Group 6: WeWantA Inc.

Coding Turk System
Phase II: Design Report
For Web Application

Version <2.2>

Revision History

Date	Version	Description	Author
<11/10/2017>	<1.0>	Provide the data structure and logic to carry out the functionalities dictated by the specification.	Thierno Diallo Chris Panican Anthony Tsui Jin Zhang
<11/15/2017>	<2.0>	Introduction and use cases are finalized. Diagrams were also inserted on corresponding sections	Thierno Diallo Chris Panican Anthony Tsui Jin Zhang
<11/16/2017>	<2.1>	Finished pseudocode of core methods and functionalities of the system. Finalized main functionality shown in Section 5.	Thierno Diallo Chris Panican Anthony Tsui Jin Zhang
<11/17/2017>	<2.2>	Reviewed Report 2 as a group to correct and improve grammar and statements.	Thierno Diallo Chris Panican Anthony Tsui Jin Zhang

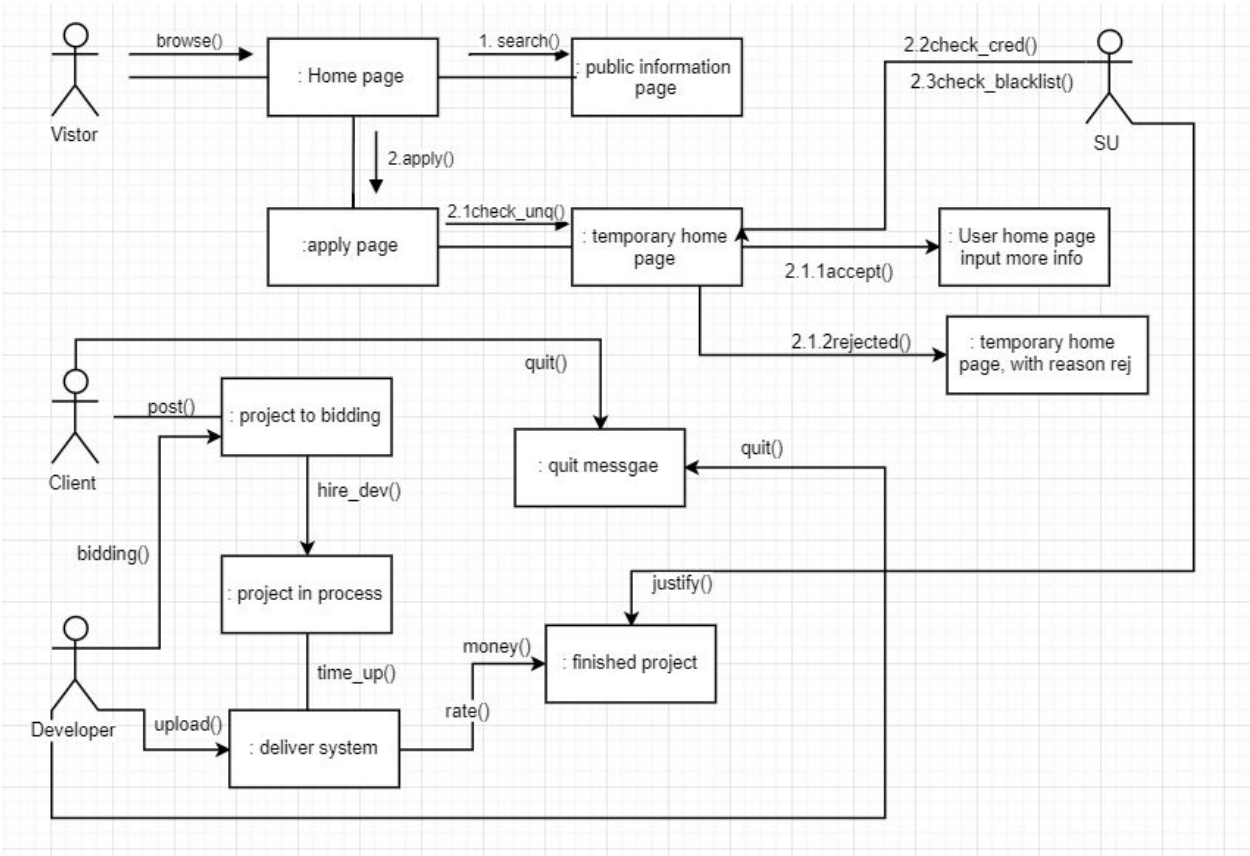
Table of Contents

Coding Turk System	Version <1.0>
Phase II: Design Report	Date: <17/11/2017>
<WWA17112017-2.2>	

1. Introduction:

The Coding Turk System is an online platform for clients and developers for system requests. The purpose of this document is to provide more technical details on how the system would work. It is recommended for the reader to have a basic knowledge of the previous report since this document will be more in-depth.

The diagram below shows the overall picture of the system using a collaboration class diagram. It shows how a visitor, client, developer, and admin (SU) have access to the site and their limitations. For instance, the visitor have access to the site and browse the bids, but unlike the client or developer, visitors cannot participate on projects and post things. The SU on the other hand, has specific actions only them can do.

