CS2102 Database Project Report

Carolend

DONG SHAOCONG, HE XINYI, LIU YULIN, WANG ZEXIN

Department of Mathematics

National University of Singapore

AY2017/18 Semester 1

Abstract

Acknowledge

We would like to thank Associate Professor Bressan Stephane for his helpful supervision throughout the course of this project.

Contents

1 Introduction

In this project, we are required to build a stuff sharing website. the system allows people to borrow or lend stuff that they own (tools, appliances, furniture or books) either free or for a fee. Users advertise stuff available (what stuff, where to pick up and return, when it is available, etc.) or can browse the available stuff and bid to borrow some stuff. The stuff owner or the system (your choice) chooses the successful bid. Each user has an account. Administrators can create, modify and delete all entries.

1.1 Developing Specifications

After seeing through the relevant products specifications and the website requirements, we decided to use PHP as back end programming language, $Javascript$ as front end developing language, $MySQL$ as our database. We used $Laravel$, which is the most popular web application framework for PHP.

From the project requirement, Eloquent ORM in built in Laravel to access and manipulate the database is not allowed. Therefore, anything related to database management, access, and manipulation is down by importing PHP $mysqli$ library and execute raw SQL queries.

To develop this web application, we utilise the Model View Controller (MVC) framework.

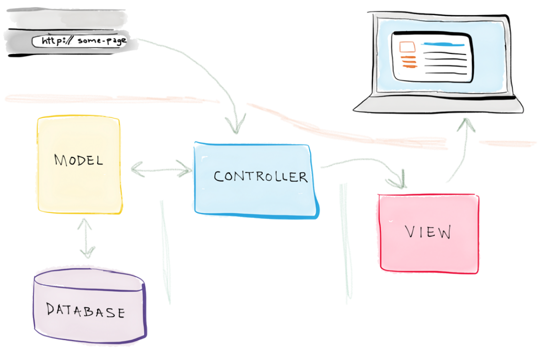


Figure 1: Model View Controller (MVC)

We use CSS Scaffolding in Laravel to take care of the views. Buttons and redirections from the user side are implemented in $Javascript$. The models and controllers are written in PHP. The specific $MySQL$ database we used it *root* user's database named *blog*. The website is still hosted on `localhost:8000` only.

From our modelling for this problem, there are two types of users. The admin users will have different interfaces and unlimited access to all entries. From the user management button in the admin user's profile page. All the users information can be modified and deleted. New users can be added on this panel as well.

2 Database Design

2.1 Entity-Relationship Diagram

2.2 Entities

Users

Attribute	Domain
email	VARCHAR(64)
username	VARCHAR(64)
password	VARCHAR(64)
mobile	INT(8)
address	VARCHAR(128)
points_available	INT(3)
admin	INT(1)
credit_rating	NUMERIC
created_at	TIMESTAMP

Items

Attribute	Domain
name	VARCHAR(64)
avatar	VARCHAR(256)
owner	VARCHAR(64)
description	TEXT
available	VARCHAR(5)
created_at	TIMESTAMP

Posts

Attribute	Domain
item	VARCHAR(64)
title	VARCHAR(64)
location	VARCHAR(128)
description	TEXT
start	TIMESTAMP
end	TIMESTAMP
created_at	TIMESTAMP

Bids

Attribute	Domain
bidder	VARCHAR(64)
post	VARCHAR(64)
status	CHAR(7)
points	INT(3)
created_at	TIMESTAMP

Loans

Attribute	Domain
bid	VARCHAR(64)
post	VARCHAR(64)
start	TIMESTAMP
end	TIMESTAMP
comments	TEXT
status	VARCHAR(8)
created_at	TIMESTAMP

2.3 Relational Schema

2.4 Schema Functions

3 SQL Queries

3.1 Simple Queries

The following simple query returns all the information about one particular bidding made. This is displayed when the bidder wants to edit his own bidding.

```
SELECT *
FROM bids
WHERE bids.bidid = bidid;
```

The following simple query returns all the information about all the items. This is displayed when user accesses the ‘items’ page.

```
SELECT *
FROM item i;
```

The following simple query returns all the information about one particular post. This is displayed when the poster wants to edit his own post.

```
SELECT *
FROM posts
WHERE posts.postid = postid;
```

The following simple query returns all the information about one user with userid specified. This is displayed when the user visited his ‘user’ page.

```
SELECT *
FROM users
WHERE users.id = userid;
```

3.2 Aggregate Queries

The following aggregate query returns the maximum bidding points for one particular post with postid specified. This is displayed when the poster wants to check the maximum bid made for his post.

```
SELECT MAX(b.points)
FROM bids b, posts p
WHERE b.post = p.postid AND p.postid = postid;
```

