CS2102: DATABASE SYSTEMS

Project

Damien Sim (A0155983N)

Rebecca Tan (A0158203M)

Khor Shao Liang (A0160529E)

Teo Wen Zong (A0104287H)

National University of Singapore

Contents

[1.0 Server 1](#_Toc497728760)

[2.0 ER Diagram 1](#_Toc497728761)

[3.0 Project 2](#_Toc497728762)

[3.1 Relational Schema 2](#_Toc497728763)

[3.1.1 Account Table 2](#_Toc497728764)

[3.1.2 Task Table 2](#_Toc497728765)

[3.1.3 Bid Table 2](#_Toc497728766)

[3.2 Sample Features 3](#_Toc497728767)

[3.2.1 Users whose task have the highest amount of bid in a particular category 3](#_Toc497728768)

[3.2.2 Pagination 3](#_Toc497728769)

[3.2.3 Stored procedure for adding user 3](#_Toc497728770)

[3.2.4 Stored procedure for dashboard completed task statistics 3](#_Toc497728771)

[3.2.5 Update bid status trigger 4](#_Toc497728772)

[3.2.6 Popular views 4](#_Toc497728773)

[4.0 Images 5](#_Toc497728774)

[4.1 Landing Page 5](#_Toc497728775)

[4.2 Login Page 6](#_Toc497728776)

[4.3 Registration Page 6](#_Toc497728777)

[4.4 Search Page 7](#_Toc497728778)

[4.5 Dashboard 7](#_Toc497728779)

[4.5.1 User Dashboard 7](#_Toc497728780)

[4.5.2 Admin Dashboard 8](#_Toc497728781)

[4.6 Admin View All Tasks 8](#_Toc497728782)

[4.7 Create Task 9](#_Toc497728783)

# 1.0 Server

We use WAPP (Windows, Apache, PHP, Postgres) Stack for our project.

Web server: Apache

Server Page language: PHP, Javascript

Database management: Postgres

# 2.0 ER Diagram

Figure 1 below shows the ER Diagram for our project.

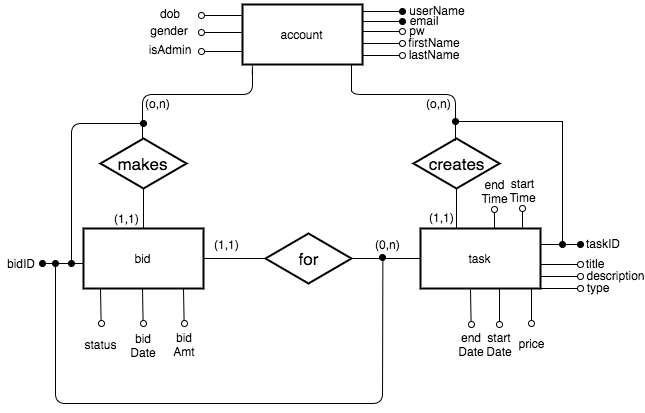


Figure 1: ER Diagram of our Project

# 3.0 Project

## 3.1 Relational Schema

### 3.1.1 Account Table

CREATE TABLE account(

userName VARCHAR(64) PRIMARY KEY,

email VARCHAR(128) UNIQUE,

pw VARCHAR(255) NOT NULL,

firstName VARCHAR(128) NOT NULL,

lastName VARCHAR(32) NOT NULL,

dob DATE NOT NULL CHECK (dob < (current\_date - interval '18' year)),

gender VARCHAR(6) NOT NULL CHECK (gender = 'Male' OR gender = 'Female'),

isAdmin boolean NOT NULL

);

### 3.1.2 Task Table

CREATE TABLE task(

taskID SERIAL,

userName VARCHAR(64) REFERENCES account(username) ON DELETE CASCADE,

title VARCHAR(255) NOT NULL,

description VARCHAR(512) NOT NULL,

type VARCHAR(64) NOT NULL,

price NUMERIC NOT NULL,

startDate DATE NOT NULL CHECK (startDate >= current\_date),

startTime TIME NOT NULL,

endDate DATE NOT NULL CHECK (endDate >= startdate),

endTime TIME NOT NULL,

PRIMARY KEY (taskID, username)

);

### 3.1.3 Bid Table

CREATE TABLE bid(

bidID SERIAL NOT NULL,

taskID INTEGER NOT NULL,

bidder VARCHAR(64) NOT NULL CHECK (bidder <> taskOwner) REFERENCES account(userName) ON DELETE CASCADE,

taskOwner VARCHAR(64) NOT NULL REFERENCES account(userName) ON DELETE CASCADE,

status varchar(8) NOT NULL CHECK (status = 'Pending' OR status = 'Accepted' OR status = 'Rejected'),

bidDate DATE NOT NULL CHECK (bidDate <= current\_date),

bidAmt NUMERIC NOT NULL,

PRIMARY KEY (bidID,taskID,bidder),

FOREIGN KEY (taskID,taskOwner) REFERENCES task(taskID,userName) ON DELETE CASCADE

);

## 3.2 Sample Features

### 3.2.1 Users whose task have the highest amount of bid in a particular category

SELECT t1.username FROM bid b1, task t1

WHERE t1.taskid = b1.taskid AND t1.username = b1.taskowner

AND t1.type = 'Miscellaneous'

GROUP BY t1.taskid, t1.username

HAVING COUNT (\*) >= ALL (SELECT COUNT (\*) FROM bid b2, task t2

WHERE t2.taskid = b2.taskid

AND t2.username = b2.taskowner

AND t2.type = 'Miscellaneous' GROUP BY t2.taskid);

Description: Retrieves the user with the highest number of bids for a particular category.

