Huskies Jobs Platform

Making the job search a walk in the park!

By: Nicholas Wong, Peisong Yuan, Xuanhao Wu

Table of contents

Problem & Motivation

Problem

- Students match jobs to major, GPA, skills, preferences, and locations
- Employers need to analyze application informations.

Motivation

- We wanted to understand how NUworks/LinkedIn actually work
- Understand how these platforms actually match users → positions

Goals

We aim to reconstruct a “mini NUworks/LinkedIn” from a database perspective, learning how to:

- Support complex filtering with data structures.
- Connect Users, skills, and positions.
- Implement automatic validation and alerts using triggers/stored procedures.

Entities

Users (Students)

- Stores student info: major, GPA, class year, contact, preference

Company

- Employer profiles: industry, size, HQ, website

Positions

- Job posting details: category, required major, skills

Application

- Records each student’s job application

Skills

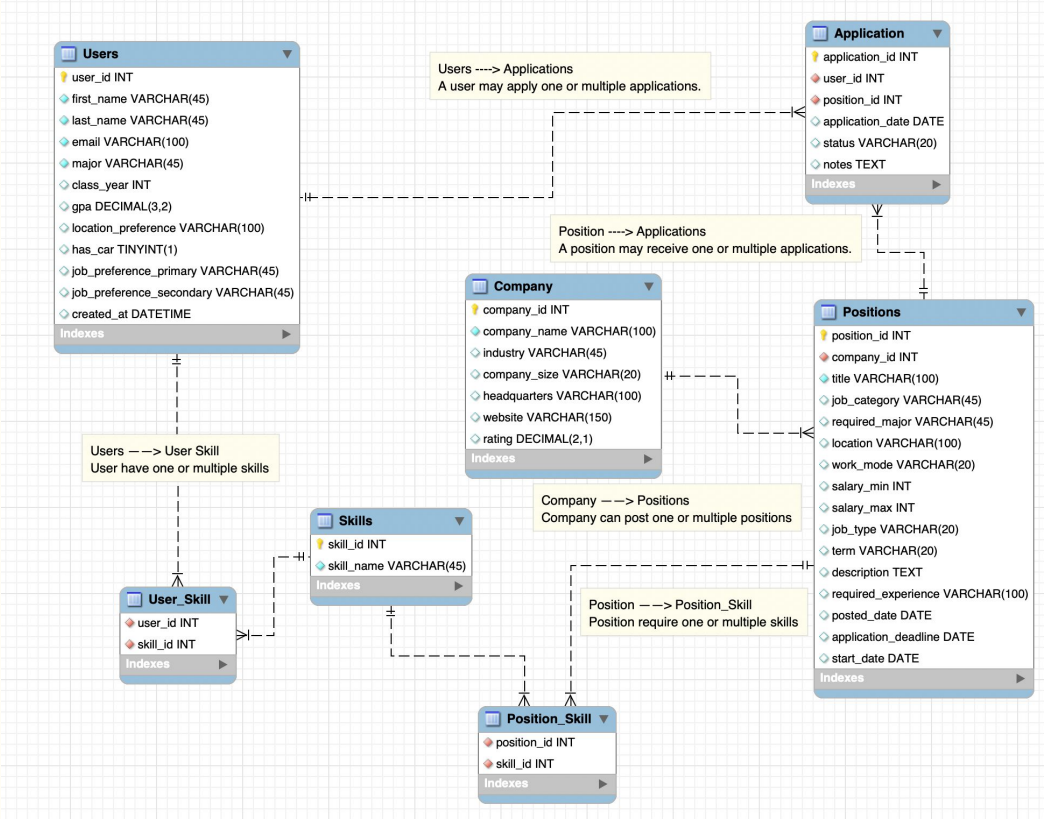
- Standardized list of skills

User_Skill

Position_Skill

Database Design

- Users ↔ Applications
  - one-to-many
- Positions ↔ Application
  - one-to-many
- Positions ↔ Skills
  - many-to-many
  - via Position_Skill
- Users ↔ Skills
  - many-to-many
  - via User_Skill
- Company ↔ Positions
  - 1-to-many



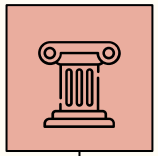
Generating Data For User Cases

Generation

Used Faker + Random to generate fake data.

Seeding

Seeded the generators to ensure replicability and predictability.



Structure

Categorical fields randomly assigned values from a preset array. Simple fields populated with Faker.

