



Bilkent University

---

Department of Computer Engineering

# CareerBridge

*Project short-name: CB (Group Number: 11)*

## Design Report

Berk Çakar - 22003021

Atak Talay Yücel - 21901636

Kutay Tire - 22001787

Yiğit Yalın - 22002178

Instructor: Özgür Ulusoy

Teaching Assistant: Zülal Bingöl

Design Report

April 19, 2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Database Systems course CS353.

# **Contents**

<b>CareerBridge</b>	<b>1</b>
<b>1 Revised E/R Model</b>	<b>5</b>
<b>2 Relation Schemas</b>	<b>6</b>
User	6
AppUser	6
Admin	6
SystemReport	7
Connection	7
Application	8
ApplicationReview	8
Thread	8
Appointment	9
Message	9
Notification	10
Post	10
Comment	11
JobAdvertisement	11
JobAdvertisementResponse	12
Interview	12
Profile	13
LanguageProficiency	13
Award	14
Project	14
Certification	15
Publication	15
TestScore	16
Skill	16
SkillInJobAdvertisement	17
Experience	17
EducationalExperience	18
Degree	18
DegreeInJobAdvertisement	18
School	19
WorkExperience	19
Company	20
VoluntaryExperience	20
NonProfitOrganization	21
SkillAssessor	21

SkillAssessment	21
CareerExpert	22
Mentorship	22
<b>3 Advanced Database Components</b>	<b>23</b>
3.1 Views	23
3.1.1 User Login View	23
3.1.2 Get User Role View	23
3.2 Triggers	23
3.2.1 Skill Assessment Notification Trigger	23
3.2.2 Comment Notification Trigger	23
3.2.3 Application Notification Trigger	24
3.2.4 Connection Request Notification Trigger	25
3.2.5 Connection Response Notification Trigger	25
3.2.6 Add User to AppUser Trigger	26
3.3 Assertions	26
3.3.1 Check if Each User Has at Most One Master Skill	26
3.3.2 Check if Number of Mentees of Each CareerExpert Does Not Exceed Mentee Limit	26
3.3.3 Check if the Receiver and Creator of a Thread are not the Same Person	27
3.4 Procedures	27
3.4.1 Procedure for Increasing the Application Count of an Advertisement by One	27
3.4.2 Procedure for Increasing the View Count of an Advertisement by One	27
<b>4 User Interface Design and SQL Statements</b>	<b>28</b>
4.1 Login Page	28
4.2 Register Page	29
4.3 Policy Page	30
4.4 Forgot Password Page	31
4.5 Browse Ads Page	32
4.6 Browse Ads Page with Filtering Options	33
4.7 Sample Ad Description	38
4.8 Ad Application Pages	40
4.8.1 Personal Information	40
4.8.2 Biography	41
4.8.3 Experience	42
4.8.4 Education	43
4.8.5 Voluntary Work	44
4.8.6 Projects	45
4.8.7 Certifications	46

4.8.8 Awards	47
4.8.9 Test Scores	48
4.8.10 Languages	49
4.8.12 Upload Resume	51
4.8.13 Uploaded Resume	52
4.9 Applied Ads Page	57
4.10 Published Ads Page	59
4.11 Candidates (Applicants) of an Ad Page	60
4.12 CareerBridge Specific Pages	62
4.12.1 Profile Page	62
4.12.2 Messages Page (Request for Skill Assessment)	67
4.12.3 Messages Page (Request for Mentorship & Appointment)	70
4.12.4 Apply For a Role Page	72
4.12.5 Send a Connection Request	73
4.12.6 Accept or Reject a Connection Request	74
4.12.7 User Feed	75
4.12.8 Admin Panel	78