### 3.2.2 Pagination

$result = pg\_query($db, "SELECT \* FROM task LIMIT 10 OFFSET $page1;");

Description: Divides the content of each page equally to a max of 10 per page

### 3.2.3 Stored procedure for adding user

CREATE FUNCTION add\_user(userName VARCHAR(64), email VARCHAR(128), pw VARCHAR(255), firstName VARCHAR(128), lastName VARCHAR(32), dob DATE, gender VARCHAR(6), isAdmin boolean)

RETURNS void AS $$

BEGIN

INSERT INTO account VALUES (username,email,pw,firstName,lastName,dob,gender,isAdmin);

END;

$$ LANGUAGE plpgsql;

Description: Adding new users to the account table in the database

### 3.2.4 Stored procedure for dashboard completed task statistics

CREATE OR REPLACE FUNCTION dashboard\_completed\_task(userid VARCHAR(64))

RETURNS TABLE (taskid INT, username VARCHAR(64), title VARCHAR(255), description VARCHAR(512), type VARCHAR(64), price NUMERIC, startdate DATE, starttime TIME, enddate DATE, endtime TIME)

AS $$

BEGIN

RETURN Query (

SELECT t.taskid, t.username, t.title, t.description, t.type, t.price, t.startdate, t.starttime, t.enddate, t.endtime

FROM task t, bid b

WHERE t.enddate < date\_trunc('day', CURRENT\_TIMESTAMP)

AND t.taskid = b.taskid

AND t.username = b.taskowner

AND t.username = userid

AND b.taskOwner = userid

AND b.status = 'Accepted'

);

END

$$ LANGUAGE plpgsql;

Description: Using stored function to retrieve the total number of task completed by a user.

### 3.2.5 Update bid status trigger

CREATE TRIGGER updateOtherBids

AFTER UPDATE

ON bid

FOR EACH ROW

EXECUTE PROCEDURE updateBidStatus();

Description: Using trigger function to update the other bid status upon a change in a specific bid status. To ensure that the integrity of the bids correspond to each other. E.g. Accepting one bid causes the other bids to be rejected.

### 3.2.6 Popular views

CREATE OR REPLACE VIEW popular\_housing\_agent AS

SELECT t.username FROM bid b, task t

WHERE t.taskid = b.taskid

AND t.username = b.taskowner

AND t.type = 'Housing Agent'