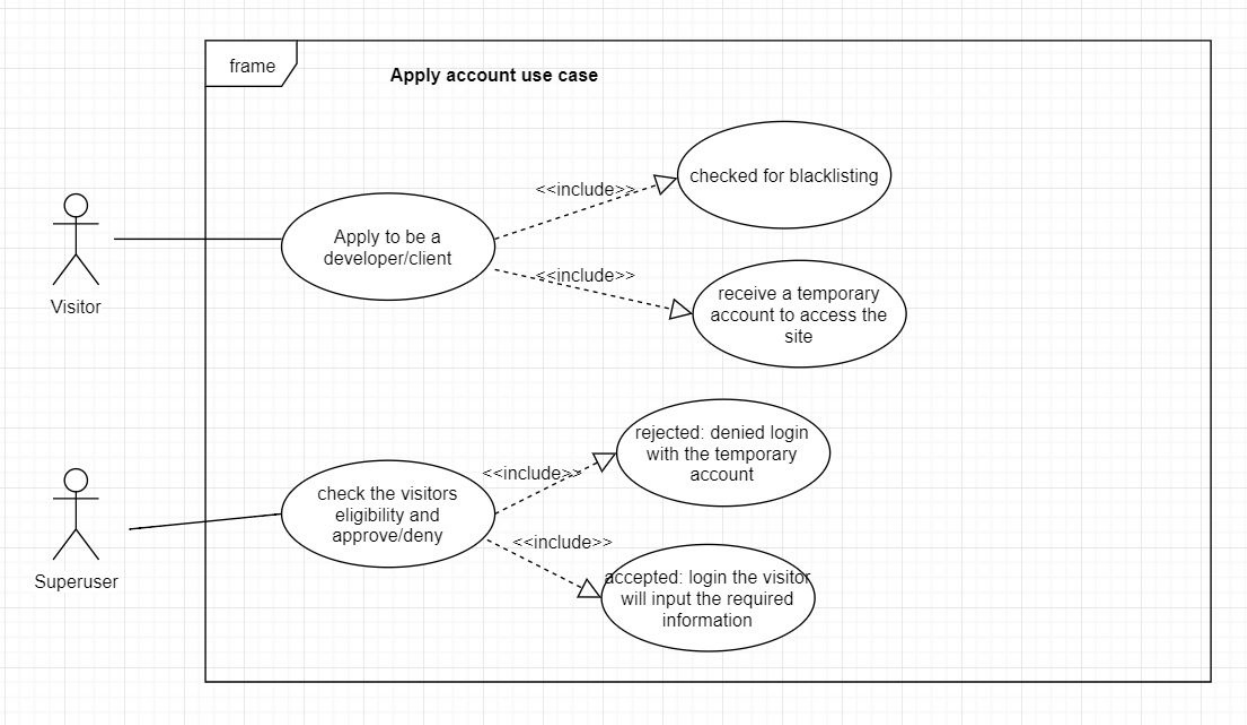


2. Use Cases

This section will show scenarios on how each user would commit actions to the site. With the use of diagrams, these scenarios can be explained better and it would be easier to follow. This section will be split in 3 parts: (1) Scenarios for each use case, (2) Collaboration diagram for each use case, and (3) Petri-net diagram for each use case.

❖ Scenarios for each use case

Use-case: Applying for an account

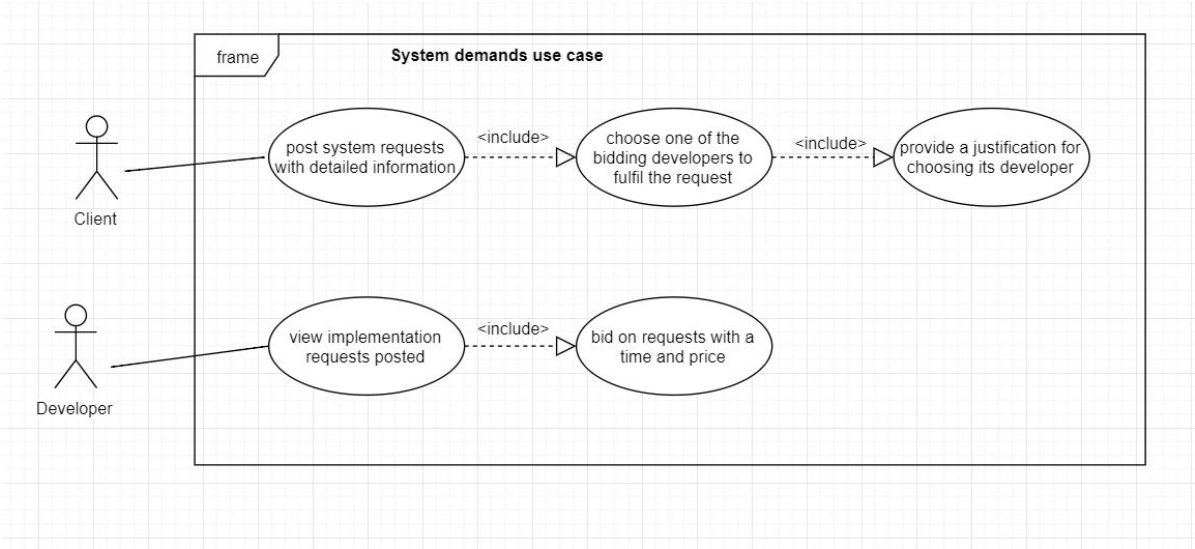


Users: Visitors

An unregistered user can apply to be a developer/client, and will be checked on blacklist

1. A visitor accessing the site can apply to be a developer/client
2. Applicants will receive a temporary account to access the site
3. SU will check the visitors eligibility and approve/deny
4. A. If rejected the visitor will be denied login with the temporary account
   B. If accepted, upon login the visitor will input registration information

Use-case: System demands



Users: Clients/Developers

For Posting System Requests/Demands:

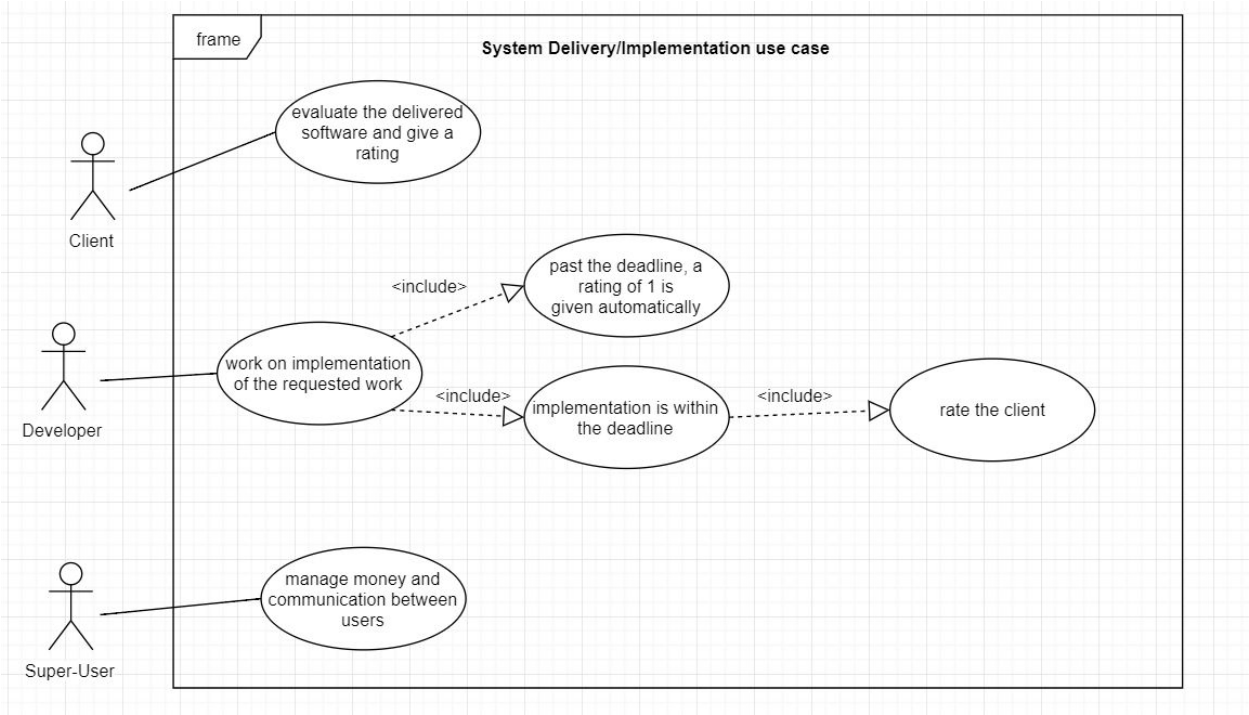
Clients can post system demands/implementations for developers to view/bid