# Design Report

Project name: CareerBridge

## 1 Revised E/R Model

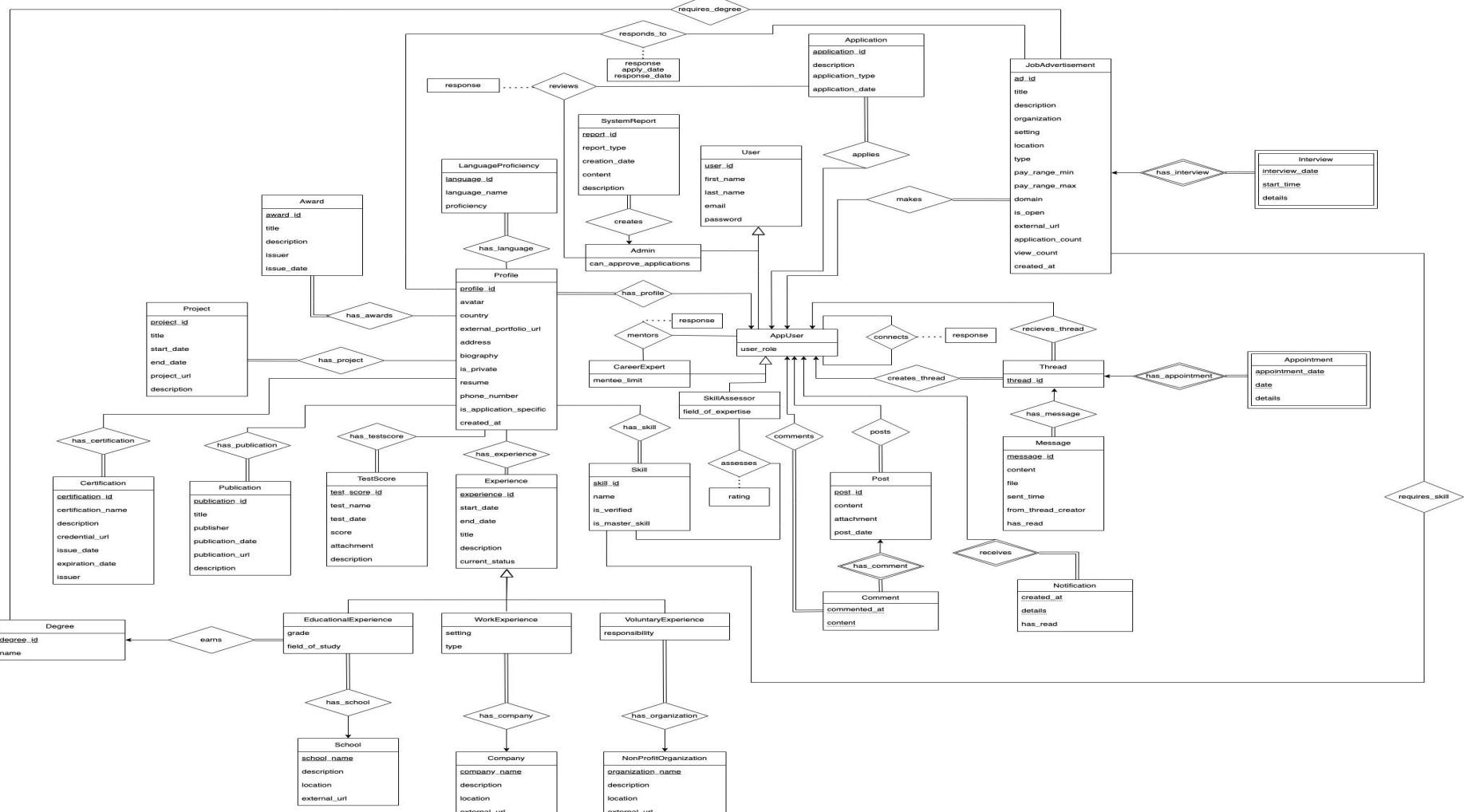


Figure 1: Revised Entity-Relationship Model of the CareerBridge Application

## 2 Relation Schemas

### User

**Relational Model:** User(user\_id, first\_name, last\_name, email, password)

**Functional Dependencies:**  $\{(user\_id \rightarrow \text{first\_name}, \text{last\_name}, \text{email}, \text{password})\}$

**Candidate Keys:**  $\{(user\_id)\}$

**Primary Key:**  $\{(user\_id)\}$

**Foreign Keys:** None

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS User (
    user_id      INT NOT NULL AUTO_INCREMENT,
    first_name   VARCHAR(50) NOT NULL,
    last_name    VARCHAR(50) NOT NULL,
    email        VARCHAR(50) NOT NULL,
    password     VARCHAR(50) NOT NULL,
    PRIMARY KEY (user_id),
    UNIQUE (email)
);
```

### AppUser

**Relational Model:** AppUser(user\_id, user\_role)

**Functional Dependencies:**  $\{(user\_id \rightarrow \text{user\_role})\}$

**Candidate Keys:**  $\{(user\_id)\}$

**Primary Key:**  $\{(user\_id)\}$

**Foreign Keys:** user\_id references User(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS AppUser(
    user_id      INT NOT NULL,
    user_role    VARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    PRIMARY KEY (user_id),
    CHECK (user_role in ('Professional', 'Recruiter', 'Skill Assessor',
    'Career Expert'))
);
```

### Admin

**Relational Model:** Admin(user\_id, can\_approve\_applications)

**Functional Dependencies:**  $\{(user\_id \rightarrow \text{can\_approve\_applications})\}$

**Candidate Keys:**  $\{(user\_id)\}$

**Primary Key:**  $\{(user\_id)\}$

**Foreign Keys:** user\_id references User(user\_id)

**Normal Form:** BCNF

### **SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Admin(
    user_id          INT NOT NULL
    can_approve_applications  BOOLEAN NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    PRIMARY KEY (user_id)
);
```

### **SystemReport**

**Relational Model:** SystemReport(report\_id, report\_type, creation\_date, content, description, admin\_id)

**Functional Dependencies:** {(report\_id → report\_type, creation\_date, admin\_id)}

**Candidate Keys:** {(report\_id)}

**Primary Key:** {(report\_id)}

**Foreign Keys:** admin\_id references Admin(user\_id)

**Normal Form:** BCNF

### **SQL Definition:**