GROUP BY t.taskid, t.username HAVING COUNT (\*) >= ALL

(SELECT COUNT (\*) FROM bid b2, task t2

WHERE t2.taskid = b2.taskid

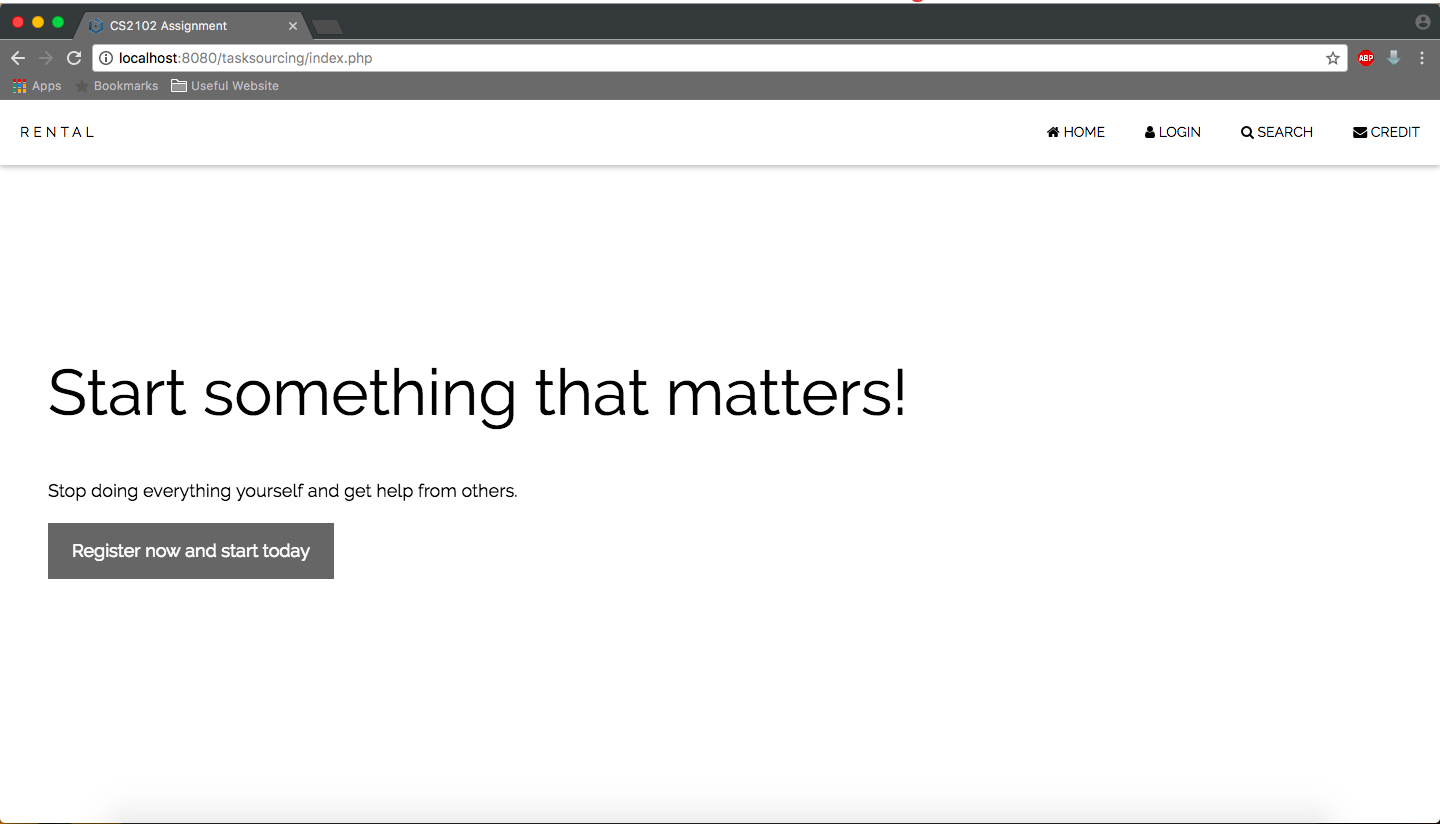
AND t2.username = b2.taskowner

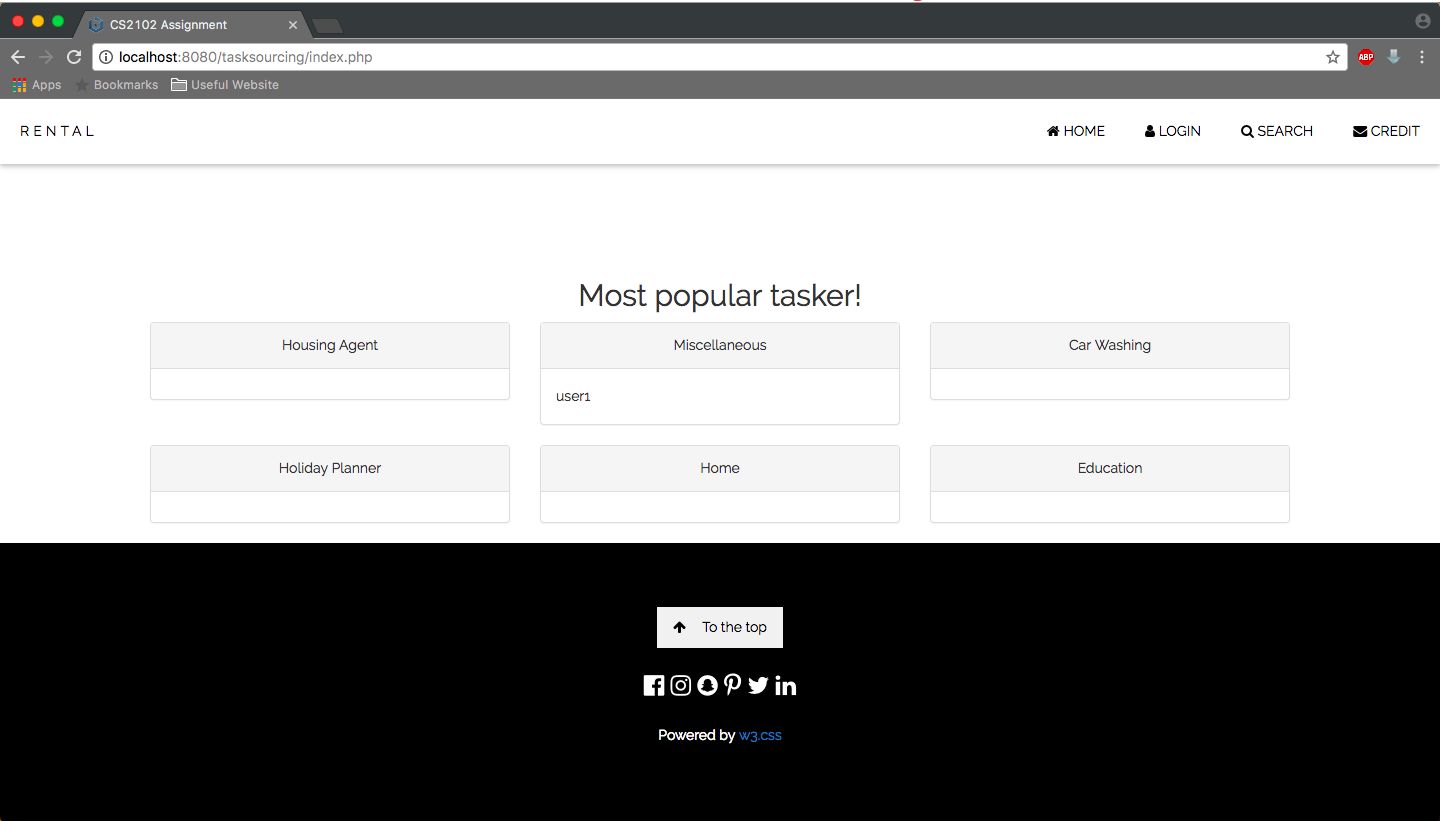
AND t2.type = 'Housing Agent' GROUP BY t2.taskid);