1. Clients post detailed system requests on the website
2. Developers can bid on posted requests
3. Clients will choose one of the bidding developers to fulfil the request
4. Clients will provide a justification for choosing their developer
5. Half of the requested money will be transferred to the developer

Developers can bid on posted requests:

1. Developers can view posted system demands by Clients
2. Developers can then bid on requests with a proposed time and price
3. Half of the proposed payment is sent to the developer from the Client's account

Use-case: System Delivery/Implementation



Users: Clients/Developers/Super-Users

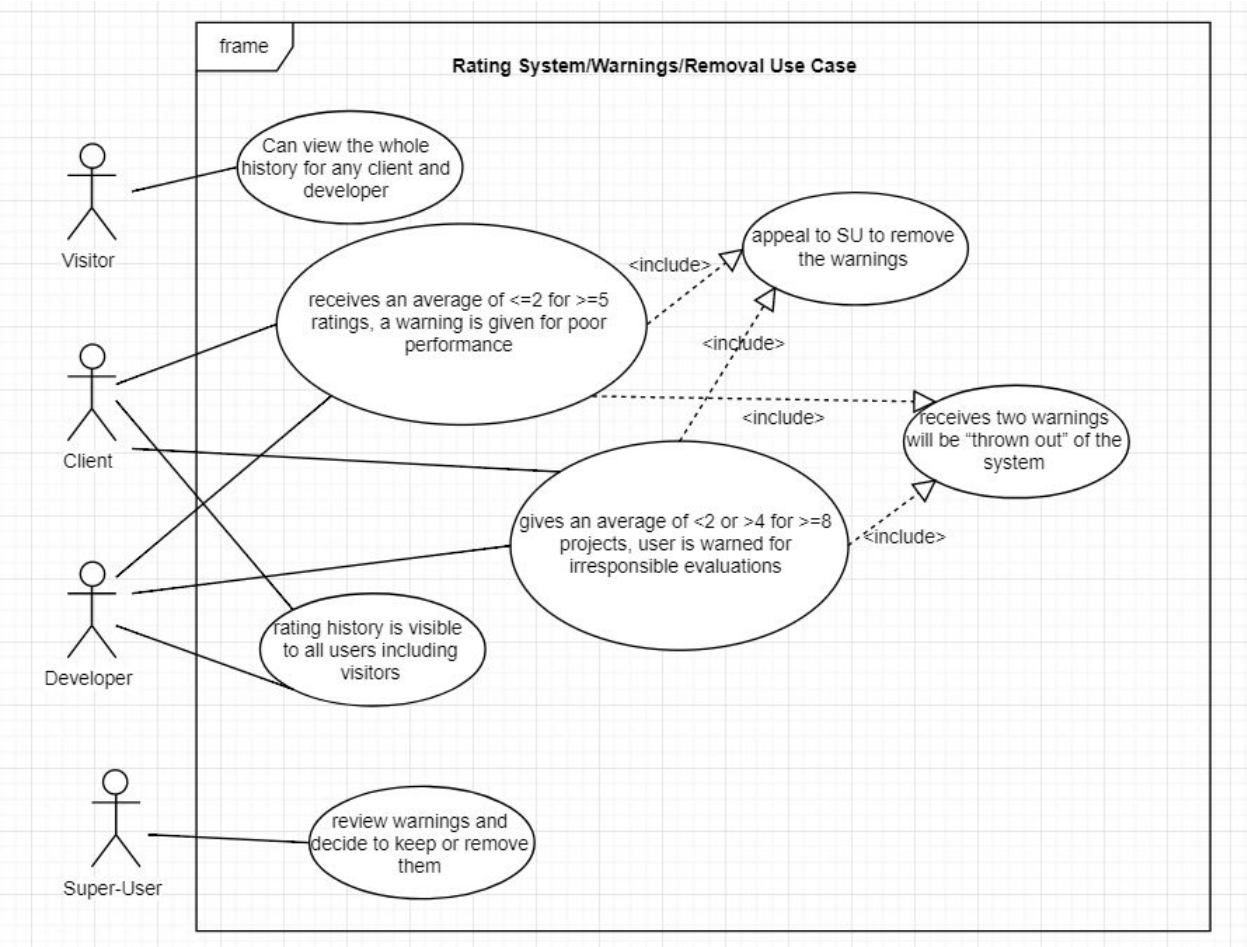
After a Client has bid on a developer for their system request, the developer is expected to implement it within the predetermined deadline. A rating system will be used afterwards for both client and developer.

1. Developer will work on implementation of the requested work
2. a. If the implementation is past the deadline, the initial payment is transferred back to the Client, with an extra penalty fee from the Developer. A rating of 1 is also given to the Developer.
   b. If the implementation is within the deadline, the remaining payment is transferred to the SU

The following assumes the implementation was on time:

3. The Client will evaluate the delivered software and give a rating
4. a. If the rating is >= 3 the payment is transferred from the SU to the Developer
   b. If the rating is < 3 the client must give a reason for the rating, with the SU evaluating the low rating
5. 5% of the payment will be charged for using the system, and transferred from the Client and Developer to the SU
6. The developer will rate the Client, providing a reason for <=2 rating

