TaskRabbit

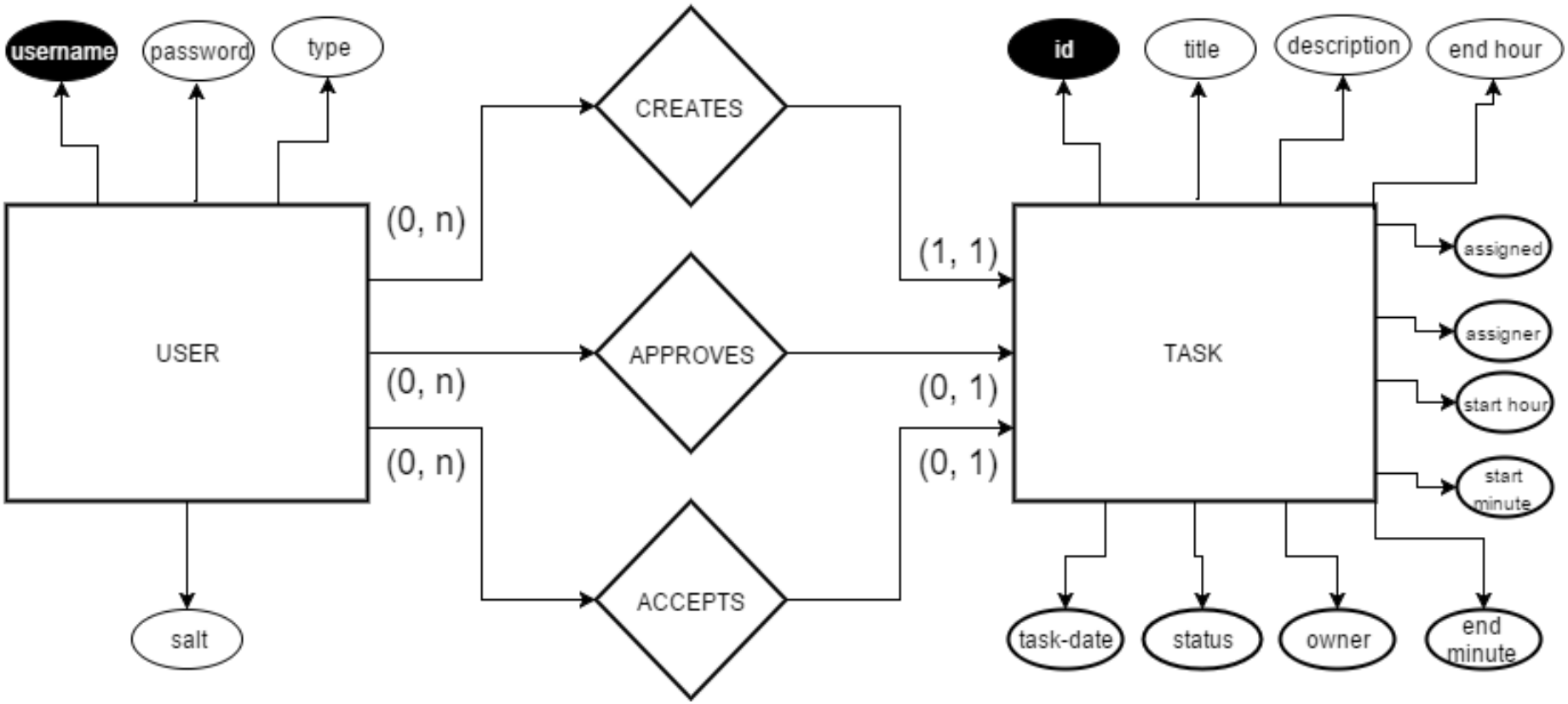
A Web Application

Group 43

Jeris Tanus
Alec Kretch
Michael Sun Zhi Hao
Daniel Korn
Ryan Mackey

Our group was tasked with creating a task sourcing web application (Option A). We completed this goal using the Bitnami Apache web server, a PostgreSQL database management system, and PHP server-side code.