Description: Retrieves user with the highest number of task created for a category.

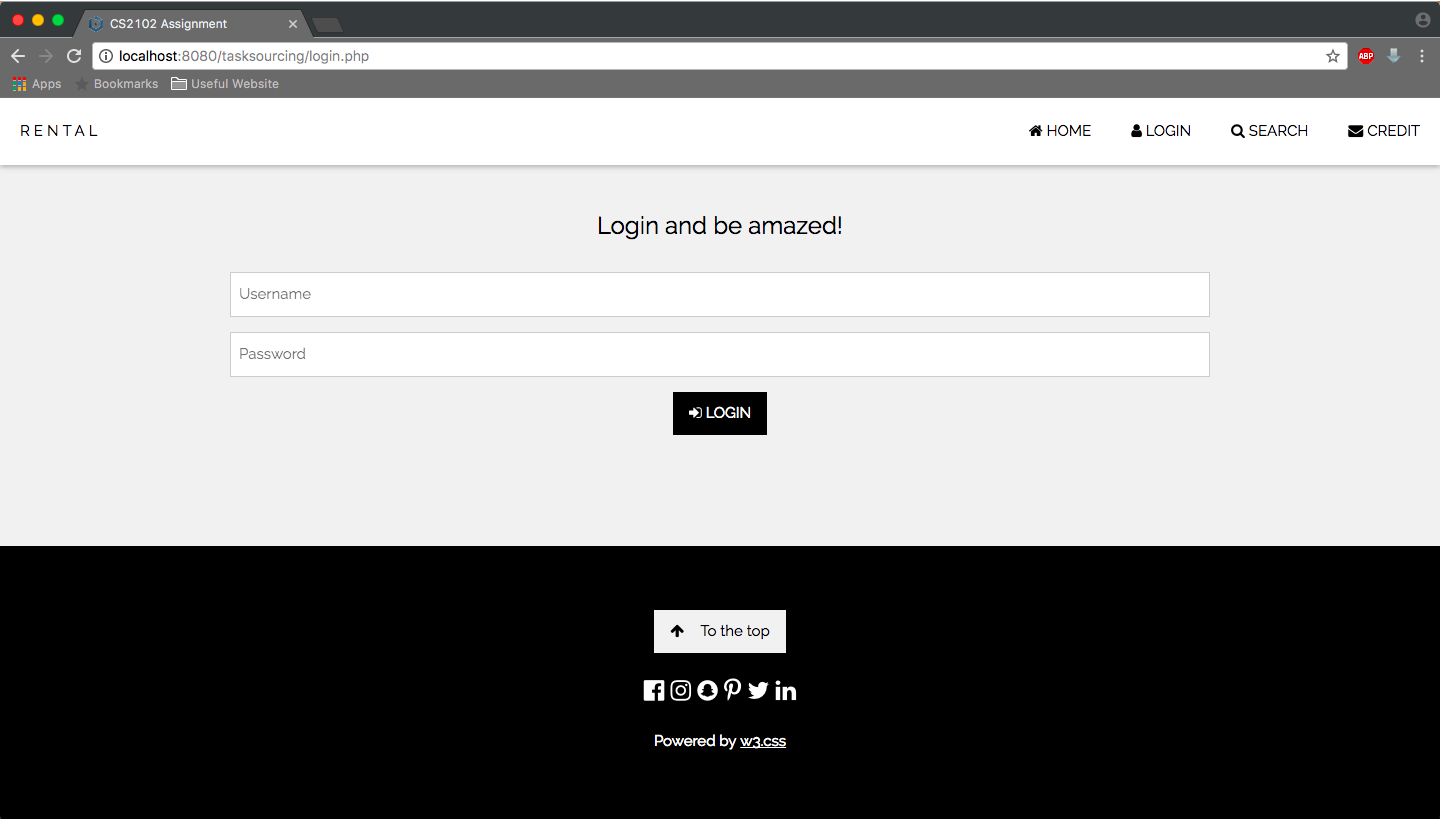
