CS2102
Project Team 1
Topic A: Task Sourcing

Lau Kar Rui - A0155936U

Nguyen Thanh Son - A0161299W

Terrence Tan - A0170074M

AY2017/18 Sem 2

Table of Content

Technologies used

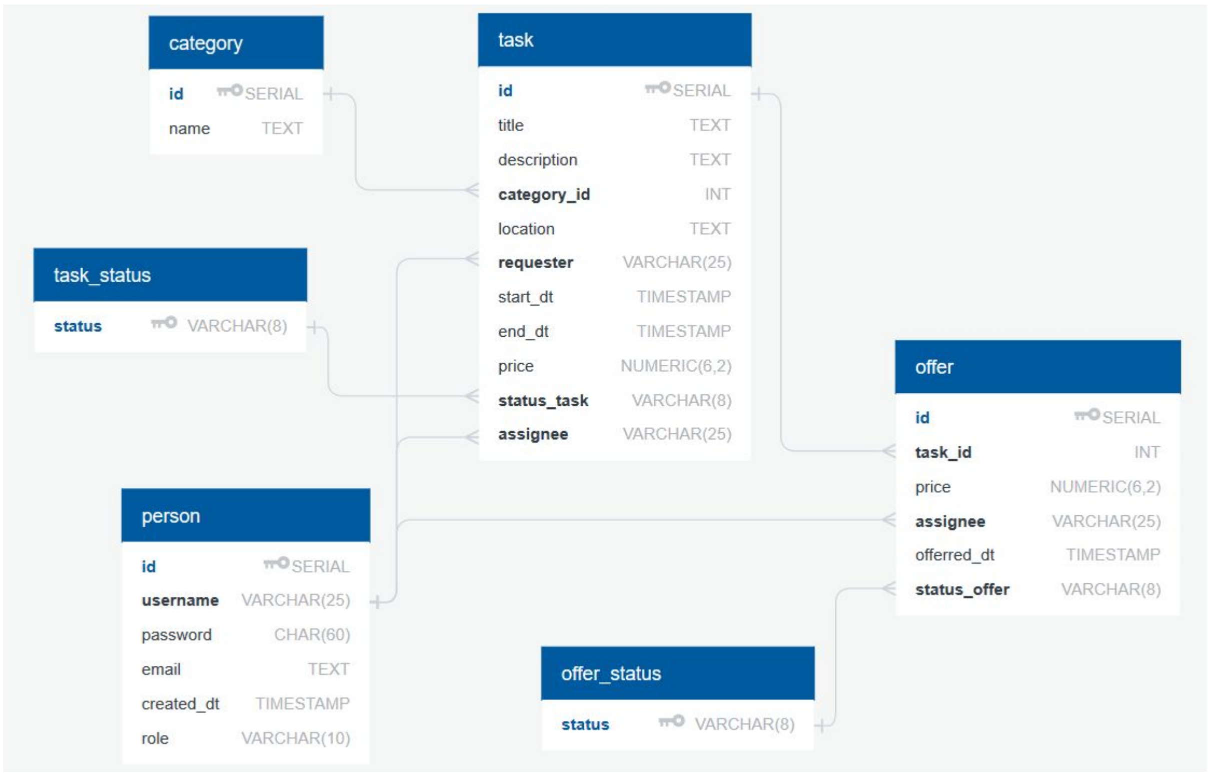
Platform: Node.js - JavaScript runtime built on Chrome's V8 JavaScript engine
Framework: Express.js - a web application framework for Node.js
Authentication: Passport.js, bcrypt.js
Testing Framework: PyTest - a Python framework to run unit tests
Continuous Integration Platform: Semaphore CI
CSS Framework: Bootstrap 3
Database: PostgreSQL v10.3
Languages used:

- JavaScript for Application stack
- SQL for Database stack
- Python for Testing stack

Functionalities

- All users can
  - Create a member account and Login.
  - See other members' profiles, including usernames, emails and created dates.
  - See all tasks, by categories, or by tasks' statuses.
  - Search for tasks:
    - Basic search: tasks having all the words from the search string in their titles and descriptions.
    - Advanced search: tasks by search string, category, location, requester, start date, minimum price, maximum price, task's status and assignee.
- Normal members can
  - Add / Edit / Delete his tasks.
  - Make / Edit / Delete offer for tasks of other members.
  - Accept / Reject offers to his tasks from other members.
- Admin members can:
  - Edit / Delete any task.
  - Delete any user.
  - Delete any offer.
  - See statistics of tasks, offers and users in the admin dashboard.

