CS 387 Project: NetWorks

Pandurang Deore (200050096)
Sarthak Mittal (200050129)
Shikhar Agrawal (200070076)
Mayank Jain (20d070050)

Contents

1 Summary

1.1 Description

We plan to build a clone of the LinkedIn Web application. We intend to incorporate most features of an online job hub. We would be implementing the same in a PERN stack framework (PostgreSQL, Express, React, Node.js).

1.2 Flow of the System

Navigating through the website, we will create a signup/login interface, user dashboard (with feed), and profile page, and maintain data of connections among people. There will be two types of users, recruiters and applicants. We will add some data by default for the demonstration and will accept new data (profiles, posts, job postings, connection requests) from logged-in users. If time permits, we will also create a chat interface in the app.
For the details of the components, we plan to have multiple fields while creating a profile, the option to add media to posts, and details of job postings (recruiter, company, location, etc.). If time permits, we will also try to visualize the connections using a graph.

2 Overview

2.1 Domain

This application will be part of online professional networks and employment portals. We intend to incorporate most features of an online job hub.

2.2 Intended Use

The application will support:

- building a portfolio and connecting with people of similar interests
- staying up to date with the professional network and opportunities
- broadcasting, searching and applying for jobs online
- real-time chatting with others in your professional network[1]

[1] if time permits

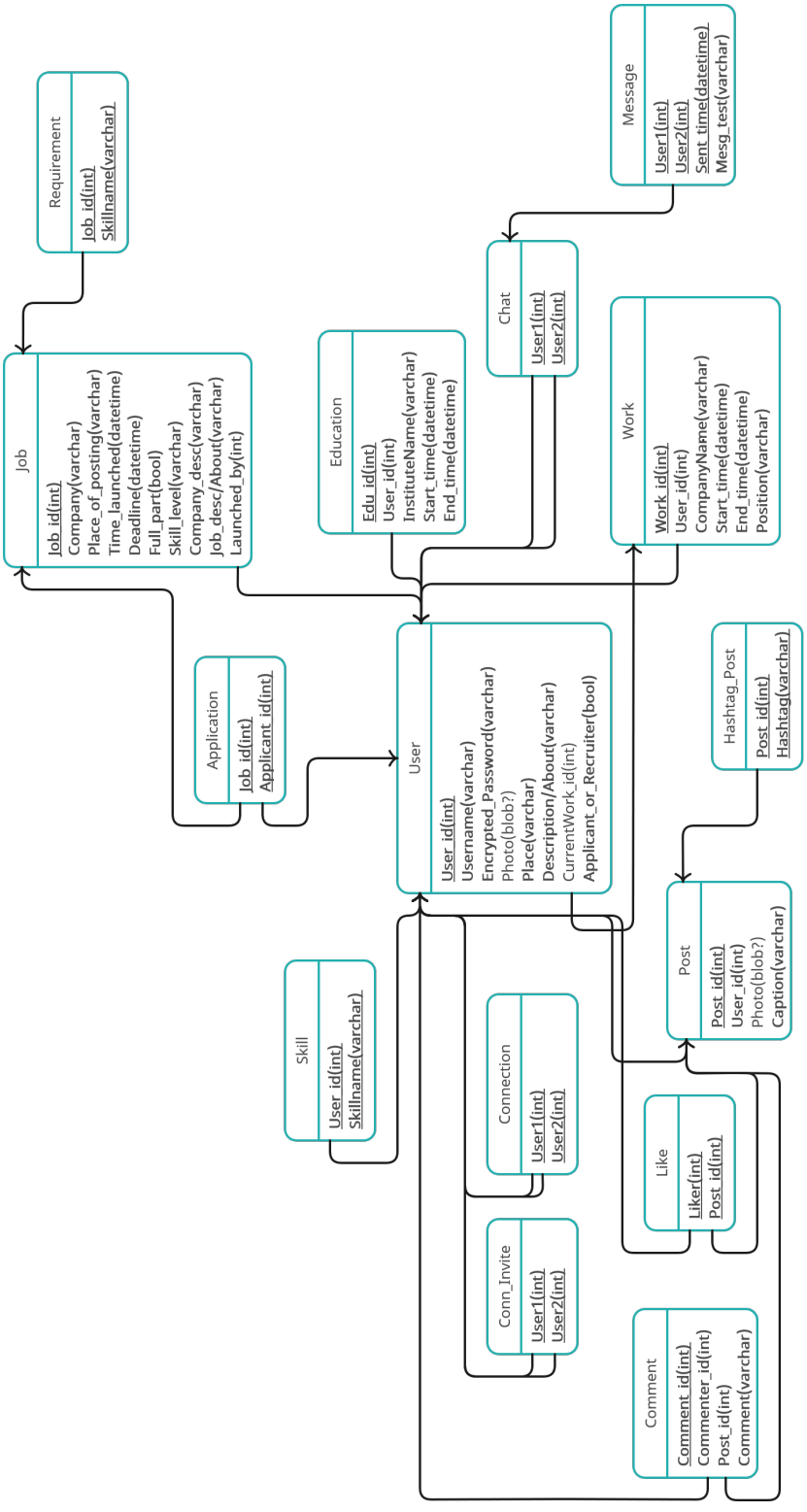

Figure 1: Entity-Relation Diagram

3 Users

There would be broadly two types of users: recruiters and applicants.

3.1 Role of a Recruiter

- looking for applicants to hire for the job postings
- gives the posting details such as company, location, duration, etc.
- waits for applicants to apply and then contacts them accordingly

3.2 Role of an Applicant

- looking for job postings to apply to and applies accordingly
- gives the profile details such as name, email, qualification, etc.
- waits for recruiters to accept their application

4 Database

4.1 Notation

- underlined attributes are part of the primary key for the entity
- **bold** attributes cannot be null
- → arrows indicate the foreign key constraints
- **data types** are indicated in (.) brackets (for some attributes, we are not yet sure of the data type, like blob for images)

