**Group 13 CS2102 Term Project**

**Team Members:**

Timoté Vaucher A0153040Y

Sean Zhang A0153658W

Mudit Gupta A0153196B

Marshall Chua A0116410Y

Martin Yim A0153831J

**Project Description:**

For this group project, our team was assigned to complete a TaskRabbit clone, which we named TaskBunny. In the application, users could add a task to be matched with another user, update their posted task, assign themselves to a task to do, and use search functions to look through the tasks. Admins could edit and delete tasks as well as using regular user functions, as well as having access to some basic statistics. Each task is assigned to a task category, where details such as salary, location, dates, and a description are present. In the backend of the application, we used Apache as the web server, while also using Php and Postgres as the server page language and database management systems, respectively. The frontend was done using HTML, CSS, and JavaScript.

Our Schema contains six tables. A Client table handles the user’s log in information. The client table is then used by the “Tasker” table to pair the Client with the Task via the appropriate ID’s. A “Privilege” table also relates to the “Client” table, where the “Privilege” level assigns the user to be either a normal user or an administrator. The “Session\_Token” table is used to clean the sessions when a user logs out. For tasks, the “Task\_Catagory” table records all possible task categories, which is a static list (eg: Gardening, Handiwork, and Miscellaneous). Finally, the “Tasker” table uses the “Client” table and “Task\_Catagory” to create the task entry (eg: Do my groceries) that should be matched with a tasker.

Navigating the application involves a quick sign up or log in process where you are prompted for your first and last name, email, and password. Following the sign up, you are brought to the “My Tasks” page with a table displaying tasks you’ve requested a tasker to complete. From the “My Tasks” page you can access the add task form and edit task form, and access the “Available Tasks” page and “Tasks I’m Helping” page. All three pages contain tables and search/filtering functions for the relevant information. “Available Tasks” has tasks available for you to sign up for, “Tasks I’m Helping” has tasks you have assigned yourself to help with, and “My Tasks” has the tasks you have requested a tasker (user) to help with.

**ER Schema:**

**Relational Schema (SQL):**

CREATE TABLE IF NOT EXISTS client (

id SERIAL PRIMARY KEY,

firstname VARCHAR(32) NOT NULL,

lastname VARCHAR(32) NOT NULL,

email VARCHAR(256) NOT NULL UNIQUE,

password VARCHAR(256) NOT NULL

);

CREATE TABLE IF NOT EXISTS privilege (

client\_id INT REFERENCES client(id) ON DELETE CASCADE,

level VARCHAR(5) CHECK(level = 'user' OR level = 'admin'),

PRIMARY KEY (client\_id)

);

CREATE TABLE IF NOT EXISTS session\_token (

client\_id INT REFERENCES client(id) ON DELETE CASCADE,

token VARCHAR(256),

PRIMARY KEY (client\_id)

);

CREATE TABLE IF NOT EXISTS task\_category (

id SERIAL PRIMARY KEY,

title VARCHAR(64),

description VARCHAR (256)

);

CREATE TABLE IF NOT EXISTS task (

id SERIAL PRIMARY KEY,

creator INT REFERENCES client(id) ON DELETE CASCADE,

category INT REFERENCES task\_category(id) ON DELETE CASCADE,

start\_time DATE,

end\_time DATE,

location VARCHAR(256),

description VARCHAR(256),

salary MONEY,

CHECK(end\_time >= start\_time)

);

CREATE TABLE IF NOT EXISTS tasker (

helper INT REFERENCES client(id) ON DELETE CASCADE,

task\_id INT REFERENCES task(id) ON DELETE CASCADE,

PRIMARY KEY (helper, task\_id)

);

**SQL Code Sample:**

**Web Interface Examples:**