At a basic level, tasks were submitted by regular users, then made available for acceptance by other regular users, with administrative users holding the ability to approve, decline, or delete tasks. The structuring of our database reflects this core-level description, with the addition of metadata (not visible to non-administrative users) necessary for the functional processing of the web application. Please refer to the following ER diagram for a more complete and detailed representation of the inner-workings of our database:


username
password
type
USER
salt
CREATES
APPROVES
ACCEPTS
(0, n)
(0, n)
(0, n)
(1, 1)
(0, 1)
(0, 1)
id
title
description
end hour
assigned
assigner
start hour
start minute
TASK
task-date
status
owner
end minute

SQL DDL code

Tasks

```
CREATE TABLE IF NOT EXISTS tasks
(id SERIAL PRIMARY KEY,
title VARCHAR(40) NOT NULL,
description VARCHAR(254) NOT NULL default '',
task_date date NOT NULL default CURRENT_DATE,
start_hour integer NOT NULL CHECK (start_hour >= 0 AND start_hour < 24),
start_min integer NOT NULL CHECK (start_min >= 0 AND start_min < 60),
end_hour integer NOT NULL CHECK (end_hour >= 0 AND end_hour < 24),
end_min integer NOT NULL CHECK (end_min >= 0 AND end_min < 60),
assigner VARCHAR(254) REFERENCES users,
owner VARCHAR(254) REFERENCES users NOT NULL,
assigned BOOLEAN default FALSE,
status VARCHAR(254) default 'pending' CHECK(status = 'pending' OR status =
'approved' OR status = 'disapproved' OR status = 'completed'));
```

Users

```
CREATE TABLE IF NOT EXISTS users
(username VARCHAR(254) PRIMARY KEY,
password VARCHAR(254),
salt VARCHAR(254),
type VARCHAR(6) CHECK(type='normal' OR type='admin'));
```

Sample SQL code

1)

```
$title = $_POST['title'];
$description = $_POST['description'];
$date = $_POST['date'];
$starttime = $_POST['starttime'];
$fullStartDate = "$date $starttime:00.0";
$endtime = $_POST['endtime'];
$fullEndDate = "$date $endtime:00.0";
$now = time();
$formatFullStartDate = strtotime($fullStartDate);
$formatFullEndDate = strtotime($fullEndDate);

if ($now > $formatFullStartDate)
    {
      $error_message = "Please enter a date that has not already occurred!";
    }
else if ($formatFullStartDate > $formatFullEndDate)
    {
      $error_message = "The task must start before it can end!";
    }
else
    {
      $start_arr = explode(":" , $starttime);
      $end_arr = explode(":" , $endtime);

      $start_hour = intval($start_arr[0]);
      $start_min = intval($start_arr[1]);
      $end_hour = intval($end_arr[0]);
      $end_min = intval($end_arr[1]);

      $sql = "INSERT INTO tasks (title , description , task date ,
      start_hour, start_min, end_hour, end_min, owner) VALUES('$title' ,
      '$description' , '$date', $start_hour, $start_min , $end_hour ,
      $end_min , '$username')";

$result = pg_query($database, $sql);
```

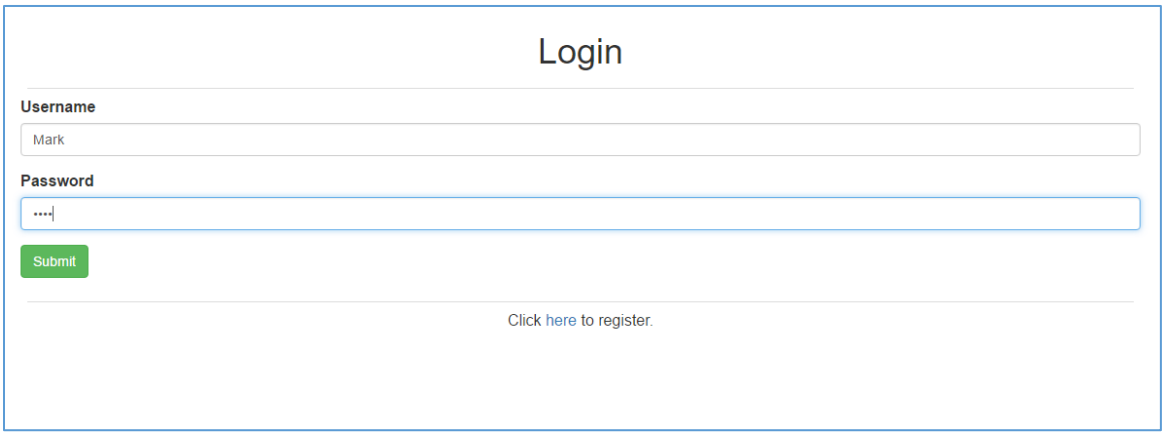
The highlighted section of the above code comprises the SQL query to create a new task in the tasks table. The variables necessary for instantiating a new record arise from the PHP code included above the SQL portion. This allowed users to input their customized values. Finally the last line communicates the query to the database and saves the result in a variable.