4.2 Relation Schema

```
CREATE DATABASE networks;

CREATE TABLE users(
    user_id INT NOT NULL,
    username VARCHAR(64) NOT NULL,
    encrypted_password VARCHAR(80) NOT NULL,
    photo bytea,
    place VARCHAR(64) NOT NULL,
    description VARCHAR(256) NOT NULL,
    currentwork_id INT,
    applicant_or_recruiter BOOLEAN NOT NULL,
```

```
    primary key (user_id)
);

CREATE TABLE application(
    job_id INT NOT NULL,
    applicant_id INT NOT NULL,
    primary key (job_id, applicant_id)
);

CREATE TABLE skill(
    user_id INT NOT NULL,
    skillname VARCHAR(64) NOT NULL,
    primary key (user_id, skillname)
);

CREATE TABLE connection(
    user1 INT NOT NULL,
    user2 INT NOT NULL,
    primary key (user1, user2)
);

CREATE TABLE conn_invite(
    user1 INT NOT NULL,
    user2 INT NOT NULL,
    primary key (user1, user2)
);

CREATE TABLE likes(
    liker INT NOT NULL,
    post_id INT NOT NULL,
    primary key (liker, post_id)
);

CREATE TABLE comment(
    comment_id INT NOT NULL,
    commenter_id INT NOT NULL,
    post_id INT NOT NULL,
    comment VARCHAR(64) NOT NULL,
    primary key(comment_id)
);

CREATE TABLE post(
    post_id INT NOT NULL,
    user_id INT NOT NULL,
    photo bytea,
    caption VARCHAR(64) NOT NULL,
```

```
    primary key (post_id)
);

CREATE TABLE hashtag_post(
    post_id INT NOT NULL,
    hashtag VARCHAR(32) NOT NULL,
    primary key (post_id, hashtag)
);

CREATE TABLE work(
    work_id INT NOT NULL,
    user_id INT NOT NULL,
    companyname VARCHAR(64) NOT NULL,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP NOT NULL,
    position VARCHAR(64) NOT NULL,
    primary key (work_id)
);

CREATE TABLE message(
    user1 INT NOT NULL,
    user2 INT NOT NULL,
    sent_time TIMESTAMP NOT NULL,
    mesg_text VARCHAR(80) NOT NULL,
    primary key (user1, user2, mesg_text)
);

CREATE TABLE chat(
    user1 INT NOT NULL,
    user2 INT NOT NULL,
    primary key (user1, user2)
);

CREATE TABLE education(
    edu_id INT NOT NULL,
    user_id INT NOT NULL,
    institutename VARCHAR(64) NOT NULL,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP NOT NULL,
    primary key (edu_id)
);

CREATE TABLE requirement(
    job_id INT NOT NULL,
    skillname VARCHAR(64) NOT NULL,
    primary key (job_id, skillname)
```

```
);

CREATE TABLE job(
    job_id INT NOT NULL,
    company VARCHAR(64) NOT NULL,
    place_of_posting VARCHAR(64) NOT NULL,
    time_launched TIMESTAMP NOT NULL,
    deadline TIMESTAMP NOT NULL,
    full_part BOOLEAN NOT NULL,
    skill_level VARCHAR(64) NOT NULL,
    company_desc VARCHAR(64) NOT NULL,
    job_desc VARCHAR(64) NOT NULL,
    launched_by INT NOT NULL,
    primary key (job_id)
);

ALTER TABLE users
    ADD CONSTRAINT fk_currentwork_id FOREIGN KEY (currentwork_id) REFERENCES
        work(work_id)
    ON DELETE SET NULL ON UPDATE CASCADE;

ALTER TABLE application
    ADD CONSTRAINT fk_job_id FOREIGN KEY (job_id) REFERENCES job(job_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE application
    ADD CONSTRAINT fk_applicant_id FOREIGN KEY (applicant_id) REFERENCES
        users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE skill
    ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE conn_invite
    ADD CONSTRAINT fk_user1 FOREIGN KEY (user1) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE conn_invite
    ADD CONSTRAINT fk_user2 FOREIGN KEY (user2) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE connection
    ADD CONSTRAINT fk_user1 FOREIGN KEY (user1) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE connection
    ADD CONSTRAINT fk_user2 FOREIGN KEY (user2) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE comment
    ADD CONSTRAINT fk_commenter_id FOREIGN KEY (commenter_id) REFERENCES