Use-case: Rating System/Warnings/Removal



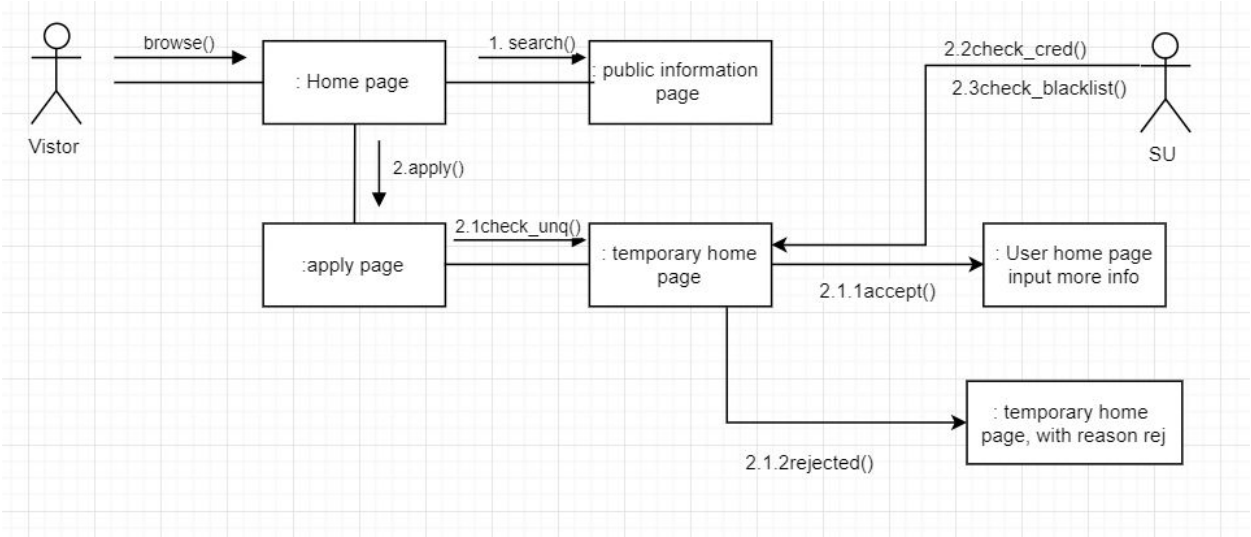
Users: Visitors/Clients/Developers/Super-Users

A rating system is used after a job between Clients and Developers, with an SU adjudicating.

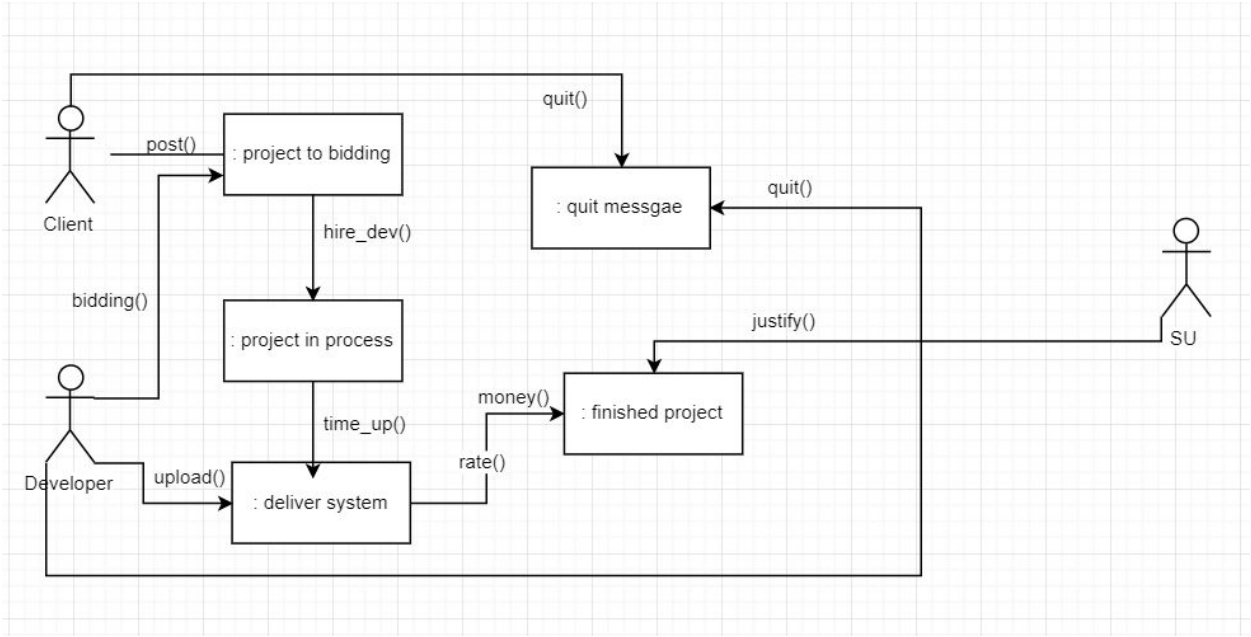
1. Clients and Developers will have their rating history visible to all users
2. If a user receives an average of <=2 for >=5 ratings, a warning is given for poor performance
3. If a user gives an average of <2 or >4 for >=8 projects, user is warned for irresponsible evaluations
4. Warned users can appeal to SU to remove the warnings
5. SU will review warnings and decide to keep or remove them
6. Any user that receives two warnings will be "thrown out" of the system
    a. Removed users will be able to login once for system closing matters
    b. Removed users can still protest to super-users to reverse this action
    c. Removed users are put on a "black-list", and cannot register for one year