Assumptions

All users are assumed to be job-seeking students. Application dates fall within a 6 month window.

```
print("Inserting users")
user_ids = []
for i in range(ROWS):
    sql = """INSERT INTO Users (first_name,
            gpa, location_preference, has_c
            job_preference_secondary, creat
            VALUES (%s, %s, %s, %s, %s, %s,

    values = (
        fake.first_name(),
        fake.last_name(),
        fake.email(),
        random.choice(majors),
        random.randint(2024, 2028),
        round(random.uniform(2.5, 4.0), 2),
        fake.state_abbr(),
        random.choice([True, False]),
        random.choice(job_categories),
        random.choice(job_categories),
        fake.date_time_between(start_date='-2y', end_date='now')
    )
    cur.execute(sql, values)
    user_ids.append(cur.lastrowid)
db.commit()
print("Users inserted")
```

```
Inserting users
Users inserted
```

```
majors = ['Computer Science', 'Data Science', 'Business Administration', 'Marketing',
          'Finance', 'Mechanical Engineering', 'Electrical Engineering', 'Biology',
          'Psychology', 'Economics', 'Information Systems', 'Accounting']
industries = ['Technology', 'Finance', 'Healthcare', 'Retail', 'Manufacturing',
              'Consulting', 'Entertainment', 'Education', 'Real Estate', 'Energy']

job_categories = ['Software Development', 'Data Analysis', 'Marketing', 'Sales',
                  'Finance', 'Operations', 'Human Resources', 'Customer Service',
                  'Engineering', 'Product Management', 'Supply Chain', 'Research']

skills_list = ['Python', 'Java', 'SQL', 'JavaScript', 'React', 'Excel', 'PowerPoint',
               'Communication', 'Leadership', 'Project Management', 'Data Analysis',
               'Machine Learning', 'AWS', 'Docker', 'Agile', 'Git', 'C++', 'R',
               'Tableau', 'Salesforce', 'Adobe Creative Suite', 'Problem Solving']

work_modes = ['Remote', 'Hybrid', 'On-site']
job_types = ['Full-time', 'Part-time', 'Internship']
terms = ['Summer', 'Fall', 'Spring', 'Year-round']
company_sizes = ['1-50', '51-200', '201-500', '501-1000', '1000+']
statuses = ['Applied', 'Under Review', 'Interview', 'Rejected', 'Accepted']
print("Inserting into Users")
```

```
fake = Faker()
Faker.seed(32)
random.seed(32)
ROWS = 200
```

Randomized Values

Each field is randomized based on a seeded generator.

Seeding

Seeded generators to maintain replicability.

Categorical Data

Predefined categorical data examples to populate fields.

Student Use Case 1: Job Search

```sql
-- User Case 1 (Easy)

-- Jack is seeking a co-op in marketing or supply chain
-- and is open to roles anywhere in Massachusetts.

SELECT p.position_id, p.title, p.location, p.required_experience, p.work_mode, c.company_name, p.job_category
FROM Positions p
JOIN Company c
    USING (company_id)
WHERE p.location LIKE "%MA%"
AND p.job_category IN ('Marketing', 'Supply Chain');
```

	position_id	title	location	required_experience	work_mode	company_name	job_category
0	111	Chiropractor	MA	2 years	Hybrid	Harris, Bell and Yu	Marketing

Student Use Case 2: Application Validation

```sql
-- User Case 3 (Hard)

-- When a student selects a position and clicks "Apply," the system should verify that the student satisfies the major requirements,
-- and that they have not already applied. If all conditions are met, the procedure should insert a new application record; otherwise,
-- it returns a clear message explaining why the application cannot be submitted.
DELIMITER //

CREATE PROCEDURE Apply_For_Position
(
    IN user_id_param INT,
    IN position_id_param INT
)
BEGIN
    DECLARE major_var VARCHAR(45);
    DECLARE required_major_var VARCHAR(45);
    DECLARE message VARCHAR(255);

    -- select relevant values into variables
    SELECT major
    INTO major_var
    FROM Users
    WHERE user_id = user_id_param;

    SELECT required_major
    INTO required_major_var
    FROM Positions
    WHERE position_id = position_id_param;

    -- check major
    IF major_var <> required_major_var THEN
    SET message = CONCAT('This position required major is ', required_major
            SIGNAL SQLSTATE 'HY000'
                SET MESSAGE_TEXT = message;
    END IF;

    INSERT INTO Application (user_id, position_id, application_date, status)
    VALUES (user_id_param, position_id_param, CURDATE(), 'Submitted');

END //

DELIMITER ;
```

```python
try: cur.execute("CALL Apply_For_Position(4, 198)")
except Exception as e: print(e)

cur.execute("CALL Apply_For_Position(3, 198)")
cur.execute("SELECT * FROM Application ORDER BY application_id DESC LIMIT 1")
```

Error message

```
1644 (HY000): This position required major is Accounting
```

	application_id	user_id	position_id	application_date	status	notes
0	5908	3	198	2025-11-30	Submitted	None

```sql
-- Company Use Case 1 (Easy)

-- DEF Company wants to know the number of applicants to a position.

SELECT COUNT(*) as num_applications
FROM Positions p
JOIN Application a ON p.position_id = a.position_id
WHERE p.position_id = 10;
```

num_applications
21

```sql
-- Company Use Case 2 (Medium)

-- Medium: XYZ Company wants to know the GPA (DESC order) of students applying to their open positions.

SELECT u.gpa, u.first_name, u.last_name
FROM Users u
JOIN (
    SELECT a.user_id
    FROM Positions p
    JOIN Application a ON p.position_id = a.position_id
    WHERE p.company_id = 10
) d
ON u.user_id = d.user_id
ORDER BY u.gpa DESC;
```