Database Schema



category	
id	SERIAL
name	TEXT

task	
id	SERIAL
title	TEXT
description	TEXT
category_id	INT
location	TEXT
requester	VARCHAR(25)
start_dt	TIMESTAMP
end_dt	TIMESTAMP
price	NUMERIC(6,2)
status_task	VARCHAR(8)
assignee	VARCHAR(25)

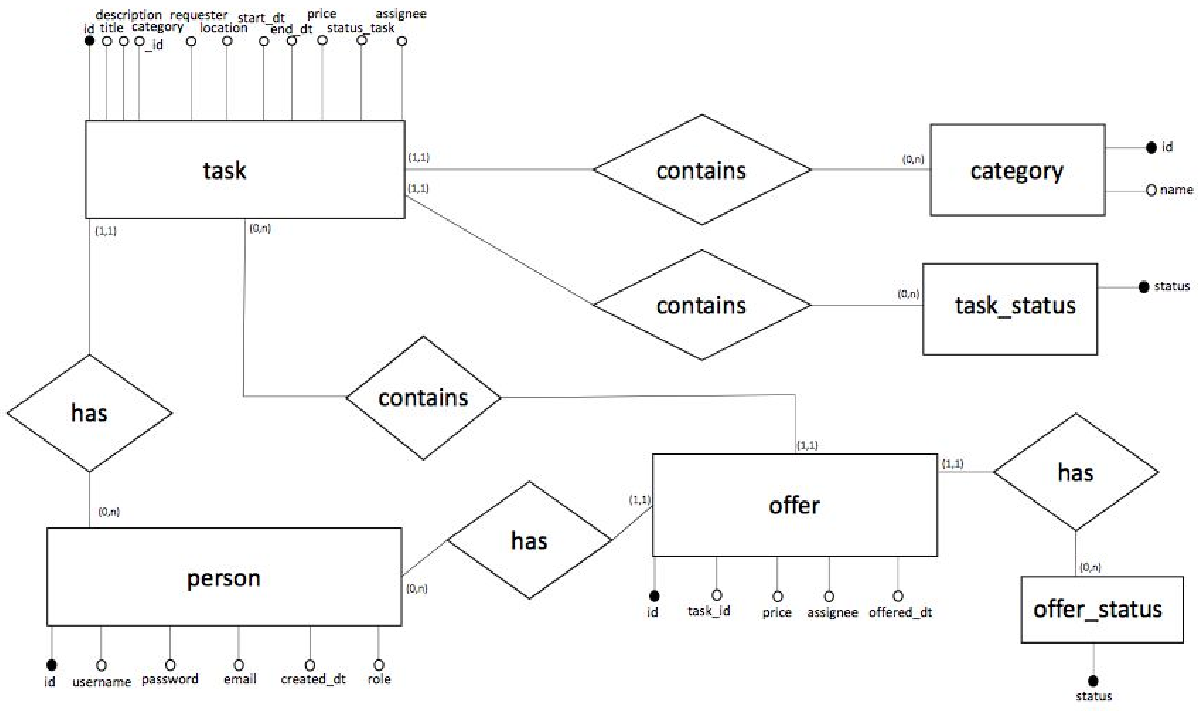
task_status	
status	VARCHAR(8)

offer	
id	SERIAL
task_id	INT
price	NUMERIC(6,2)
assignee	VARCHAR(25)
offerred_dt	TIMESTAMP
status_offer	VARCHAR(8)

person	
id	SERIAL
username	VARCHAR(25)
password	CHAR(60)
email	TEXT
created_dt	TIMESTAMP
role	VARCHAR(10)

offer_status	
status	VARCHAR(8)

ER Diagram



SQL Queries

Relational Schema

Constraints are highlighted in pale orange.

Person schema

```
CREATE TABLE IF NOT EXISTS person (
        id              SERIAL          PRIMARY KEY,
        username        VARCHAR(25)     UNIQUE NOT NULL,
        password        CHAR(60)        NOT NULL,
        email           TEXT            UNIQUE NOT NULL,
        created_dt      TIMESTAMP       NOT NULL,
        role            VARCHAR(10)     DEFAULT 'member' NOT NULL
);
```

Category schema

```
CREATE TABLE IF NOT EXISTS category (
        id              SERIAL          PRIMARY KEY,
        name            TEXT            UNIQUE NOT NULL
);
```

Task_Status schema

```
CREATE TABLE IF NOT EXISTS task_status (
        status          VARCHAR(8)      PRIMARY KEY
);
```

Offer_Status schema

```
CREATE TABLE IF NOT EXISTS offer_status (
        status          VARCHAR(8)      PRIMARY KEY
);
```

Offer schema

```
CREATE TABLE IF NOT EXISTS offer (
        id              SERIAL          PRIMARY KEY,
        task_id         INTEGER         NOT NULL
                        REFERENCES task(id) ON DELETE CASCADE,
        price           NUMERIC(6, 2)   NOT NULL,
        assignee        VARCHAR(25)     NOT NULL
                        REFERENCES person(username) ON DELETE CASCADE,
        offered_dt      TIMESTAMP       NOT NULL,
        status_offer    VARCHAR(8)      DEFAULT 'pending' NOT NULL
                        REFERENCES offer_status(status) ON UPDATE CASCADE,
        UNIQUE (task_id, assignee),
        CHECK (price >= 0 and price < 10000)
);
```