❖ Collaboration diagram for each use case

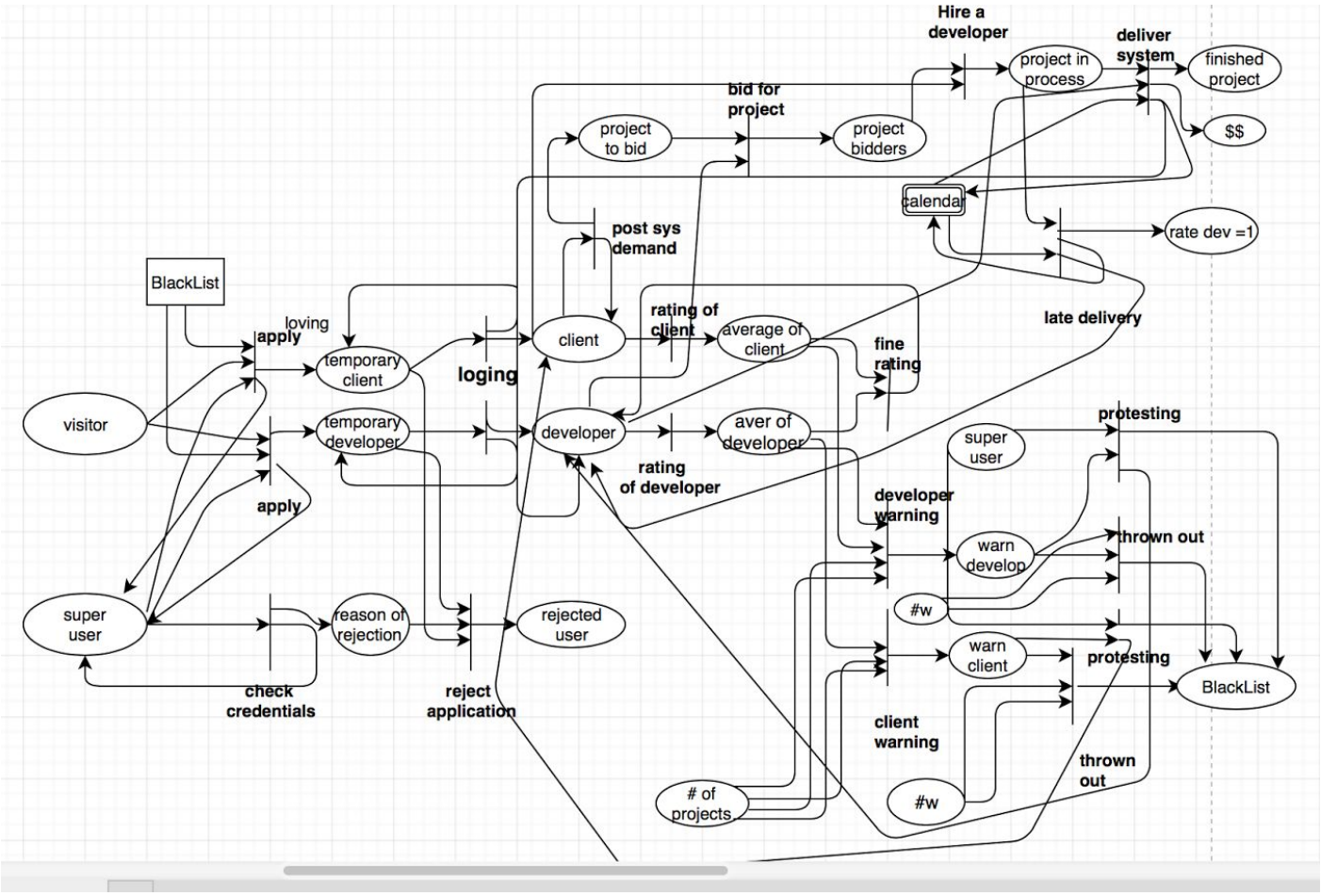
Use-case: Applying an account



Use-case: Post/delivery/rate project process



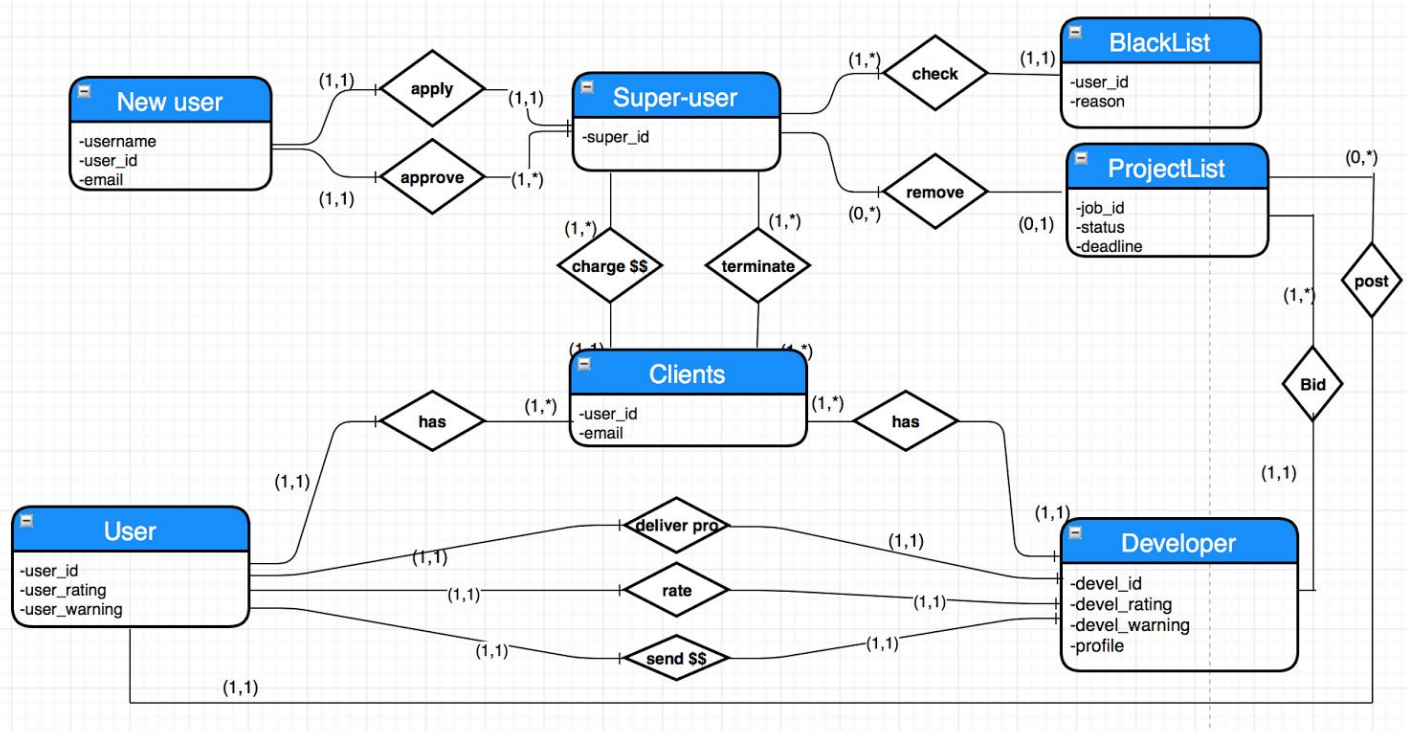
❖ Petri-net diagram for each case



3. E-R Diagram for the entire system

This section explains the Entity Relationship diagram of the entire system. It shows how the database stores its data and how the data is used to interact with each other. Each shapes has its own meaning, rectangle is an entity, diamond is an action, connecting lines show the relationship between entities. In the diagram, each entity has an attribute.

For example, a User entity has three attributes namely, user_id, user_rating, and user_warning. Each User can have/be a client, it can rate others, deliver a project, send money, and make a post.



4. Detailed Design

The following are the core methods utilized in the system, along with the pseudocode that will be used as guidelines for hard coding them. The pseudocode is tentative and is more of a general concept of the method rather than set rules for development.

Registration():

```
{
        Prompt user input for name and email;
        If (name in blacklist database) { return to register page with error message;}
        Else:
                Generate random visitor_id and password;
                insert visitor_id and password into visitor_accounts database;
                {send user to login_page with success message;}
}
```

Login():

```
{
        Prompt user input for username and password;
        if(username is visitor_account)
                { Send user to new_account page;
                  Prompt user for username, password, and confirm password;
                  if(username in user_accounts database)
                        {return to new_account page with error message;}
                 if(password =/= confirm_password)
                        {return to new_account page with error message;}
                 Insert username and password into user_accounts database;
                 Return user to login_page with success message;}
        if(username in user_accounts)
                { if (password corresponds to username  in databases)
                        {set session to user_id
                         set profile and user_controls corresponding to user_id
                         Send user to home_page with success message;
                }
}
```

Logout():

```
{
        Triggered by button press
        Prompt user for confirmation
        if(user presses yes)
        {reset session; send user to home_page with logout message;}
        if(user presses no or anywhere else on popup}
        {exit logout;}
}
```