	gpa	first_name	last_name
0	3.96	Brent	Johnson
1	3.93	Tracy	Kennedy
2	3.86	Shannon	Wall
3	3.84	Steven	Miller
4	3.84	Christopher	Miller
5	3.81	Jessica	Mcclure
6	3.79	Austin	Ortiz
7	3.60	Samantha	West
8	3.58	Breanna	Webb
9	3.53	Jacob	Cantu
10	3.44	Melissa	Castro
11	3.43	Lisa	Wheeler
12	3.33	Shannon	Mayer
13	3.32	Rhonda	Miller
14	3.25	Megan	Yu
15	3.08	Melissa	Jefferson
16	2.75	Nicole	Shaw
17	2.75	Evan	Sanders
18	2.73	Karen	Patterson
19	2.66	Jason	Hanson
20	2.52	Erin	Richardson

Company Use Case: Analytics

Automated Notifications

1.

```sql
-- Company Use Case 3 (Hard)

-- ABC Company wants to create an application position for Biology majors. This notifies Sean,
-- who is a Biology major looking for a co-op.

DELIMITER //

CREATE TRIGGER Notify_Matching_Majors
AFTER INSERT ON Positions
FOR EACH ROW
BEGIN
    DECLARE company_name_var VARCHAR(100);

    SELECT company_name INTO company_name_var
    FROM Company
    WHERE company_id = NEW.company_id;

    INSERT INTO Application (user_id, position_id, application_date, status, notes)
    SELECT
        u.user_id,
        NEW.position_id,
        CURDATE(),
        'Notified',
        CONCAT(u.first_name, '! A new position is available: ', NEW.title, ' at ', company_name_var)
    FROM Users u
    WHERE u.major = NEW.required_major;
END //

DELIMITER ;
```

2.

```sql
ification trigger

INSERT INTO Positions (company_id, title, job_category, required_major,
                       location, work_mode, salary_min, salary_max, job_type, term, description,
                       required_experience, posted_date, application_deadline, start_date)
VALUES (
    5,
    'Biology Research Assistant',
    'Research',
    'Biology',
    'MA',
    'On-site',
    20,
    25,
    'Co-op',
    'Spring 2025',
    'Some odd lowkey occulty research opportunity in molecular biology',
    'None required',
    CURDATE(),
    DATE_ADD(CURDATE(), INTERVAL 30 DAY),
    '2026-01-15'
);
```

3.

```sql
cur.execute("""
    SELECT u.first_name, u.last_name, u.major, a.notes
    FROM Application a
    JOIN Users u ON a.user_id = u.user_id
    WHERE a.position_id = %s
""", (position_id,))
```

4.

	first_name	last_name	major	notes
0	Erik	Smith	Biology	Erik! A new position is available: Biology Res...
1	Maureen	Roberts	Biology	Maureen! A new position is available: Biology ...
2	Tiffany	Hamilton	Biology	Tiffany! A new position is available: Biology ...
3	Meagan	Lewis	Biology	Meagan! A new position is available: Biology R...
4	Rhonda	Miller	Biology	Rhonda! A new position is available: Biology R...
5	Daniel	Phillips	Biology	Daniel! A new position is available: Biology R...
6	Greg	Robinson	Biology	Greg! A new position is available: Biology Res...
7	Tony	Jacobs	Biology	Tony! A new position is available: Biology Res...
8	Hannah	Horn	Biology	Hannah! A new position is available: Biology R...
9	Christina	Velazquez	Biology	Christina! A new position is available: Biolog...
10	Larry	Ramos	Biology	Larry! A new position is available: Biology Re...
11	John	Forbes	Biology	John! A new position is available: Biology Res...
12	Kathleen	Andrews	Biology	Kathleen! A new position is available: Biology...
13	Kelly	Hurst	Biology	Kelly! A new position is available: Biology Re...
14	Alexandra	Mcdaniel	Biology	Alexandra! A new position is available: Biolog...

Insights & Key Takeaways

- SQL queries enable both student-focused and employer-focused analytics.
- Matching logic depends heavily on clean relationships between users, skills, positions, and applications.
- Automation can greatly enhances usability and realism.
- Synthetic data generation (Faker) allows full testing without privacy concerns.

Thanks!

Do you have any questions?

References

- Faraglia, D. (n.d.). *Welcome to Faker's documentation!* Welcome to Faker's documentation! - Faker 38.2.0 documentation. https://faker.readthedocs.io/en/master/
- Python Software Foundation. (n.d.). *Random - generate pseudo-random numbers*. Python documentation. https://docs.python.org/3/library/random.html
- Northeastern University. (n.d.). NUworks. https://nuworks.northeastern.edu
- LinkedIn Corporation. (n.d.). *Jobs on LinkedIn*. https://www.linkedin.com/jobs
- Introducing How You Match on LinkedIn Jobs. (2018). Linkedin.com. https://www.linkedin.com/blog/member/career/introducing-how-you-match-on-linkedin-jobs