Task schema

```
CREATE TABLE IF NOT EXISTS task (
        id              SERIAL          PRIMARY KEY,
        title           TEXT            NOT NULL,
        description     TEXT            NOT NULL,
        category_id     INTEGER         NOT NULL
                        REFERENCES category(id) ON UPDATE CASCADE,
        location        TEXT            NOT NULL,
        requester       VARCHAR(25)     NOT NULL
                        REFERENCES person(username) ON DELETE CASCADE,
        start_dt        TIMESTAMP       NOT NULL,
        end_dt          TIMESTAMP       NOT NULL,
        price           NUMERIC(6, 2)   NOT NULL,
        status_task     VARCHAR(8)      DEFAULT 'open' NOT NULL
                        REFERENCES task_status(status) ON UPDATE CASCADE,
        assignee        VARCHAR(25)     DEFAULT NULL
                        REFERENCES person(username) ON DELETE SET NULL,
        CHECK (start_dt <= end_dt),
        CHECK (price >= 0 and price < 10000)
);
```

For the full rundown of all the SQL queries used, check out here.

All *non-mutating* DML queries are obtained from *Views* to minimize the possibility of external agents modifying our data. The created views are identical to their origin table with the exception of the view_all_task view, where we also include the category name in order to display the name in the browser application.

Noteworthy SQL DML Code

Insert one task

```
SELECT insert_one_task($1, $2, $3, $4, $5, $6, $7, $8);

CREATE OR REPLACE FUNCTION insert_one_task (
        _title TEXT,
        _description TEXT,
        _category_id INTEGER,
        _location TEXT,
        _requester VARCHAR(25),
        _start_dt TIMESTAMP,
        _end_dt TIMESTAMP,
        _price NUMERIC(6, 2)
)
RETURNS void AS $BODY$
    BEGIN
        INSERT INTO task ( title, description, category_id, location, requester, start_dt,
        end_dt, price )
        VALUES ( _title, _description, _category_id, _location, _requester, _start_dt,
        _end_dt, _price );
    END; $BODY$ LANGUAGE 'plpgsql' VOLATILE COST 100
;
```

Update offer when user changes their offer amount

```
SELECT update_offer_by_assignee_taskid($1, $2, $3, $4);

CREATE OR REPLACE FUNCTION update_offer_by_assignee_taskid (
        _assignee VARCHAR(25),
        _task_id INTEGER,
        _price NUMERIC(6, 2),
        _offered_dt TIMESTAMP
)
RETURNS void AS $BODY$
```

```
    BEGIN
        UPDATE offer
        SET price = _price, offered_dt = _offered_dt, status_offer = 'pending'
        WHERE 1=1
            AND assignee = _assignee AND task_id = _task_id
        ;

        UPDATE task
        SET status_task = 'offered'
        WHERE 1=1
            AND id = _task_id AND status_task IS NOT DISTINCT FROM 'open'
        ;
    END; $BODY$ LANGUAGE 'plpgsql' VOLATILE COST 100
;
```

Get all tasks by category

```
SELECT
        view_all_task.id,
        view_all_task.title,
        view_all_task.description,
        view_all_task.category_id,
        view_all_task.category_name,
        view_all_task.location,
        view_all_task.requester,
        view_all_task.start_dt,
        view_all_task.end_dt,
        view_all_task.price,
        view_all_task.status_task,
        view_all_task.assignee
FROM view_all_task
WHERE 1=1
    AND view_all_task.category_id = $1
ORDER BY view_all_task.id DESC ;
```

Advanced SQL Functions in the project

Update offer’s status and task’s status upon rejecting offer

Whenever a task’s owner rejects an offer, this function changes the offer’s status to ‘rejected’ and updates the task’s status to ‘open’ if all offers for the task are ’rejected’.

```
CREATE OR REPLACE FUNCTION update_task_upon_rejecting_offer_by_task_id (
        _task_id         INTEGER,
        _offer_id        INTEGER
)
RETURNS void AS $BODY$
    BEGIN
        UPDATE offer
        SET status_offer = 'rejected'
        WHERE 1=1
            AND id = _offer_id
        ;
        UPDATE task
        SET status_task = 'open'
        WHERE 1=1
            AND id = _task_id
            AND NOT EXISTS (
                SELECT 1
                FROM offer
                WHERE 1=1
                    AND task_id = _task_id
                    AND status_offer IS DISTINCT FROM 'rejected'
            )
        ;
    END; $BODY$ LANGUAGE 'plpgsql' VOLATILE COST 100
;
```

Advanced Search method

The following function is used as an advanced search method for tasks of the project. The user indicates the attributes of tasks he is looking for and the application will return all the tasks that satisfy the conditions.

For example, the user searches for location ‘nuS’. The application will return all the tasks whose location have the term ‘NUS’, including ’NUS CELC’, ‘NUS’, ‘nus soc’, etc.