3.3 Nested Queries

3.4 Queries using INNER JOIN

3.5 Queries using EXISTS

3.6 Queries using set operations

3.7 Insertions, Deletions and Updates

INSERT INTO bids (bidder, post, points) VALUES (email, postid, points); INSERT INTO loans (bid, post, status) values($bidId, postid, using_status); INSERT INTO items (description, available, name, owner, avatar) values (description, available, name, owner, filename); INSERT INTO posts (item, title, location, description) VALUES (itemId, title, location, description); INSERT INTO bids (status, bidder, post, points) VALUES ('FAILURE', email, postid, point);

3.8 Query using view

Create a view item_popularity which contains all the popularity information about the items.

```
CREATE VIEW item_popularity AS
SELECT i.itemid as itemid, i.owner as owner, COUNT(*) AS popularity
FROM items i, posts p, bids b
WHERE i.itemid = p.item AND p.postid = b.post
GROUP BY i.itemid, i.owner;
```

The following query selects the average popularity of the items posted by each user.

```
SELECT u.email, AVERAGE(i.popularity)
FROM users u, item_popularity i
WHERE u.email = i.owner
GROUP BY u.email;
```

The following query selects the average popularity of the items bided by each user.

```
SELECT u.email, AVERAGE(i.popularity)
FROM users u, item_popularity i, posts p, bids b, loans loans
WHERE u.email = b.bidder AND b.bidid = l.bid AND b.post = p.postid AND p.item = i.itemid
GROUP BY u.email;
```

The following query remove the view created.

```
DROP VIEW if exists item_popularity
```

4 Web Interface Design

We aims to have a friendly, easy-to-use interface for our web application. The following sections are about the pages in our web interface.

4.1 Sign Up Page

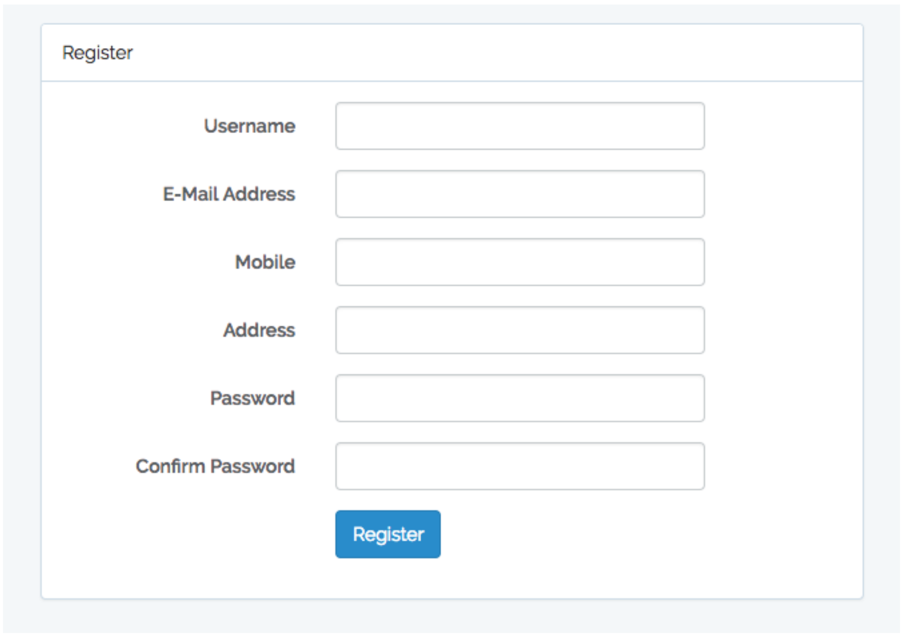


Figure 2: New User registration page

From this page, new users will be able to sign up and enter our system.

4.2 Log In Page

Other users will be able to log in via this page.

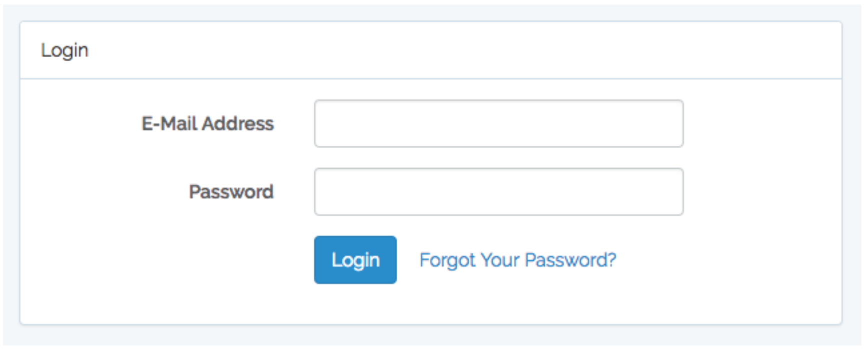
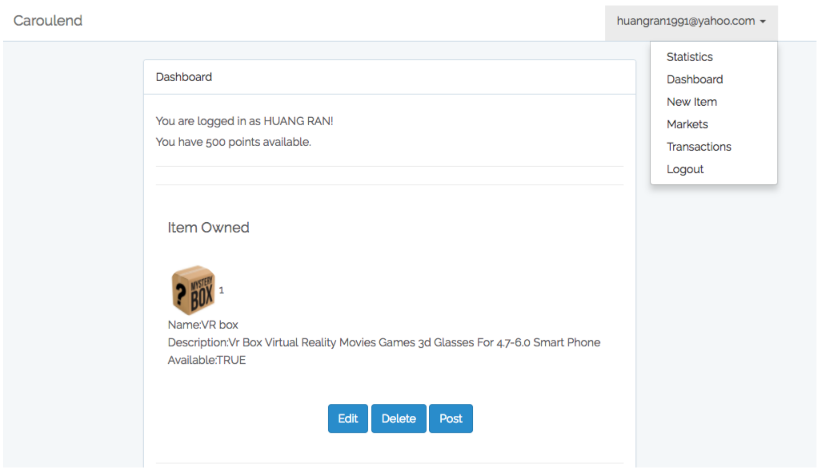