        users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE comment
    ADD CONSTRAINT fk_post_id FOREIGN KEY (post_id) REFERENCES post(post_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE likes
    ADD CONSTRAINT fk_liker_id FOREIGN KEY (liker_id) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE likes
    ADD CONSTRAINT fk_post_id FOREIGN KEY (post_id) REFERENCES post(post_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE post
    ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE hashtag_post
    ADD CONSTRAINT fk_post_id FOREIGN KEY (post_id) REFERENCES post(post_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE chat
    ADD CONSTRAINT fk_user1 FOREIGN KEY (user1) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE chat
    ADD CONSTRAINT fk_user2 FOREIGN KEY (user2) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE work
    ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE message
    ADD CONSTRAINT fk_users FOREIGN KEY(user1, user2) REFERENCES chat(user1,
        user2)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE education
    ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE requirement
    ADD CONSTRAINT fk_job_id FOREIGN KEY (job_id) REFERENCES job(job_id)
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE job
    ADD CONSTRAINT fk_launched_by FOREIGN KEY (launched_by) REFERENCES
        users(user_id)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

5 Application

5.1 Backend API

We will provide endpoints for the following functionality:

- login/signup
- dashboard
- profile
- connections
- job board
- chat[2]

5.2 User Interface

The web application will provide interfaces for the following functionality:

5.2.1 Login

- Form (email/username)
- Button (login/signup)

[2] if time permits

5.2.2 Toolbar

- Home (dashboard/feed)
- Network (connections)
- Jobs (applying/posting)
- Chat (messaging)[3]
- Profile page (information)
- Sign out (logout page)

5.2.3 Dashboard

- Toolbar
- Make a post
- Feed
- Overview of profile
- Events[3]

5.2.4 Profile

- Toolbar
- Photo (option to change[3])
- Name
- Place
- Description (about)
- Current work
- Previous work
- Connections
- Education
- Skills

[3]if time permits

5.2.5 Network

- Toolbar
- List of new connection invites
- Current connections

5.2.6 Jobs

- Toolbar
- Applicant (filter by: company name, place, skill, list of jobs)

5.2.7 Chat

- Toolbar
- List of chats
- Current chat

5.2.8 Post object

- Name of poster
- Small description of poster
- Photo of poster
- Caption
- Hashtags[4]
- Photo/Video[4] for description
- Like
- Comment

[4]if time permits

5.2.9 Posting interface ("Make a post")

- Username
- Photo
- Description
- Caption
- Photo/Video[5] for description
- Hashtags[5]
- Post button

5.2.10 Commenting interface

- Post
- Add comment
- List of comments

5.2.11 Comment object

- Name of the commenter
- Comment

5.2.12 Profile object

- Toolbar
- Name
- Photo
- Description
- Place
- Position
- Send connection invite
- Send chat message (if connected)[5]

[5] if time permits

- Connections
- Past work experience
- Education
- Skills

5.2.13 Job object

- Name of post
- Company
- Place of posting
- Time of posting
- Deadline to apply

5.2.14 Expanded job object

- Name of post
- Company
- Place of posting
- Apply button
- Time of posting
- Deadline to apply
- Number of people applied
- Fulltime/parttime
- Skill level needed
- Skill set needed
- Company description
- Details/About

5.2.15 Chat object

[6]

- Name of "chatee"
- List of chat messages
- Write a new message (no image)
- Send button

5.2.16 Chat message object

[6]

- Sender
- Text

5.3 Security

We will use Express for session management.

[6] if time permits