As an implementation of *pg-node*, the library used for for bidirectional communication with the database and the server to avoid SQL injection, all the inputs of any function call cannot be *null*; therefore, all parameters of the function when called by the application are at least an empty string.

Function `get_matching_percent` is an user-defined function that calculates the Levenshtein distance between two strings, for the *Advanced Search Method*. It uses the *fuzzystrmatch* extension - a module provides several functions to determine similarities and distance between strings. The function can be viewed from here.

```
CREATE OR REPLACE FUNCTION get_tasks_with_advanced_search (
        _search_string   TEXT             DEFAULT NULL,
        _category_id     TEXT             DEFAULT NULL,
        _location        TEXT             DEFAULT NULL,
        _requester       TEXT             DEFAULT NULL,
        _start_dt        TEXT             DEFAULT NULL,
        _min_price       TEXT             DEFAULT NULL,
        _max_price       TEXT             DEFAULT NULL,
        _status_task     TEXT             DEFAULT NULL,
        _assignee        TEXT             DEFAULT NULL
)
RETURNS SETOF task AS $BODY$
    BEGIN
        if _search_string = '' THEN _search_string = NULL; END if;
        if _category_id = '' THEN _category_id = NULL; END if;
        if _location = '' THEN _location = NULL; END if;
        if _requester = '' THEN _requester = NULL; END if;
        if _start_dt = '' THEN _start_dt = NULL; END if;
        if _min_price = '' THEN _min_price = NULL; END if;
        if _max_price = '' THEN _max_price = NULL; END if;
        if _status_task = '' THEN _status_task = NULL; END if;
        if _assignee = '' THEN _assignee = NULL; END if;

        RETURN QUERY
        SELECT
            task.id,
            task.title,
            task.description,
            task.category_id,
            task.location,
            task.requester,
            task.start_dt,
            task.end_dt,
            task.price,
            task.status_task,
            task.assignee
        FROM
            task,
            (
                SELECT unnest(string_to_array(coalesce(_search_string, ' '), ' ')) AS word
            ) AS sub
        WHERE 1=1
            AND (
                -- If the title or description contains all the words
                task.title ILIKE '%' || sub.word || '%'
                OR task.description ILIKE '%' || sub.word || '%'
            )
            AND task.category_id = coalesce(CAST(_category_id AS NUMERIC(6, 2)), task.category_id)
```

```
                AND (
                    -- If location matches more than 40%
                    get_matching_percent(task.location, _location) >= 0.4
                    OR task.location ILIKE '%' || coalesce(_location, '') || '%'
                )
                -- If requester matches more than 80%
                AND get_matching_percent(task.requester, _requester) >= 0.8
                -- Same status_task, if NULL
                AND task.start_dt::DATE = coalesce(_start_dt, to_char(task.start_dt, 'YYYY-MM-DD'))::DATE
                -- If price >= min_price
                AND task.price >= coalesce(CAST(_min_price AS NUMERIC(6, 2)), 0)
                -- If price <= max_price
                AND task.price <= coalesce(CAST(_max_price AS NUMERIC(6, 2)), 9999.99)
                -- Same status_task, if NULL
                AND task.status_task = coalesce(_status_task, task.status_task)
                -- If assignee matches more than 80%
                AND get_matching_percent(coalesce(task.assignee, ''), _assignee) >= 0.8

            GROUP BY task.id
            HAVING count(task.id) IS NOT DISTINCT
                FROM array_length(string_to_array(coalesce(_search_string, ' '), ' '), 1)
            ORDER BY task.id DESC
            ;
    END; $BODY$ LANGUAGE 'plpgsql' VOLATILE COST 100
;
```

Implementation of Non-trivial Constraints

Refer to the Relational Schema section for all constraints (highlighted in pale orange).

Making sure constraints are satisfied when adding a task or offer

We have a check constraint in both the `task` and `offer` table, namely

```
CHECK (price >= 0 and price < 10000)
```

to make sure the starting price of the task is not below $0. We also cap the maximum price to be below $10,000 arbitrarily as we do not expect any task to be above $10,000, and also to prevent any large numbers.

A second check in just the `task` table

```
CHECK (start_dt <= end_dt),
```

makes sure that the end date is not before the start date when adding a task.