View_Bids();

```
{
        Pulls up open bids on bids database
        Display small list of details on bids: (time, pay,title, etc)
        if(user searches by keyword or sorts in some way)
        {
          Find specifics in bids database, returns with search criteria
        }
        Enable user navigations options for each such as: bid, view details, etc
}
```

View_bid_details():

```
{
        Sends user to either pop up page or temporary viewing page for the bid
        From bids database:
        {Access the bid_id of the selected bid and returns the details on the page
         Including description, pay, language specifics, creator, deadline, etc;}
        Enable options for user such as bid, message creator, etc;
}
```

Place_bid():

```
{
        Prompt user for confirmation and bid amount;
        check(if user_id is a developer or client)
        if(user_id is a client) { return error that only developers can bid;}
        In bids database, find bids_id, update bid_amount list/set with additional bid with name
        check(bidders for bid_id):
         { if (user_id is in bidders) { update bid_amount corresponding to bidder;}
           else: {insert bid_amount and user_id into bidders;}
}
```

Post _project()

{

Accessed through user profile or homepage
Bid-posting will not be available for developers, but just to be safe,
system will check user_id:
if(user_id.checkrole() == developer)
{return user to homepage with error message;)
Prompt user for project details: project_title, description, deadline
Automatically collects data on: clients user_id, date created
Generates unique id for project_id and puts details into projects database
Creates corresponding data on bids database using project_id as key
On active bids/project page, will display the project once entered into the database
Sends user to active bids/project page with success message

}

Choose_bidder()

{

From active bids/project homepage
Prompt user for confirmation on button press (corresponding to the project)
Clients can only choose bidders on their own projects, so when confirmation occurs:
if(project_id.checkclient() != user_id) { return user to active bids/profile page with error}
if(developers bid != highest bid for the project_id)
{prompt client for explanation under project_notes/client_comments}
Removes the bid from the active bids/project page
Change values for: project_deadline , developer , and project_payment
Prompt user to go to either: active bids/developers profile page
Notify developer based on user_id for all in bidders for project_id
if(user_id == developer that is, the developer chosen)
{sends message/notification either thru email or on website that they were picked}
Else: { sends notification that bidding has closed for the project; }

}

submit_project()

{

Prompts developer to upload file or deliver externally {store method in delivery_method}
Regardless of which one call **rate_user()** on both project_developer and project_client
Remaining half of project_payment is transferred to project_developer
Update properties of project_id including deadline = CLOSED

}

rate_user(user_id)

```
{
        Prompts user upon system delivery/completion
        Client rates developer based on user_id under project_id
        Developer rates client based on user_id under project_client
        Users interact with GUI to submit: rating and comments, comments are optional
        Upon submitting the results:
        Run rating_check(user_id)() for both client and developer;

}
```

Rating_check

```
{
        Calculate average of last several ratings (different if client or developer)
        if(user_id.checkrole() == developer){
        if(user_rating =< developer_warn this is a constant) { increment user_warnings by 1;}}
        else: {if(user_rating >= client_warn also a constant) { increment user_warnings by 1;}}
        if(warning was sent) { send notifications to user_id;}
        Call warning_check() on user_id if warning sent
}
```

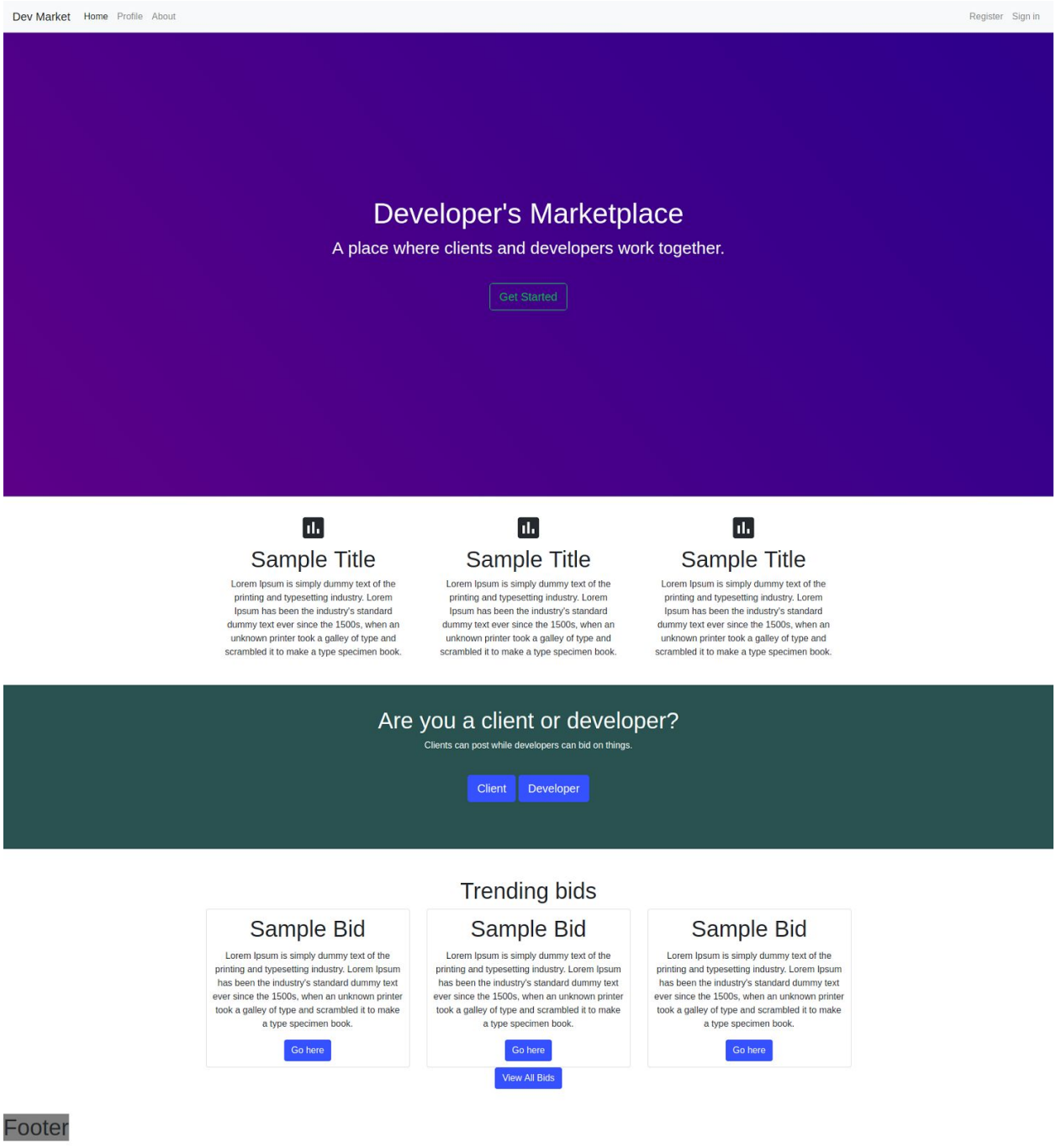
warning_check(user_id)

```
{
        If(user_id warnings >= 2) {
                send notification to user and super-user;
                Insert user_id into blacklist database;
                Forcibly logout banned user either through a password change or disconnect
                }
}
```