```
CREATE TABLE IF NOT EXISTS SystemReport(
    report_id          INT NOT NULL AUTO_INCREMENT,
    report_type        VARCHAR(25) NOT NULL,
    content            JSON NOT NULL,
    description        VARCHAR(250),
    creation_date      DATETIME NOT NULL,
    admin_id           INT NOT NULL,
    FOREIGN KEY (admin_id) REFERENCES Admin(user_id),
    PRIMARY KEY (report_id),
    CHECK (report_type IN ('User Analytics', 'Ad Analytics'));
```

### **Connection**

**Relational Model:** Connection(sender\_id, receiver\_id, response)

**Functional Dependencies:** {(sender\_id, receiver\_id → response)}

**Candidate Keys:** {(sender\_id, receiver\_id)}

**Primary Key:** {(sender\_id, receiver\_id)}

**Foreign Keys:** sender\_id references AppUser(user\_id), receiver\_id references AppUser(user\_id)

**Normal Form:** BCNF

### **SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Connection(
    sender_id          INT NOT NULL,
    receiver_id         INT NOT NULL,
    response           BOOLEAN,
    FOREIGN KEY (sender_id) REFERENCES AppUser(user_id) ON DELETE CASCADE,
    FOREIGN KEY (receiver_id) REFERENCES AppUser(user_id) ON DELETE CASCADE
    PRIMARY KEY (sender_id, receiver_id);
```

## **Application**

**Relational Model:** Application(application\_id, user\_id, description, application\_type, application\_date)

**Functional Dependencies:** {(application\_id → user\_id, description, application\_type, application\_date)}

**Candidate Keys:** {(application\_id)}

**Primary Key:** {(application\_id)}

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Application(
    application_id          INT NOT NULL AUTO_INCREMENT,
    user_id                 INT NOT NULL,
    description             TEXT,
    application_type        VARCHAR(25) NOT NULL,
    application_date        DATETIME NOT NULL,
    FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE CASCADE,
    PRIMARY KEY (application_id),
    CHECK (application_type in ('Career Expert', 'Recruiter', 'Skill Assessor'))
);
```

## **ApplicationReview**

**Relational Model:** ApplicationReview(application\_id, admin\_id, response)

**Functional Dependencies:** {(application\_id, admin\_id → response)}

**Candidate Keys:** {(application\_id, admin\_id)}

**Primary Key:** {(application\_id, admin\_id)}

**Foreign Keys:** application\_id references Application(application\_id), admin\_id references Admin(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS ApplicationReview(
    application_id          INT NOT NULL,
    admin_id                INT,
    response                BOOLEAN,
    FOREIGN KEY (application_id) REFERENCES Application(application_id) ON DELETE CASCADE,
    FOREIGN KEY (admin_id) REFERENCES Admin(user_id) ON DELETE SET NULL,
    PRIMARY KEY (application_id, admin_id)
);
```

## **Thread**

**Relational Model:** Thread(thread\_id, creator\_id, receiver\_id)

**Functional Dependencies:** {(thread\_id → creator\_id, receiver\_id)}

**Candidate Keys:** {(thread\_id)}

**Primary Key:** {(thread\_id)}

**Foreign Keys:** creator\_id references AppUser(user\_id), receiver\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Thread(
    thread_id           INT NOT NULL AUTO_INCREMENT,
    creator_id          INT NOT NULL,
    receiver_id         INT NOT NULL,
    FOREIGN KEY (creator_id) REFERENCES AppUser(user_id) ON
    DELETE CASCADE,
    FOREIGN KEY (receiver_id) REFERENCES AppUser(user_id) ON
    DELETE CASCADE,
    PRIMARY KEY (thread_id),
    CONSTRAINT unique_participants UNIQUE(creator_id, receiver_id)
);
```

### Appointment

**Relational Model:** Appointment(thread\_id, date, details)

**Functional Dependencies:** {(thread\_id, date → details)}

**Candidate Keys:** {(thread\_id, date)}

**Primary Key:** {(thread\_id, date)}

**Foreign Keys:** thread\_id references Thread(thread\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Appointment(
    thread_id           INT NOT NULL,
    date                DATETIME NOT NULL,
    details             TEXT,
    FOREIGN KEY (thread_id) REFERENCES Thread(thread_id) ON
    DELETE CASCADE,
    PRIMARY KEY (thread_id, date)
);
```

### Message

**Relational Model:** Message(message\_id, thread\_id, content, file, sent\_time, from\_thread\_creator)

**Functional Dependencies:** {(message\_id → thread\_id, content, file, sent\_time, from\_thread\_creator)}

**Candidate Keys:** {(message\_id)}

**Primary Key:** {(message\_id)}

**Foreign Keys:** thread\_id references Thread(thread\_id), sender\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Message(
    message_id          INT NOT NULL AUTO_INCREMENT,
    thread_id           INT NOT NULL,
```