Prevent duplicate usernames and emails of users

To ensure that every username or email is unique, we have a “unique” constraint on our `person` table, namely

```
username    VARCHAR(25)     UNIQUE NOT NULL,
email       TEXT            UNIQUE NOT NULL
```

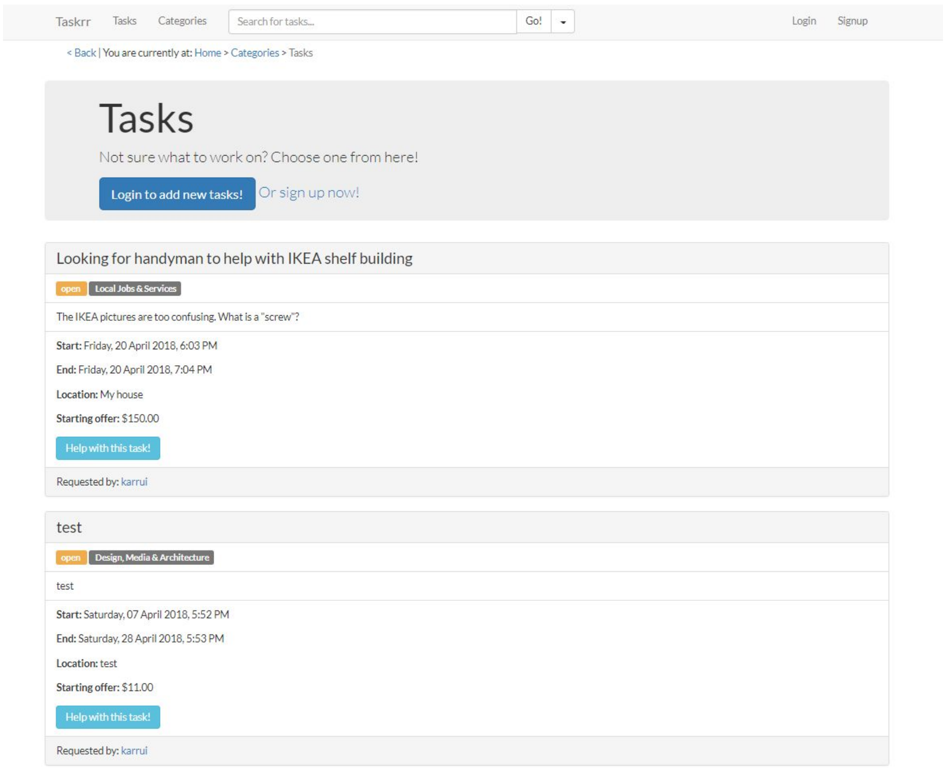
Ensuring each user only gives one offer to a task

We want to prevent a user making multiple offers to the same task and thus added a constraint to the `offer` table to ensure the (task_id, assignee) pair is unique.

```
UNIQUE (task_id, assignee)
```

Screenshots

All tasks (and the search bar)



Taskrr Tasks Categories Search for tasks... Go! Login Signup

< Back | You are currently at: Home > Categories > Tasks

Tasks

Not sure what to work on? Choose one from here!

Login to add new tasks! Or sign up now!

Looking for handyman to help with IKEA shelf building

open Local Jobs & Services

The IKEA pictures are too confusing. What is a "screw"?

Start: Friday, 20 April 2018, 6:03 PM

End: Friday, 20 April 2018, 7:04 PM

Location: My house

Starting offer: $150.00

Help with this task!

