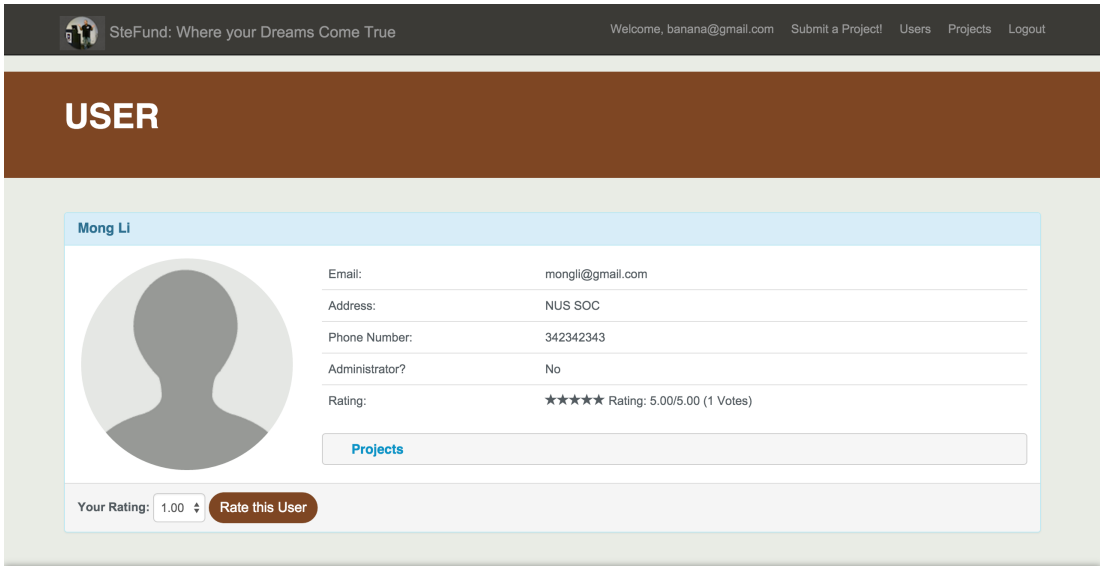
## 6) Project Submission Page

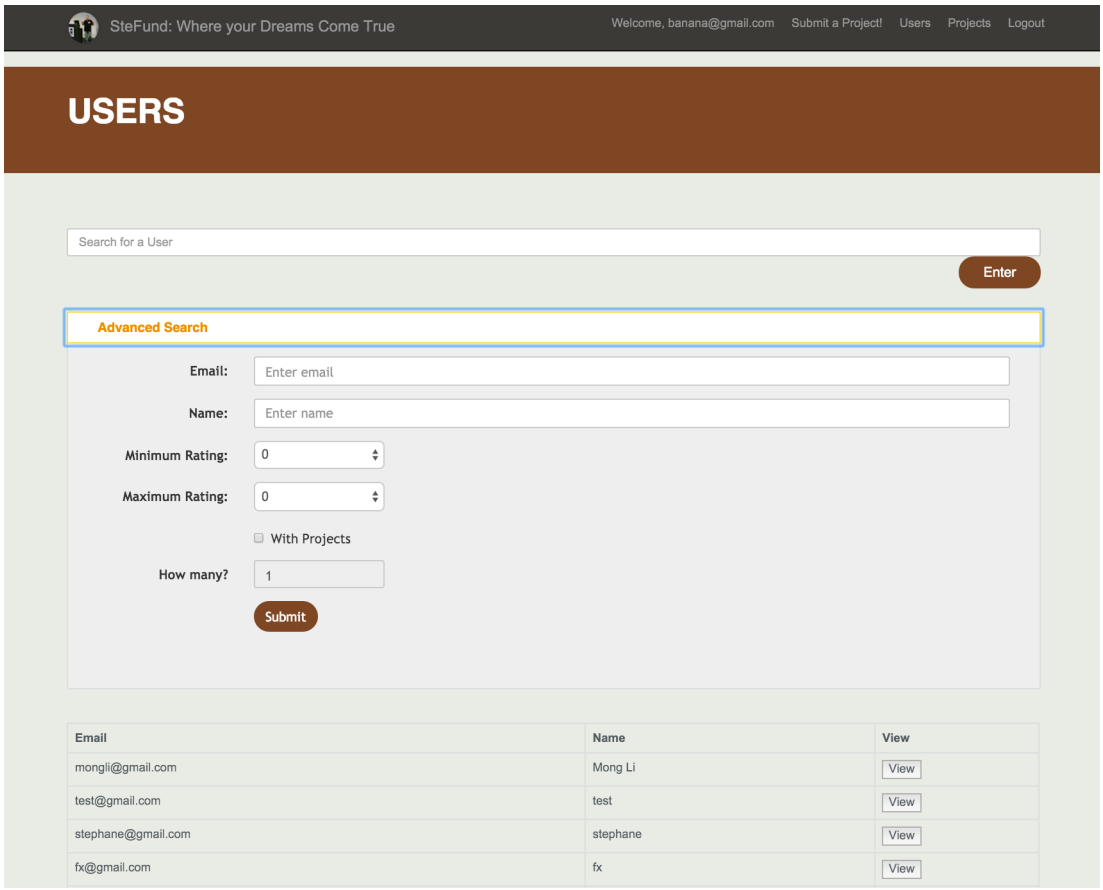
Members of SteFund may access the feature of creating new projects. This webpage displays a form to be filled in for a newly initiated project.



## 7) User Information Page, User List

Members of SteFund has access to the list of other members. Similar to the project list, the user list also has an ‘Advanced Search’ option to further filter the results. Members may click to view another member’s profile page, as known as a User Information Page.



# VI. Conclusion

Our group hopes to continue working on our website even after the conclusion of this project. We hope to explore and add functionalities that will enable our website to be operational; for example, integration with online payment systems such as Credit Cards, Paypal and Google Wallet. In addition, we are also aiming to improve the user interface and design of our website.

If we are successful in launching our website in the future, it will certainty make a real difference to our community, especially for startups or even social causes which are looking for a platform to raise awareness and funds for their projects. Nonetheless, whether or not we are eventually successful, we believe that the idea-generation and programming skills we have acquired throughout the course of this project is invaluable and will be greatly useful for future projects that we embark on.