```

content          TEXT NOT NULL,
file            BLOB,
sent_time       DATETIME NOT NULL,
has_read        BOOLEAN DEFAULT FALSE,
from_thread_creator  BOOLEAN NOT NULL,
FOREIGN KEY (thread_id) REFERENCES Thread(thread_id) ON
DELETE CASCADE,
PRIMARY KEY (message_id)
);

```

## **Notification**

**Relational Model:** Notification(user\_id, created\_at, details)

**Functional Dependencies:** None

**Candidate Keys:**  $\{(user\_id, created\_at, details)\}$

**Primary Key:**  $\{(user\_id, created\_at, details)\}$

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Notification(
user_id           INT NOT NULL,
created_at        DATETIME NOT NULL,
details           VARCHAR(1000) NOT NULL,
has_read          BOOLEAN DEFAULT FALSE,
FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE
CASCADE,
PRIMARY KEY (user_id, created_at, details)
);

```

## **Post**

**Relational Model:** Post(post\_id, user\_id, content, attachment, post\_date)

**Functional Dependencies:**  $\{(post\_id \rightarrow user\_id, content, attachment, post\_date)\}$

**Candidate Keys:**  $\{(post\_id)\}$

**Primary Key:**  $\{(post\_id)\}$

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Post(
post_id          INT NOT NULL AUTO_INCREMENT,
user_id          INT NOT NULL,
content          VARCHAR(2048) NOT NULL,
attachment       BLOB,
post_date        DATETIME NOT NULL,
FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE
CASCADE,
PRIMARY KEY (post_id)
);

```

### **Comment**

**Relational Model:** Comment(post\_id, content, commented\_at, user\_id)

**Functional Dependencies:** {(post\_id, content, commented\_at) → user\_id}

**Candidate Keys:** {(post\_id, content, commented\_at)}

**Primary Key:** {(post\_id, content, commented\_at)}

**Foreign Keys:** post\_id references Post(post\_id), user\_id references AppUser(user\_id)

**Normal Form:** BCNF

### **SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Comment(
    post_id          INT NOT NULL,
    user_id          INT NOT NULL,
    content          VARCHAR(2048) NOT NULL,
    commented_at     DATETIME NOT NULL,
    FOREIGN KEY (post_id) REFERENCES Post(post_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE CASCADE,
    PRIMARY KEY (post_id, content, commented_at)
);
```

### **JobAdvertisement**

**Relational Model:** JobAdvertisement(ad\_id, creator\_id, title, description, organization, setting, location, type, pay\_range\_min, pay\_range\_max, domain, is\_open, external\_url, application\_count, view\_count, created\_at)

**Functional Dependencies:** {(ad\_id → creator\_id, title, description, organization, setting, location, type, pay\_range\_min, pay\_range\_max, domain, is\_open, external\_url, application\_count, view\_count, created\_at)}

**Candidate Keys:** {(ad\_id)}

**Primary Key:** {(ad\_id)}

**Foreign Keys:** creator\_id references AppUser(user\_id)

**Normal Form:** BCNF

### **SQL Definition:**

```
CREATE TABLE IF NOT EXISTS JobAdvertisement(
    ad_id           INT NOT NULL AUTO_INCREMENT,
    creator_id      INT NOT NULL,
    title           VARCHAR(50) NOT NULL,
    description     TEXT NOT NULL,
    organization    VARCHAR(100) NOT NULL,
    setting          VARCHAR(15) NOT NULL,
    location         VARCHAR(100) NOT NULL,
    type             VARCHAR(15) NOT NULL,
    pay_range_min   INT NOT NULL,
    pay_range_max   INT NOT NULL,
    domain           VARCHAR(50),
    is_open          BOOLEAN NOT NULL,
```

```

external_url          VARCHAR(1000),
application_count     INT NOT NULL DEFAULT 0,
view_count            INT NOT NULL DEFAULT 0,
created_at            DATETIME NOT NULL,
FOREIGN KEY (creator_id) REFERENCES AppUser(user_id) ON
DELETE CASCADE,
PRIMARY KEY (ad_id),
CHECK (setting IN ('On-site', 'Remote', 'Hybrid'),
CHECK (type IN ('Internship', 'Part-time Job', 'Job', 'Contract',
'Project-based'))
);