Requested by: karrui

test

open Design, Media & Architecture

test

Start: Saturday, 07 April 2018, 5:52 PM

End: Saturday, 28 April 2018, 5:53 PM

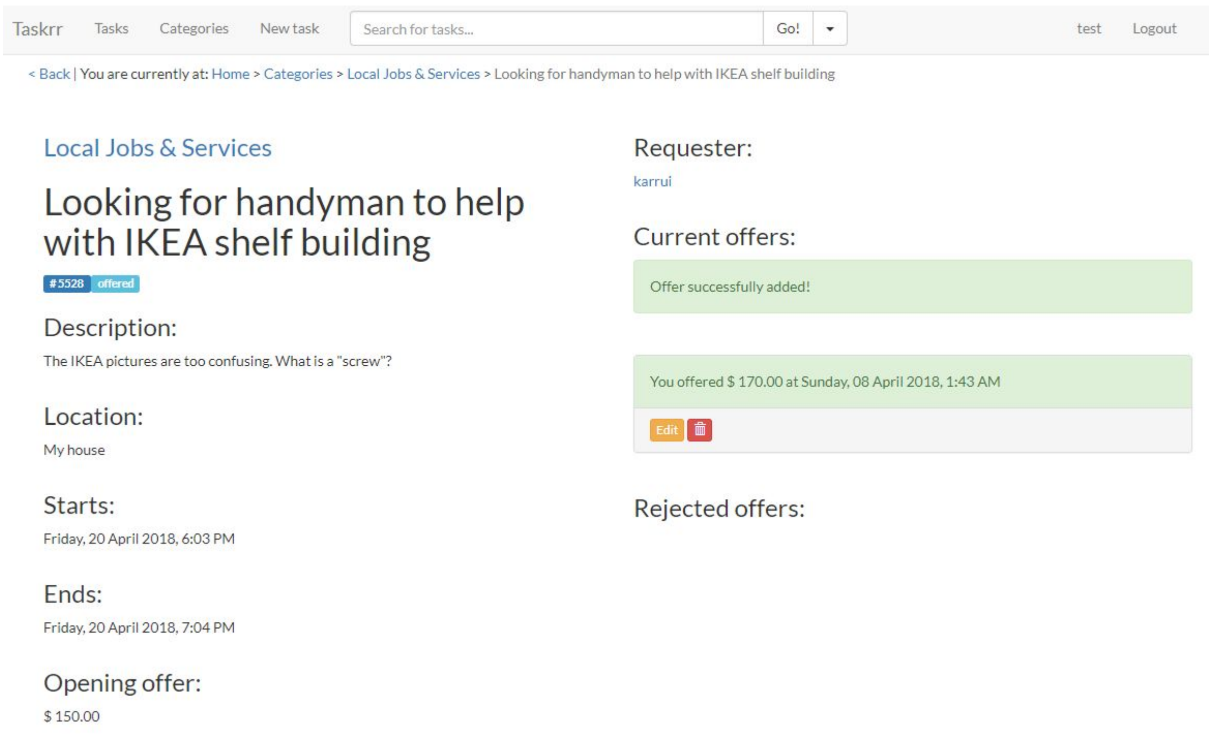
Location: test

Starting offer: $11.00

Help with this task!

Requested by: karrui

Task detail / offer for task



Taskrr Tasks Categories New task Search for tasks... Go! test Logout

< Back | You are currently at: Home > Categories > Local Jobs & Services > Looking for handyman to help with IKEA shelf building

Local Jobs & Services

Looking for handyman to help with IKEA shelf building

#5528 offered

Description:

The IKEA pictures are too confusing. What is a "screw"?

Location:

My house

Starts:

Friday, 20 April 2018, 6:03 PM

Ends:

Friday, 20 April 2018, 7:04 PM

Opening offer:

$ 150.00

Requester:

karrui

Current offers:

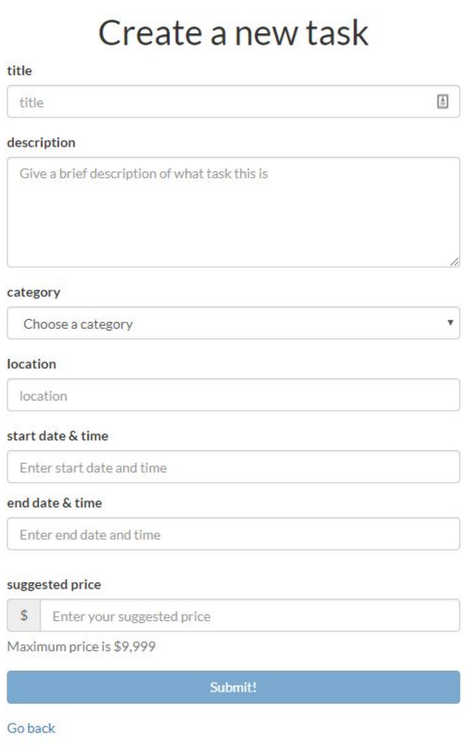
Offer successfully added!

You offered $ 170.00 at Sunday, 08 April 2018, 1:43 AM

Edit

Rejected offers:

New task/ Edit task



Create a new task

title

title

description

Give a brief description of what task this is

category

Choose a category

location

location

start date & time

Enter start date and time

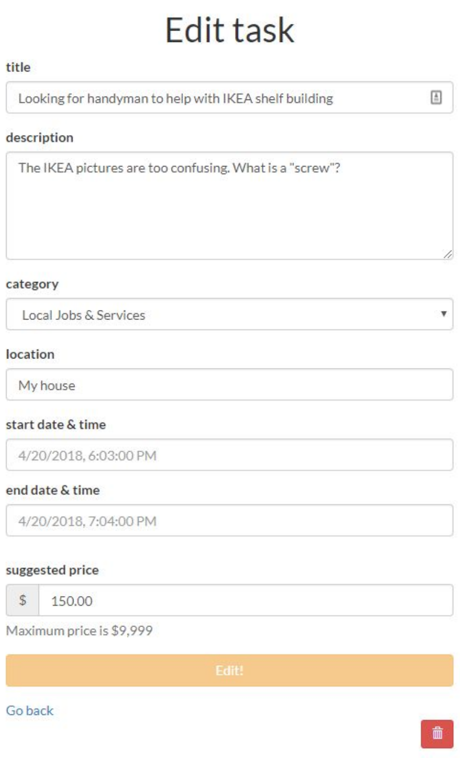
end date & time

Enter end date and time

suggested price

$ Enter your suggested price

Maximum price is $9,999



Edit task

title

Looking for handyman to help with IKEA shelf building

description

The IKEA pictures are too confusing. What is a "screw"?