Figure 3: User login page

4.3 Application

Upon logging in, the user will be directed to his dashboard. He can easily see the points available for him or her. Besides that, the items the user owning, the posts he or she has submitted, the posts the user is currently bidding and the items the user is currently borrowing will be displayed sequentially thought this the page.
In addition, from the top right drop down list, the user can go to other pages ranging form his



You are logged in as HUANG RAN!
You have 500 points available.

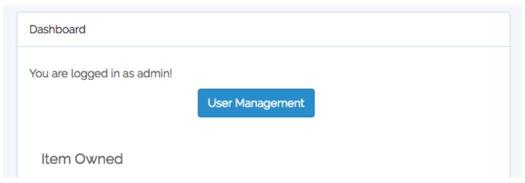
Name:VR box
Description:Vr Box Virtual Reality Movies Games 3d Glasses For 4.7-6.0 Smart Phone
Available:TRUE

Figure 4: User login page

or her using statistics. User can also create new item, go to the posting markets and checkout current transactions and transacting history. This includes the user's posts that are being bid by other users, the user's current lending items and the past transactions.

4.4 Administration

There is a special type of user called admin user. They have their admin column equal to 1 and the can have unlimited access to access, delete, update and create any entries in our database. Users are not able to sign up an admin account. The only possible way to become an admin is to update the table in our database directly. Implemented in this way, the administration page is very secure and the current normal users' information are properly protected. As shown in this screen shot,



You are logged in as admin!

Figure 5: User login page

the admin user can access any entries inside this application. If he or she wants to manage the current users, they can do so by clicking the user management button and the user records can be updated, deleted or created on our administration user management panel.

5 Sample table and diagram insertion

StrikePrice	Closed-form formula	Ordinary Monte Carlo	Control Variate
105	10.0022021172	10.005252032814727	10.005252032814751
110	8.02638469385	8.0166707887876427	8.016670788787664
115	6.37924904693	6.3659464432937911	6.3659464432937733
120	5.02541348179	5.0148870431746415	5.0148870431746184
125	3.92690420603	3.9241698581683577	3.9241698581683728
130	3.04592058431	3.044679765401427	3.0446797654014213
135	2.34679877596	2.3310307686205207	2.3310307686205225
140	1.79723400902	1.8055531375480696	1.8055531375480751
145	1.36889248498	1.3630119824610754	1.3630119824610734
150	1.03756650489	1.0254470920297398	1.0254470920297361
155	0.783018613011	0.77819086371547286	0.77819086371547308
160	0.588637155719	0.5924848566970754	0.59248485669707973
165	0.440997057228	0.44342591182216823	0.4434259118221664
170	0.329392108384	0.32449718396190719	0.32449718396190735
175	0.245381782063	0.2462392632801686	0.24623926328016907
180	0.182377553986	0.17995020687354496	0.1799502068735449
185	0.13528073067	0.13478417666883458	0.13478417666883413
190	0.100175092579	0.099449778570450814	0.099449778570450523
195	0.0740722950356	0.074452813719303179	0.074452813719303304
200	0.0547050187389	0.051631534936688268	0.051631534936688074

Figure 6: Effect of control variate in pricing European call options

6 Conclusion

correct way of citing something:[1]

References

[1] Yves Hilpisch, *Python for Finance*. O'Reilly Media, 2015.

[2] Paul Glasserman, *Monte Carlo methods in financial engineering*. Springer, 2010.

[3] Daniel Duffy, *Finite difference methods in financial engineering: A partial differential equation approach*. John Wiley&Sons, 2006.

[4] Mark Broadie, Paul Glasserman, Steven Kou, *A Continuity Correction for Discrete Barrier Options*. Mathematical Finance, 1997.