# 4.0 Images

## 4.1 Landing Page

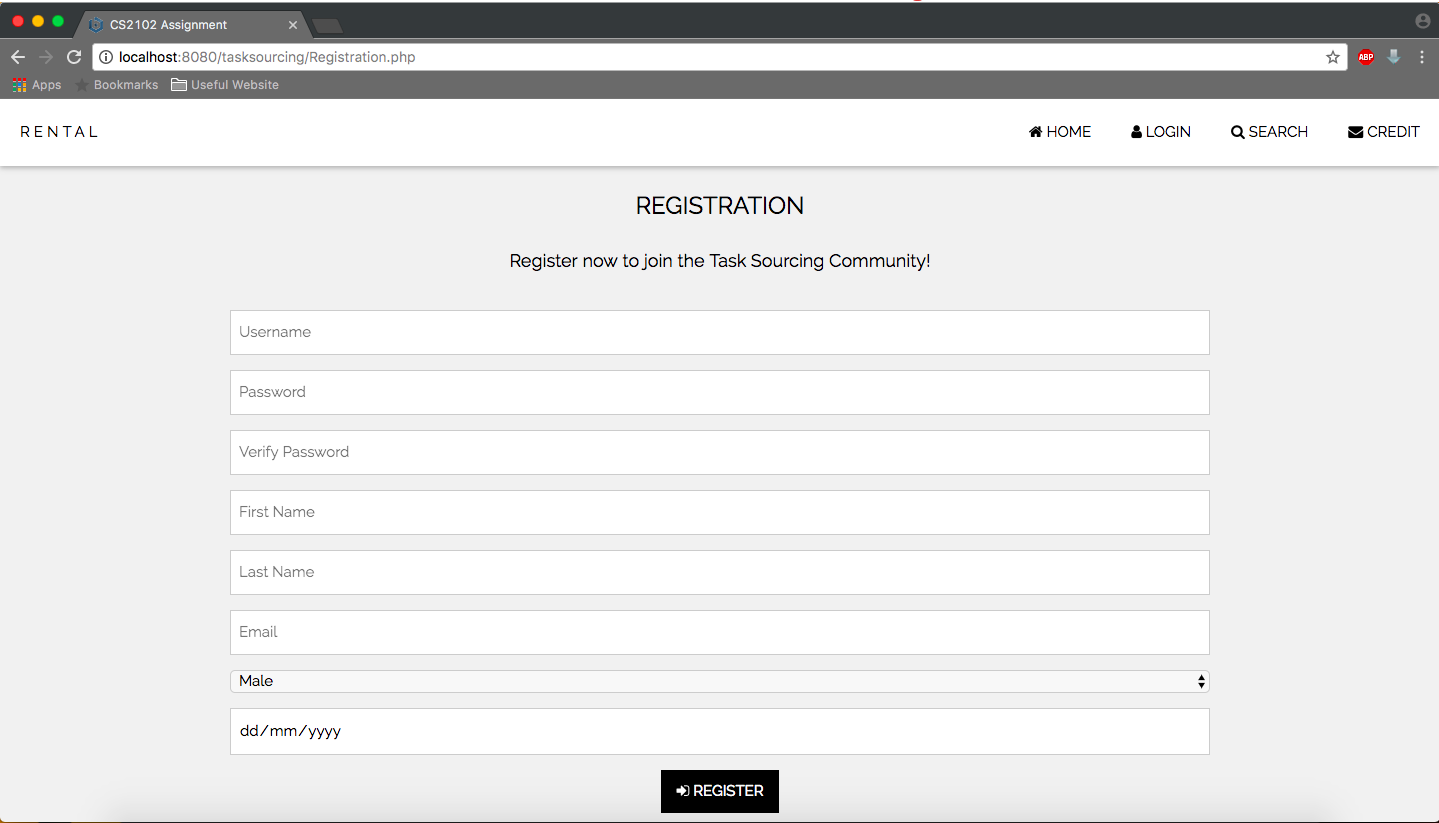




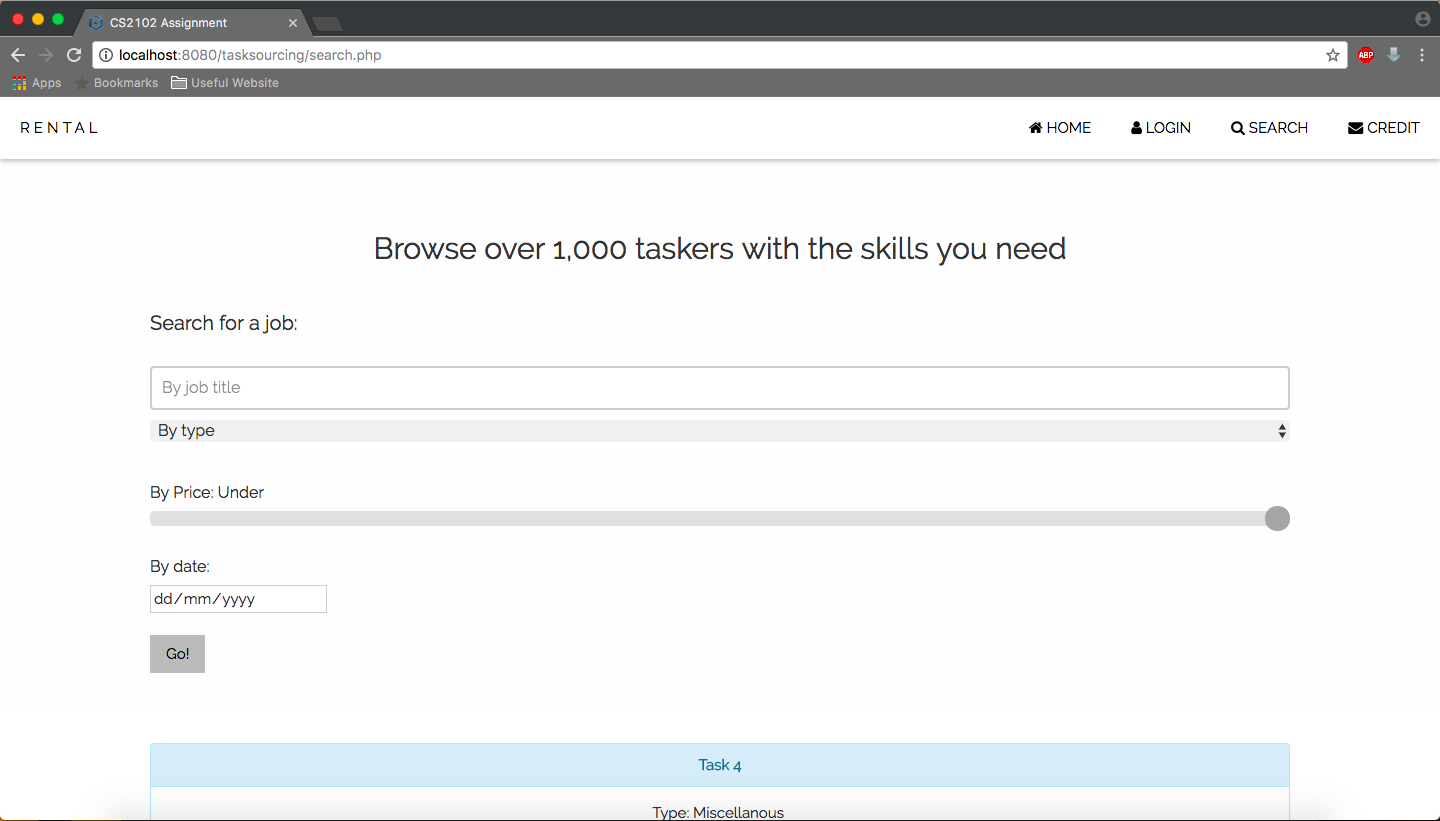