5. System Screen

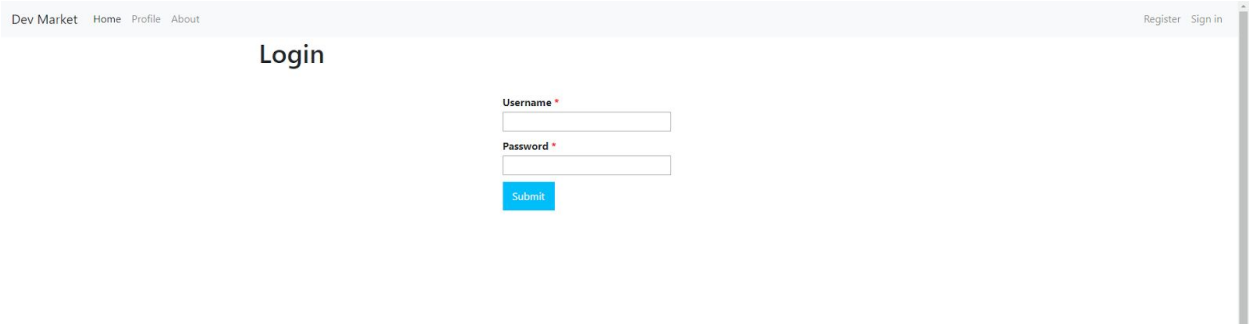
These are some of the prototype pages that are currently in our web application. These are our initial designs and there will be more improvements in the future. This section also shows one of our website functionality and we will feature the steps on how a client creates a posting.