category

Local Jobs & Services

location

My house

start date & time

4/20/2018, 6:03:00 PM

end date & time

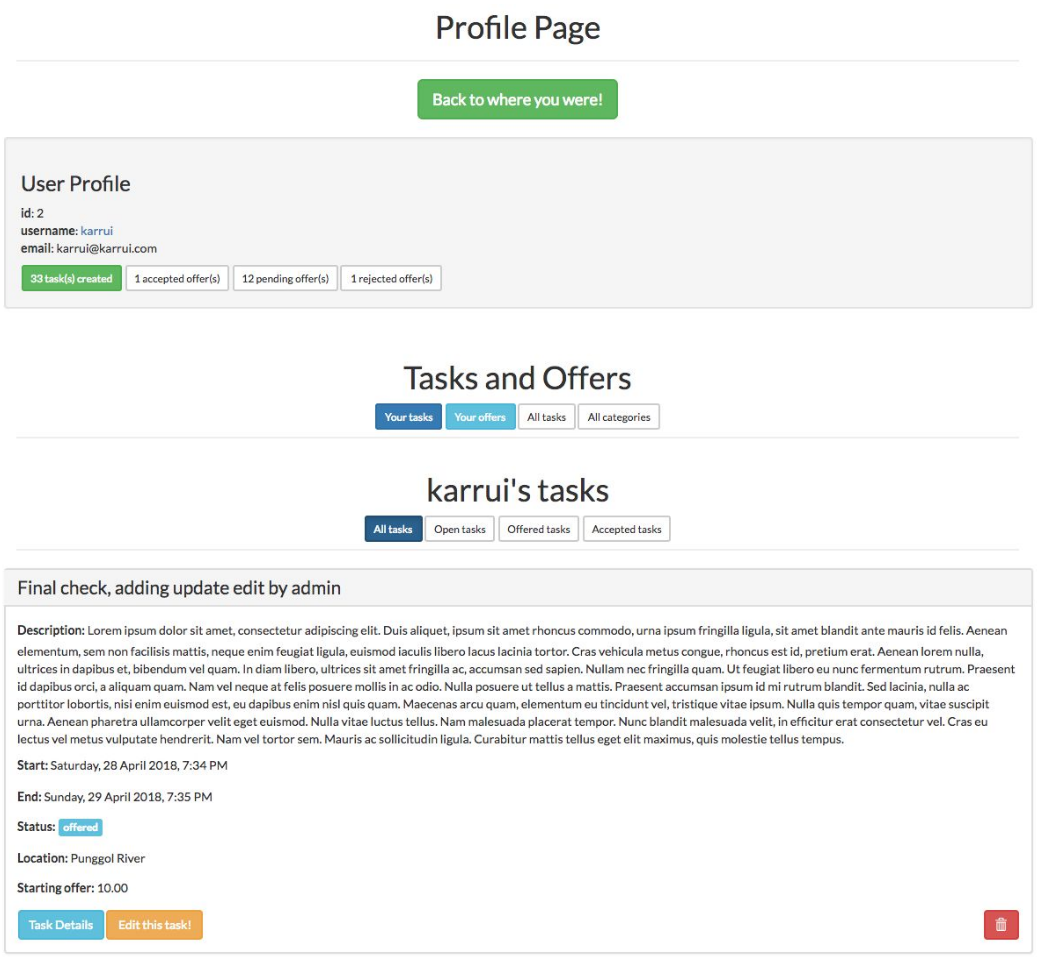
4/20/2018, 7:04:00 PM

suggested price

$ 150.00

Maximum price is $9,999

Profile



Profile Page

User Profile

id: 2
username: karrui
email: karrui@karrui.com

Tasks and Offers

karrui's tasks

Final check, adding update edit by admin

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis aliquet, ipsum sit amet rhoncus commodo, urna ipsum fringilla ligula, sit amet blandit ante mauris id felis. Aenean elementum, sem non facilisis mattis, neque enim feugiat ligula, euismod iaculis libero lacus lacinia tortor. Cras vehicula metus congue, rhoncus est id, pretium erat. Aenean lorem nulla, ultrices in dapibus et, bibendum vel quam. In diam libero, ultrices sit amet fringilla ac, accumsan sed sapien. Nullam nec fringilla quam. Ut feugiat libero eu nunc fermentum rutrum. Praesent id dapibus orci, a aliquam quam. Nam vel neque at felis posuere mollis in ac odio. Nulla posuere ut tellus a mattis. Praesent accumsan ipsum id mi rutrum blandit. Sed lacinia, nulla ac porttitor lobortis, nisi enim euismod est, eu dapibus enim nisl quis quam. Maecenas arcu quam, elementum eu tincidunt vel, tristique vitae ipsum. Nulla quis tempor quam, vitae suscipit urna. Aenean pharetra ullamcorper velit eget euismod. Nulla vitae luctus tellus. Nam malesuada placerat tempor. Nunc blandit malesuada velit, in efficitur erat consectetur vel. Cras eu lectus vel metus vulputate hendrerit. Nam vel tortor sem. Mauris ac sollicitudin ligula. Curabitur mattis tellus eget elit maximus, quis molestie tellus tempus.