## 4.2 Login Page



## 4.3 Registration Page

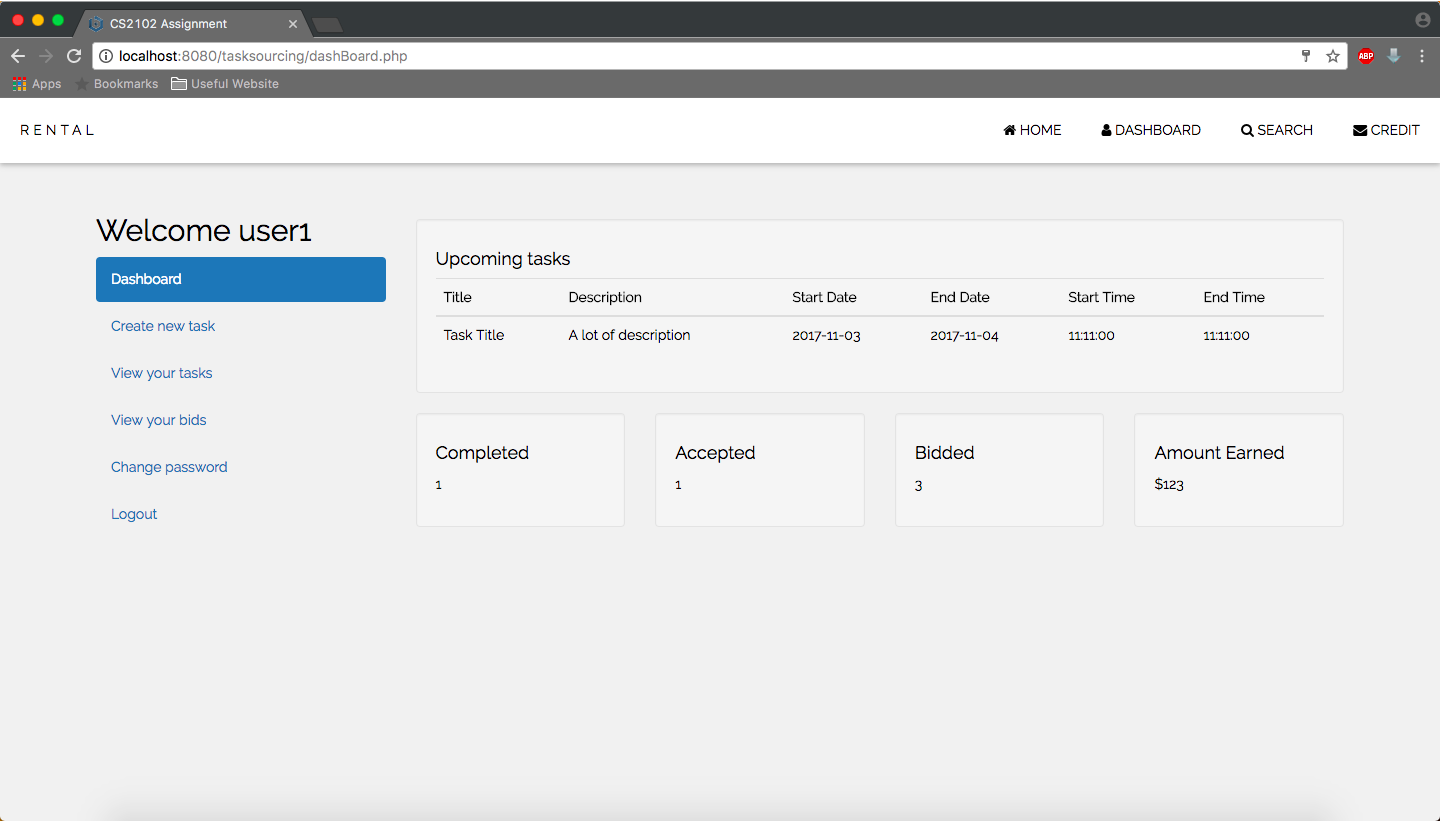