Home page

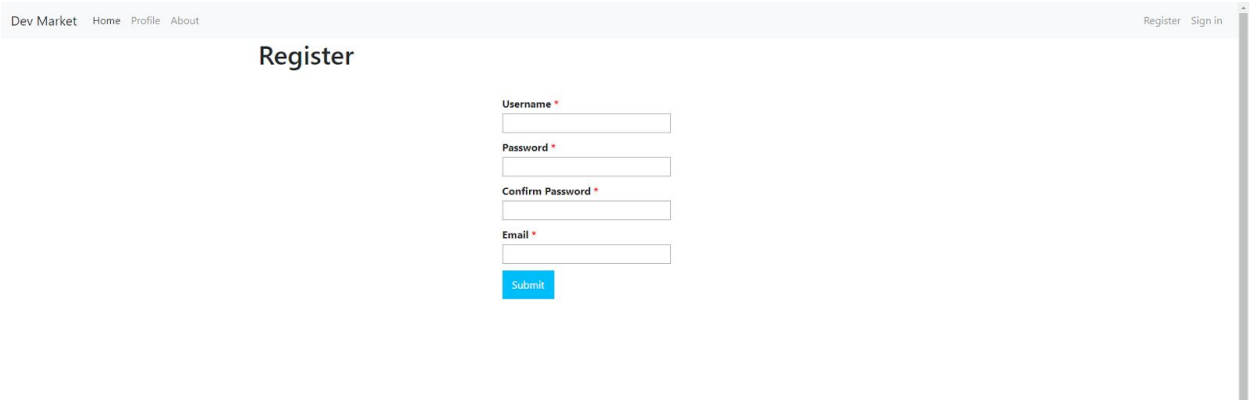


Footer

Login Page

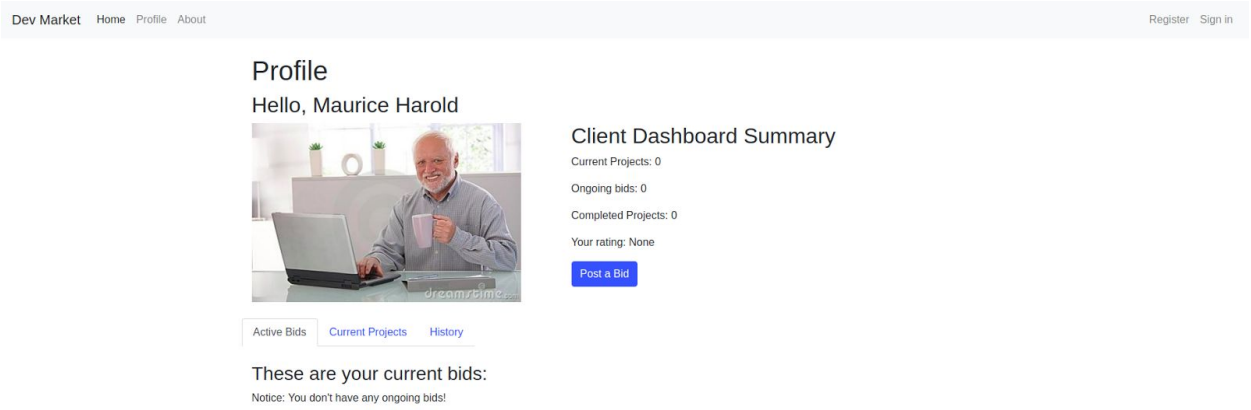


Register Page

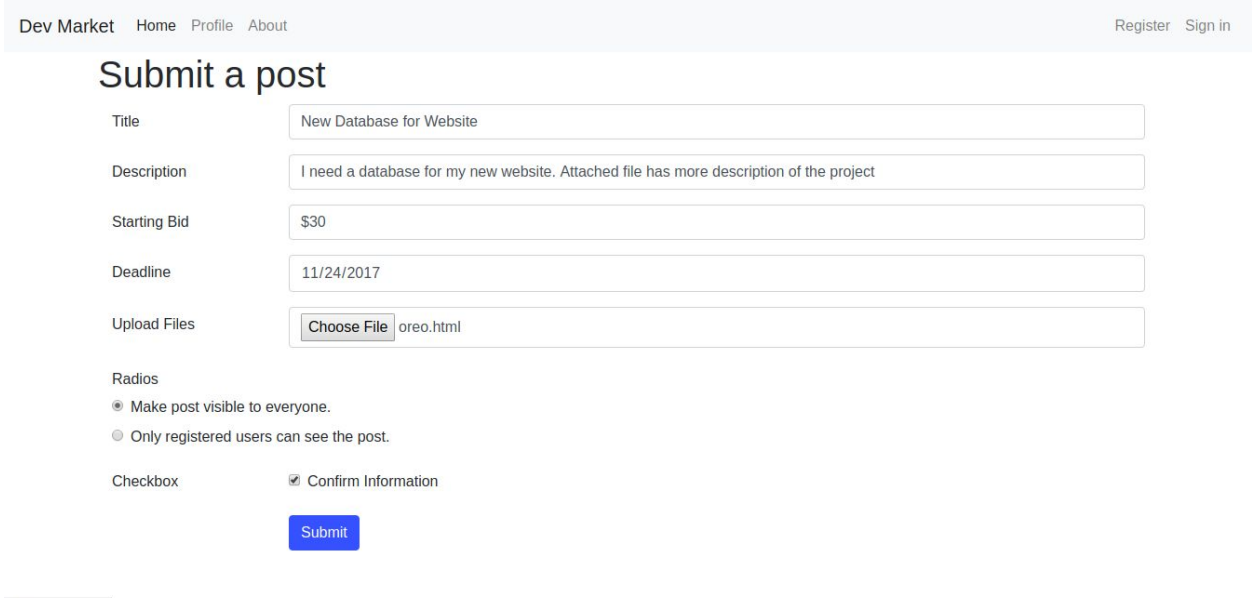


We will provide a prototype on how a client would post a bid to the platform. Shown below are the screenshots of pages that are used to achieve this functionality.

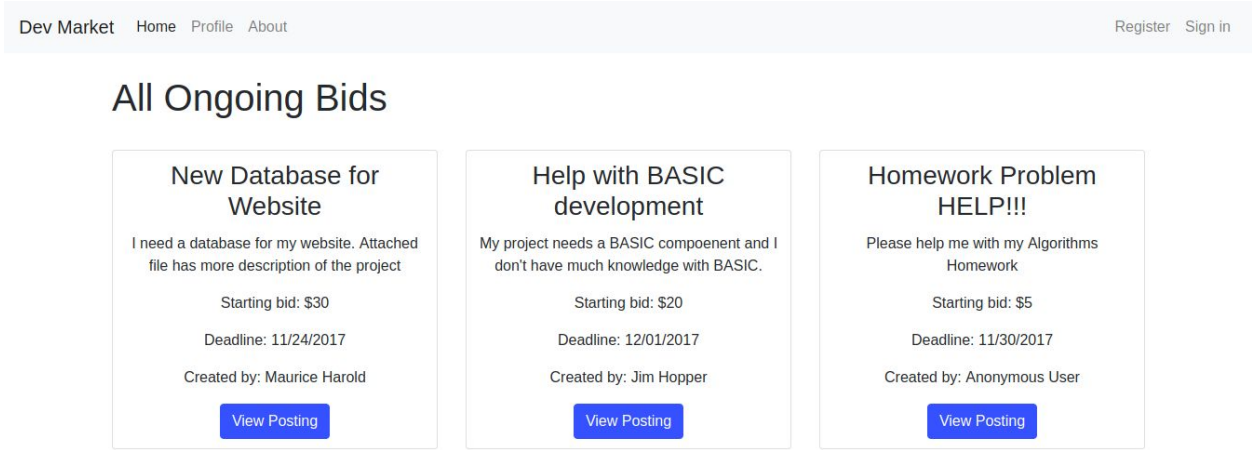
First, the client should click at the “Post a Bid” button on their home-page or profile page.



Then, they will be greeted by a new page which prompts them to enter a title, project description, starting bid, deadline, an option to upload files, etc.

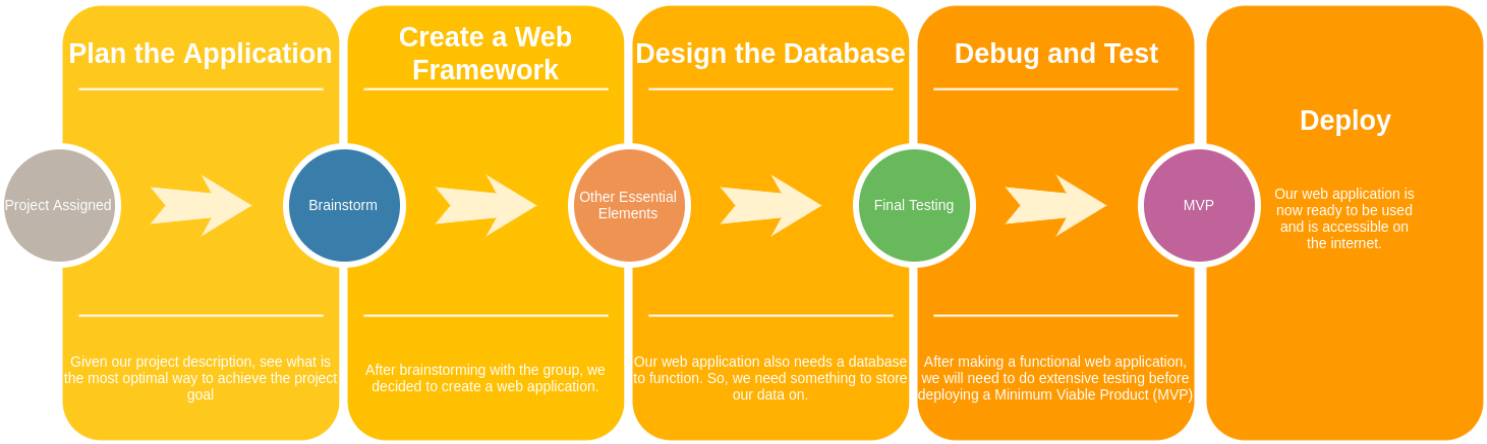


After submitting their bid, the post will come up on the bidding page, where developers can decide which projects to take on.



6. Minutes of Group Meetings

Every Tuesday and Friday, WeWantA Inc. meet together to discuss our future plans and current progress. Since our group practices Agile Scrum methodology, we believe that we can make a good product in short amount of time, rather than following the waterfall methodology. Every group meetings, we all speak one by one, and a person states a summary of their progress and their future plans. On top of these small tasks is a general roadmap that we follow. This roadmap consists some of the most essential steps to produce a Minimum Viable Product (MVP). Shown below is our roadmap:



However, even after accomplishing the roadmap, we will still continue to work on the project to optimize our application, make UI more appealing, etc.

By following our agenda, we managed to produce some great results in short amount of time. We keep track of these milestones and here are some of them:

Date Completed	Milestone
11/17/2017	Completed client posting functionality.
11/10/2017	Completed database design
11/03/2017	Completed home page design and UI
10/27/2017	Started HTML/CSS for index and base pages
10/20/2017	Completed initial design of the website

7. The first phase report

We've reviewed our initial diagrams from the first report based on the comments and suggestions provided. We fixed the errors the was told to us such as incorrect notations on diagrams etc.