```

### **JobAdvertisementResponse**

**Relational Model:** JobAdvertisementResponse(profile\_id, ad\_id, response)

**Functional Dependencies:**  $\{(user\_id, ad\_id \rightarrow response)\}$

**Candidate Keys:**  $\{(profile\_id, ad\_id)\}$

**Primary Key:**  $\{(profile\_id, ad\_id)\}$

**Foreign Keys:** user\_id references Profile(profile\_id), ad\_id references JobAdvertisement(ad\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS JobAdvertisementResponse(
profile_id           INT NOT NULL,
ad_id                INT NOT NULL,
apply_date           DATETIME NOT NULL,
response_date        DATETIME,
response             VARCHAR(15),
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
FOREIGN KEY (ad_id) REFERENCES JobAdvertisement(ad_id) ON
DELETE CASCADE,
PRIMARY KEY (user_id, ad_id),
CHECK (response IN ('Waiting', 'Interview', 'Accepted', 'Rejected'))
);

```

### **Interview**

**Relational Model:** Interview(ad\_id, start\_time, interview\_date, details)

**Functional Dependencies:**  $\{(ad\_id, start\_time, interview\_date \rightarrow details)\}$

**Candidate Keys:**  $\{(ad\_id, start\_time, interview\_date)\}$

**Primary Key:**  $\{(ad\_id, start\_time, interview\_date)\}$

**Foreign Keys:** ad\_id references JobAdvertisement(ad\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Interview(
ad_id                INT NOT NULL,
interview_date        DATETIME NOT NULL,

```

```

start_time           DATETIME NOT NULL,
details             TEXT,
FOREIGN KEY (ad_id) REFERENCES JobAdvertisement(ad_id) ON
DELETE SET NULL,
PRIMARY KEY (ad_id, start_time, interview_date)
);

```

## Profile

**Relational Model:** Profile(profile\_id, user\_id, avatar, country, external\_portfolio\_url, address, biography, is\_private, resume, phone\_number, is\_application\_specific, created\_at)

**Functional Dependencies:** {(profile\_id → user\_id, avatar, country, external\_portfolio\_url, address, biography, is\_private, resume, phone\_number, is\_application\_specific, created\_at)}

**Candidate Keys:** {(profile\_id)}

**Primary Key:** {(profile\_id)}

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Profile(
profile_id          INT NOT NULL AUTO_INCREMENT,
user_id             INT NOT NULL,
avatar              BLOB,
country             VARCHAR(50),
external_portfolio_url VARCHAR(100),
address             VARCHAR(300),
biography            TEXT,
is_private           BOOLEAN NOT NULL,
resume               BLOB,
phone_number         VARCHAR(50),
is_application_specific BOOLEAN NOT NULL,
created_at           DATETIME NOT NULL,
FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE
CASCADE,
PRIMARY KEY (profile_id)
);

```

## LanguageProficiency

**Relational Model:** LanguageProficiency(language\_id, profile\_id, language\_name, proficiency)

**Functional Dependencies:** {(language\_id → profile\_id, language\_name, proficiency)}

**Candidate Keys:** {(language\_id)}

**Primary Key:** {(language\_id)}

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS LanguageProficiency(
```

```

language_id          INT NOT NULL AUTO_INCREMENT,
profile_id           INT NOT NULL,
language_name        VARCHAR(50) NOT NULL,
proficiency          VARCHAR(50),
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (language_id)
);

```

### Award

**Relational Model:** Award(award\_id, profile\_id, title, description, issue\_date, issuer)

**Functional Dependencies:** {(award\_id → profile\_id, title, description, issue\_date, issuer)}

**Candidate Keys:** {(award\_id)}

**Primary Key:** {(award\_id)}

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Award(
award_id          INT NOT NULL AUTO_INCREMENT,
profile_id           INT NOT NULL,
title               VARCHAR(50) NOT NULL,
description          VARCHAR(1000),
issue_date          DATETIME,
issuer              VARCHAR(100),
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (award_id)
);

```

### Project

**Relational Model:** Project(project\_id, profile\_id, title, description, start\_date, end\_date, project\_url)

**Functional Dependencies:** {(project\_id → profile\_id, title, description, start\_date, end\_date, project\_url)}

**Candidate Keys:** {(project\_id)}

**Primary Key:** {(project\_id)}

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Project(
project_id          INT NOT NULL AUTO_INCREMENT,
profile_id           INT NOT NULL,
title               VARCHAR(50) NOT NULL,
description          VARCHAR(1000),
start_date          DATETIME,
end_date             DATETIME,

```

```

project_url          VARCHAR(100),
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (project_id)
);

```

### **Certification**

**Relational Model:** Certification(certification\_id, profile\_id, certification\_name, description, credential\_url, issue\_date, issuer, expiration\_date)

**Functional Dependencies:**  $\{(certification\_id \rightarrow profile\_id, certification\_name, description, credential\_url, issue\_date, issuer, expiration\_date)\}$

**Candidate Keys:**  $\{(certification\_id)\}$

**Primary Key:**  $\{(certification\_id)\}$

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Certification(
certification_id      INT NOT NULL AUTO_INCREMENT,
profile_id            INT NOT NULL,
certification_name    VARCHAR(50) NOT NULL,
description           VARCHAR(1000),
credential_url        VARCHAR(100),
issue_date            DATETIME,
issuer                VARCHAR(100),
expiration_date       DATETIME,
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (certification_id)
);

```

### **Publication**

**Relational Model:** Publication(publication\_id, profile\_id, title, description, publication\_date, publisher, publication\_url)

**Functional Dependencies:**  $\{(publication\_id \rightarrow profile\_id, title, description, publication\_date, publisher, publication\_url)\}$

**Candidate Keys:**  $\{(publication\_id)\}$

**Primary Key:**  $\{(publication\_id)\}$

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Publication(
publication_id         INT NOT NULL AUTO_INCREMENT,
profile_id             INT NOT NULL,
title                  VARCHAR(50) NOT NULL,
description            VARCHAR(1000),
publication_date       DATETIME,
publisher              VARCHAR(100),