2)

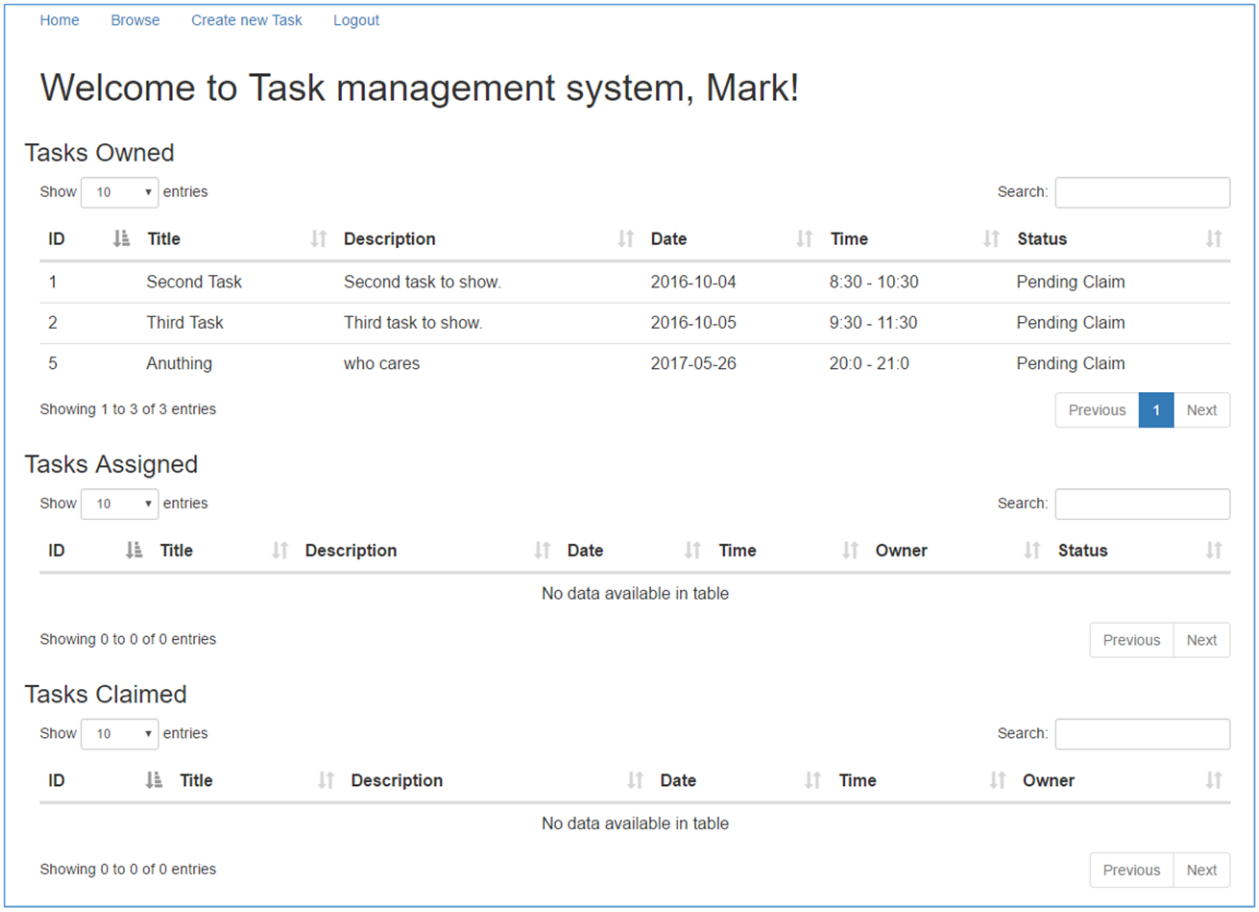
```
$sql="SELECT username FROM users WHERE username='$username' AND type='admin'";
$result = pg_query($database, $sql);
$count = pg_num_rows($result);
$isAdmin = false;
if ($count == 1)
  {
    $isAdmin = true;
  }
```