## 4.4 Search Page

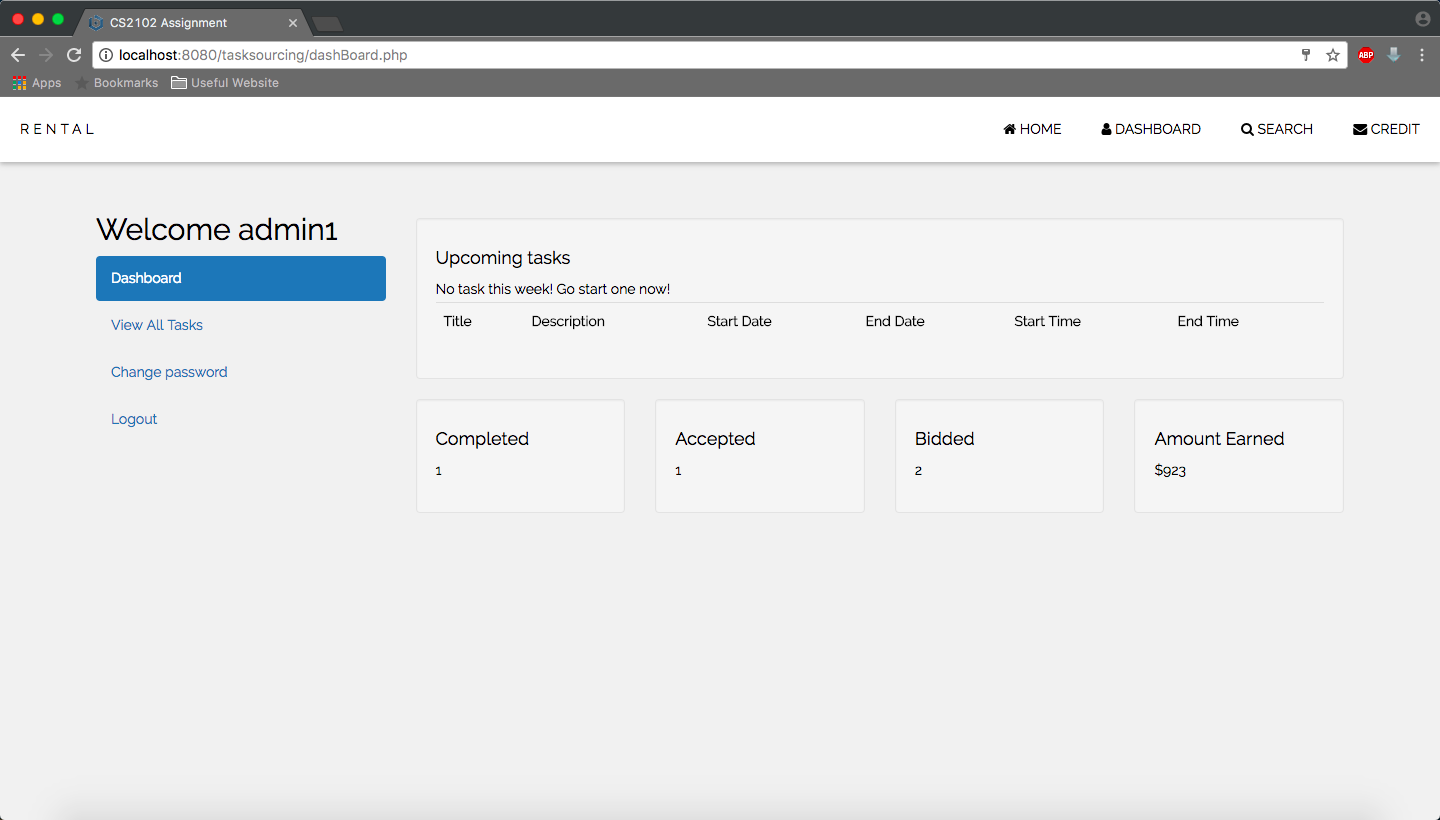


## 4.5 Dashboard

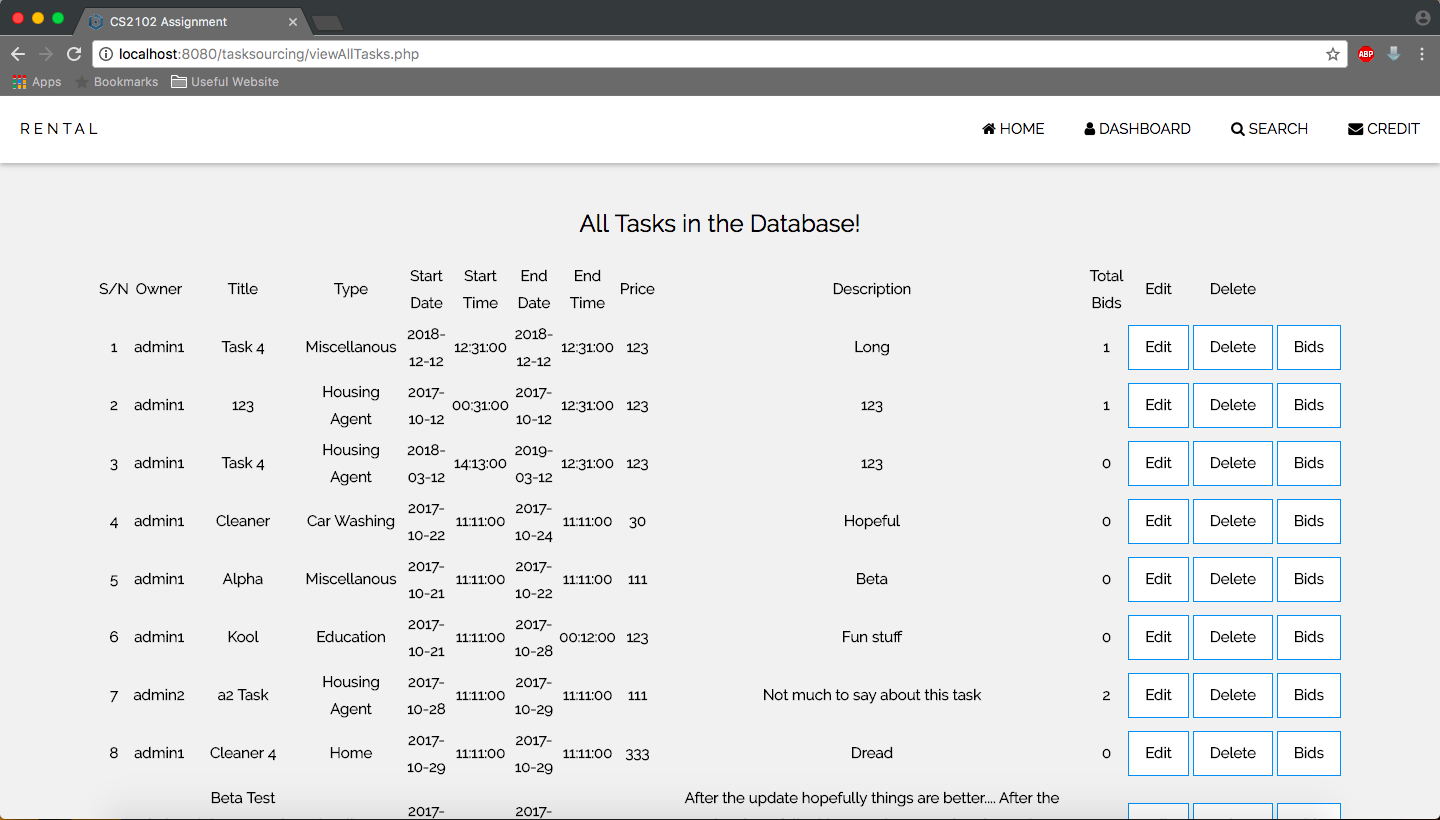
### 4.5.1 User Dashboard



### 4.5.2 Admin Dashboard



## 4.6 Admin View All Tasks



## 4.7 Create Task