```

```

publication_url           VARCHAR(100),
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (publication_id)
);

```

### **TestScore**

**Relational Model:** TestScore(test\_score\_id, profile\_id, test\_name, description, test\_date, score, attachment)

**Functional Dependencies:**  $\{(test\_score\_id \rightarrow \text{profile\_id}, \text{test\_name}, \text{description}, \text{test\_date}, \text{score}, \text{attachment})\}$

**Candidate Keys:**  $\{(test\_score\_id)\}$

**Primary Key:**  $\{(test\_score\_id)\}$

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS TestScore(
test_score_id           INT NOT NULL AUTO_INCREMENT,
profile_id               INT NOT NULL,
test_name                VARCHAR(50) NOT NULL,
description              VARCHAR(1000),
test_date                DATETIME,
score                    VARCHAR(50),
attachment               BLOB,
FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (test_score_id)
);

```

### **Skill**

**Relational Model:** Skill(skill\_id, profile\_id, name, is\_verified, is\_master\_skill)

**Functional Dependencies:**  $\{(skill\_id \rightarrow \text{profile\_id}, \text{name}, \text{is\_verified}, \text{is\_master\_skill})\}$

**Candidate Keys:**  $\{(skill\_id)\}$

**Primary Key:**  $\{(skill\_id)\}$

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Skill(
skill_id                INT NOT NULL AUTO_INCREMENT,
profile_id               INT NOT NULL,
name                     VARCHAR(50) NOT NULL,
is_verified              BOOLEAN NOT NULL DEFAULT
FALSE,
is_master_skill          BOOLEAN NOT NULL DEFAULT
FALSE,

```

```

FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
DELETE CASCADE,
PRIMARY KEY (skill_id)
);

```

### **SkillInJobAdvertisement**

**Relational Model:** SkillInJobAdvertisement(skill\_id, ad\_id)

**Functional Dependencies:** None

**Candidate Keys:** {(skill\_id, ad\_id)}

**Primary Key:** {(skill\_id, ad\_id)}

**Foreign Keys:** skill\_id references Skill(skill\_id), ad\_id references JobAdvertisement(ad\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS SkillInJobAdvertisement(
    skill_id           INT NOT NULL,
    ad_id              INT NOT NULL,
    FOREIGN KEY (skill_id) REFERENCES Skill(skill_id) ON DELETE
    CASCADE,
    FOREIGN KEY (ad_id) REFERENCES JobAdvertisement(ad_id) ON
    DELETE CASCADE,
    PRIMARY KEY (skill_id, ad_id)
);

```

### **Experience**

**Relational Model:** Experience(experience\_id, profile\_id, title, start\_date, end\_date, description, current\_status)

**Functional Dependencies:** {(experience\_id → profile\_id, title, start\_date, end\_date, description, current\_status)}

**Candidate Keys:** {(experience\_id)}

**Primary Key:** {(experience\_id)}

**Foreign Keys:** profile\_id references Profile(profile\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Experience(
    experience_id      INT NOT NULL AUTO_INCREMENT,
    profile_id         INT NOT NULL,
    title              VARCHAR(50) NOT NULL,
    start_date         DATETIME,
    end_date           DATETIME,
    description        VARCHAR(1000),
    current_status     VARCHAR(25),
    FOREIGN KEY (profile_id) REFERENCES Profile(profile_id) ON
    DELETE CASCADE,
    PRIMARY KEY (experience_id),
    CHECK (current_status in ('Working', 'Past Working Experience'))
);

```

### **EducationalExperience**

**Relational Model:** EducationalExperience(experience\_id, grade, field\_of\_study, school\_name, degree\_id)

**Functional Dependencies:** {(experience\_id → grade, field\_of\_study, school\_name, degree\_id)}

**Candidate Keys:** {(experience\_id)}

**Primary Key:** {(experience\_id)}

**Foreign Keys:** experience\_id references Experience(experience\_id), school\_name references School(school\_name), degree\_id references Degree(degree\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS EducationalExperience(
    experience_id           INT NOT NULL,
    grade                   VARCHAR(50),
    field_of_study          VARCHAR(50),
    school_name             VARCHAR(50) NOT NULL,
    degree_id               INT NOT NULL,
    FOREIGN KEY (experience_id) REFERENCES
        Experience(experience_id) ON DELETE CASCADE,
    FOREIGN KEY (school_name) REFERENCES School(school_name)
        ON DELETE CASCADE,
    FOREIGN KEY (degree_id) REFERENCES Degree(degree_id) ON
        DELETE CASCADE,
    PRIMARY KEY (experience_id)
);
```

### **Degree**

**Relational Model:** Degree(degree\_id, name)

**Functional Dependencies:** {(degree\_id → name)}

**Candidate Keys:** {(degree\_id)}

**Primary Key:** {(degree\_id)}

**Foreign Keys:** None

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS Degree(
    degree_id               INT NOT NULL,
    name                   VARCHAR(50),
    PRIMARY KEY (degree_id)
);
```