The code above checks whether the current user is an administrator or not. Again, the SQL query is highlighted. In this case, the user does not enter any custom data; only the username linked to the current session is used in the query. This is followed by PHP code that determines the amount of results from the query, then infers the user's status from this information.

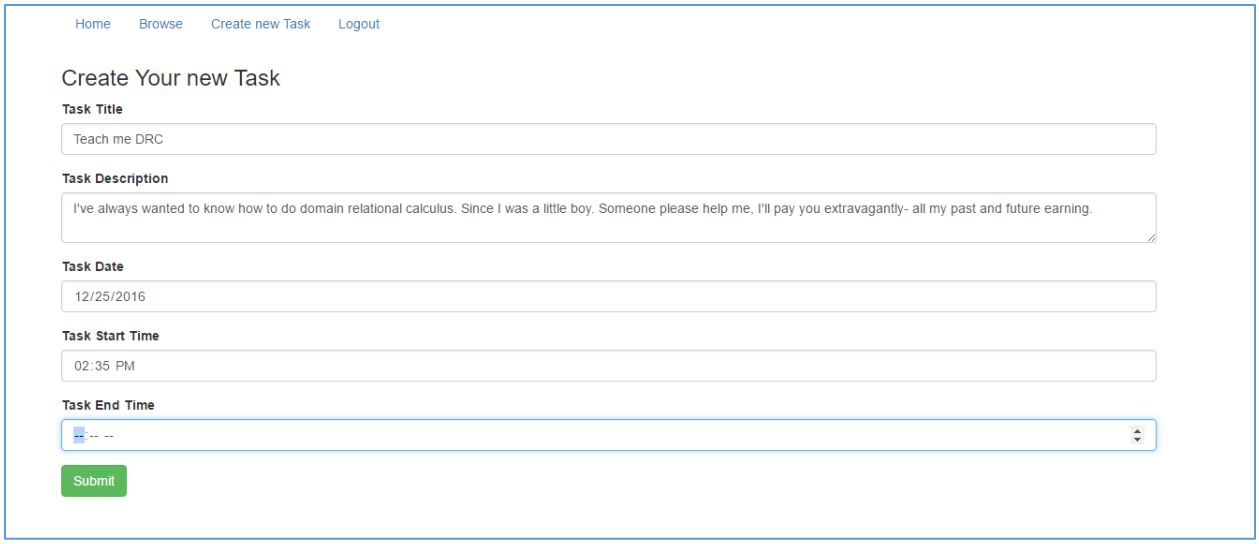
Screenshots



This is the Login screen for our web application. Both administrators and regular users login through this portal. Additionally, the option to register as a new user is offered underneath the entry forms.



This is the user's personal page within the web application. All of the tasks that the user has posted populate the top table, his posted tasks that have been taken by other users populate the middle table, and other's tasks that he has claimed fill the bottom table. The user is able to select how many tasks will be displayed at a given time, and which criteria the sorting will be based upon. Beyond this, the navigation bar at the top offers users the ability to explore other sections of the web application, and logout.



This screenshot shows the task creation page. These forms coincide with the PHP code from the first coding example, included on page 5. The user can enter whatever they like, and the results will be passed to an administrator. If the administrator approves the task, it will be added to the user's homepage, and made available for other users to browse and claim.