Start: Saturday, 28 April 2018, 7:34 PM

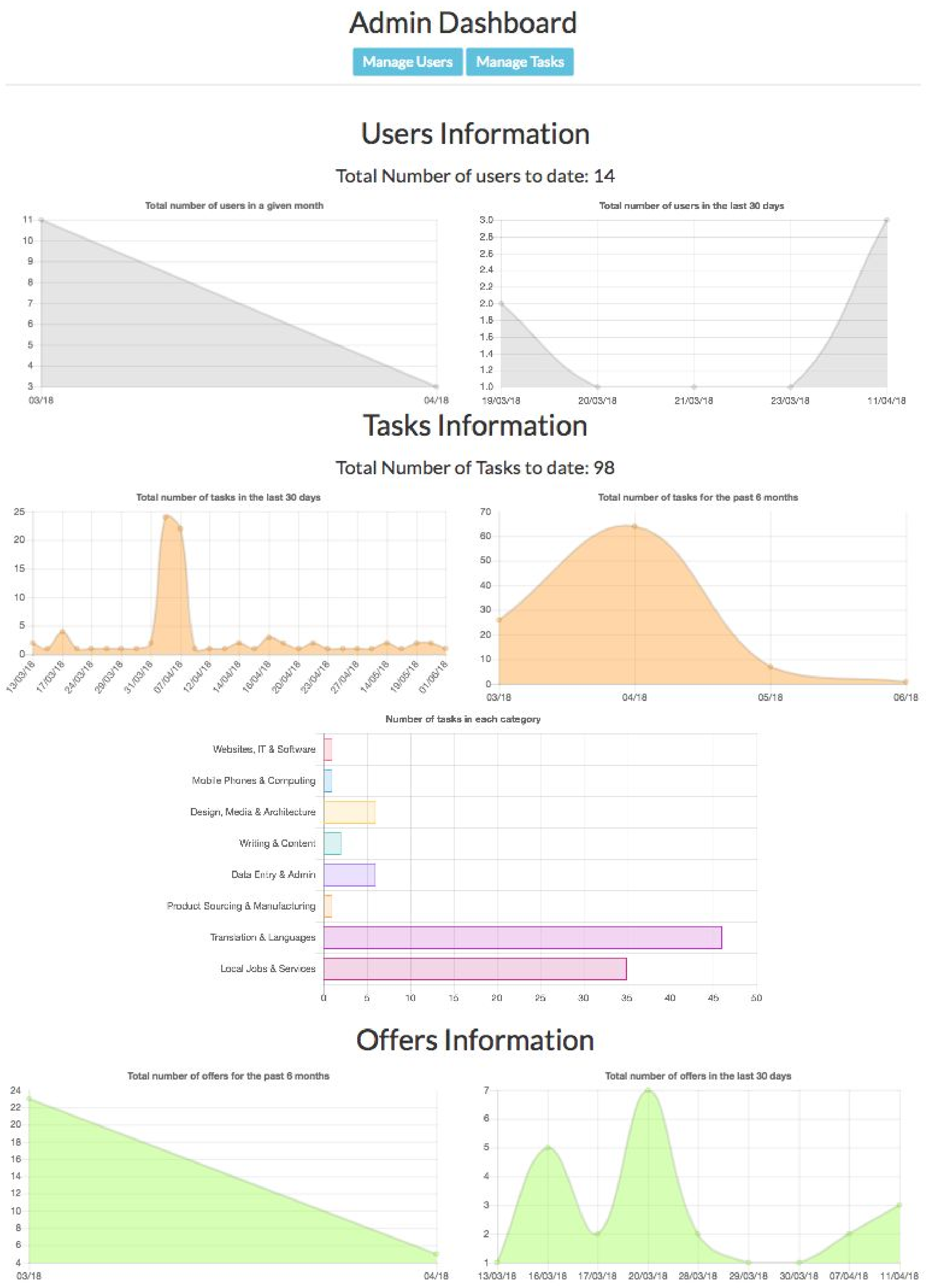
End: Sunday, 29 April 2018, 7:35 PM

Status: offered

Location: Punggol River

Starting offer: 10.00

Admin Page - Statistics


Admin Dashboard

Users Information

Total Number of users to date: 14

Tasks Information

Total Number of Tasks to date: 98

Offers Information
Manage Users
Manage Tasks
Total number of users in a given month
03/18
04/18
Total number of users in the last 30 days
19/03/18
20/03/18
21/03/18
23/03/18
11/04/18
Total number of tasks in the last 30 days
Total number of tasks for the past 6 months
03/18
04/18
05/18
06/18
Number of tasks in each category
Websites, IT & Software
Mobile Phones & Computing
Design, Media & Architecture
Writing & Content
Data Entry & Admin
Product Sourcing & Manufacturing
Translation & Languages
Local Jobs & Services
Total number of offers for the past 6 months
03/18
04/18
Total number of offers in the last 30 days
13/03/18
16/03/18
17/03/18
20/03/18
28/03/18
29/03/18
30/03/18
07/04/18
11/04/18