### **DegreeInJobAdvertisement**

**Relational Model:** DegreeInJobAdvertisement(degree\_id, ad\_id)

**Functional Dependencies:** None

**Candidate Keys:** {(degree\_id, ad\_id)}

**Primary Key:** {(degree\_id, ad\_id)}

**Foreign Keys:** degree\_id references Degree(degree\_id), ad\_id references JobAdvertisement(ad\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS DegreeInJobAdvertisement(
    degree_id           INT NOT NULL,
    ad_id                INT NOT NULL,
    FOREIGN KEY (degree_id) REFERENCES Degree(degree_id) ON
    DELETE CASCADE,
    FOREIGN KEY (ad_id) REFERENCES JobAdvertisement(ad_id) ON
    DELETE CASCADE,
    PRIMARY KEY (degree_id, ad_id)
);
```

### School

**Relational Model:** School(school\_name, description, location, external\_url)

**Functional Dependencies:**  $\{(school\_name \rightarrow description, location, external\_url)\}$

**Candidate Keys:**  $\{(school\_name)\}$

**Primary Key:**  $\{(school\_name)\}$

**Foreign Keys:** None

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS School(
    school_name          VARCHAR(50) NOT NULL,
    description           TEXT,
    location              VARCHAR(100),
    external_url          VARCHAR(100),
    PRIMARY KEY (school_name)
);
```

### WorkExperience

**Relational Model:** WorkExperience(experience\_id, setting, type, company\_name)

**Functional Dependencies:**  $\{(experience\_id \rightarrow setting, type, company\_name)\}$

**Candidate Keys:**  $\{(experience\_id)\}$

**Primary Key:**  $\{(experience\_id)\}$

**Foreign Keys:** experience\_id references Experience(experience\_id), company\_name references Company(company\_name)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS WorkExperience (
    experience_id        INT NOT NULL,
    setting               VARCHAR(25) NOT NULL,
    type                  VARCHAR(15),
    company_name          VARCHAR(50) NOT NULL,
```

```

FOREIGN KEY (experience_id) REFERENCES
Experience(experience_id) ON DELETE CASCADE,
FOREIGN KEY (company_name) REFERENCES
Company(company_name) ON DELETE CASCADE,
PRIMARY KEY (experience_id),
CHECK (type IN ('Internship', 'Part-time Job', 'Job', 'Contract',
'Project-based')),
CHECK (setting IN ('On-site', 'Remote', 'Hybrid'))
);

```

### **Company**

**Relational Model:** Company(company\_name, description, location, external\_url)

**Functional Dependencies:**  $\{(company\_name \rightarrow description, location, external\_url)\}$

**Candidate Keys:**  $\{(company\_name)\}$

**Primary Key:**  $\{(company\_name)\}$

**Foreign Keys:** None

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Company(
company_name           VARCHAR(50) NOT NULL,
description            VARCHAR(1000),
location               VARCHAR(100),
external_url            VARCHAR(100),
PRIMARY KEY (company_name)
);

```

### **VoluntaryExperience**

**Relational Model:** VoluntaryExperience(experience\_id, responsibility, organization\_name)

**Functional Dependencies:**  $\{(experience\_id \rightarrow responsibility, organization\_name)\}$

**Candidate Keys:**  $\{(experience\_id)\}$

**Primary Key:**  $\{(experience\_id)\}$

**Foreign Keys:** experience\_id references Experience(experience\_id), organization\_name references NonProfitOrganization(organization\_name)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS VoluntaryExperience(
experience_id          INT NOT NULL,
responsibility          TEXT,
organization_name        VARCHAR(50) NOT NULL,
FOREIGN KEY (experience_id) REFERENCES
Experience(experience_id) ON DELETE CASCADE,
FOREIGN KEY (organization_name) REFERENCES
NonProfitOrganization(organization_name) ON DELETE CASCADE,
PRIMARY KEY (experience_id));

```

### **NonProfitOrganization**

**Relational Model:** NonProfitOrganization(organization\_name, description, location, external\_url)

**Functional Dependencies:**  $\{(organization\_name \rightarrow description, location, external\_url)\}$

**Candidate Keys:**  $\{(organization\_name)\}$

**Primary Key:**  $\{(organization\_name)\}$

**Foreign Keys:** None

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS NonProfitOrganization(
    organization_name          VARCHAR(50) NOT NULL,
    description                VARCHAR(1000),
    location                   VARCHAR(100),
    external_url               VARCHAR(100),
    PRIMARY KEY (organization_name)
);
```

### **SkillAssessor**

**Relational Model:** SkillAssessor(user\_id, field\_of\_expertise)

**Functional Dependencies:**  $\{(user\_id \rightarrow field\_of\_expertise)\}$

**Candidate Keys:**  $\{(user\_id)\}$

**Primary Key:**  $\{(user\_id)\}$

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS SkillAssessor(
    user_id                  INT NOT NULL,
    field_of_expertise       VARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE CASCADE,
    PRIMARY KEY (user_id)
);
```

### **SkillAssessment**

**Relational Model:** SkillAssessment(skill\_id, assessor\_id, rating)

**Functional Dependencies:**  $\{(skill\_id, assessor\_id \rightarrow rating)\}$

**Candidate Keys:**  $\{(skill\_id, assessor\_id)\}$

**Primary Key:**  $\{(skill\_id, assessor\_id)\}$

**Foreign Keys:** skill\_id references Skill(skill\_id), assessor\_id references SkillAssessor(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```
CREATE TABLE IF NOT EXISTS SkillAssessment(
    skill_id                 INT NOT NULL,
    assessor_id               INT NOT NULL,
    rating                   VARCHAR(50) NOT NULL,
```

```

FOREIGN KEY (skill_id) REFERENCES Skill(skill_id) ON DELETE
CASCADE,
FOREIGN KEY (assessor_id) REFERENCES SkillAssessor(user_id) ON
DELETE SET NULL,
PRIMARY KEY (skill_id, assessor_id)
);

```

### **CareerExpert**

**Relational Model:** CareerExpert(user\_id, mentee\_limit)

**Functional Dependencies:**  $\{(user_id \rightarrow \text{mentee\_limit})\}$

**Candidate Keys:**  $\{(user_id)\}$

**Primary Key:**  $\{(user_id)\}$

**Foreign Keys:** user\_id references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS CareerExpert(
    user_id          INT NOT NULL,
    mentee_limit     INT,
    FOREIGN KEY (user_id) REFERENCES AppUser(user_id) ON DELETE
    CASCADE,
    PRIMARY KEY (user_id)
);

```

### **Mentorship**

**Relational Model:** Mentorship(mentor\_id, mentee\_id, response)

**Functional Dependencies:**  $\{(mentor_id, \text{mentee\_id} \rightarrow \text{response})\}$

**Candidate Keys:**  $\{(mentor_id, \text{mentee\_id})\}$

**Primary Key:**  $\{(mentor_id, \text{mentee\_id})\}$

**Foreign Keys:** mentor\_id references CareerExpert(user\_id), mentee\_id  
references AppUser(user\_id)

**Normal Form:** BCNF

**SQL Definition:**

```

CREATE TABLE IF NOT EXISTS Mentorship(
    mentor_id        INT NOT NULL,
    mentee_id        INT NOT NULL,
    response         BOOLEAN,
    FOREIGN KEY (mentor_id) REFERENCES CareerExpert(user_id) ON
    DELETE CASCADE,
    FOREIGN KEY (mentee_id) REFERENCES AppUser(user_id) ON
    DELETE CASCADE,
    PRIMARY KEY (mentor_id, mentee_id)
);

```

## 3 Advanced Database Components

### 3.1 Views

#### 3.1.1 User Login View

```
CREATE VIEW UserLogin AS
SELECT u.user_id, u.first_name, u.last_name, u.email,
u.password
FROM User u
LEFT OUTER JOIN AppUser au ON u.user_id = au.user_id
LEFT OUTER JOIN Admin ad ON u.user_id = ad.user_id;
```

#### 3.1.2 Get User Role View

```
CREATE VIEW GetUserRole AS
SELECT u.user_id, au.user_role,
CASE WHEN ad.can_approve_applications IS NOT NULL THEN
1 ELSE 0 END AS is_admin
FROM User u
LEFT OUTER JOIN AppUser au ON u.user_id = au.user_id
LEFT OUTER JOIN Admin ad ON u.user_id = ad.user_id;
```

### 3.2 Triggers

#### 3.2.1 Skill Assessment Notification Trigger

```
CREATE Trigger skillAssessmentNotificationTrigger
AFTER INSERT ON SkillAssesment
REFERENCING NEW ROW AS new_row
FOR EACH ROW
BEGIN
WITH user_skill AS (
    SELECT user_id, name
    FROM Skill
    WHERE skill_id = new_row.skill_id
)
INSERT INTO Notification (user_id, notification_date,
notification_content)
SELECT user_id, NOW(),
CONCAT('You have been assessed a ', new_row.rating, ' '
for ', name)
FROM user_skill;
END;
```

#### 3.2.2 Comment Notification Trigger

```
CREATE Trigger commentNotificationTrigger
```

```

AFTER INSERT ON Comment
REFERENCING NEW ROW AS new_row
FOR EACH ROW
BEGIN
WITH post_temp AS (
    SELECT
        CONCAT(U.first_name, " ", U.last_name) AS
commentor_name,
        P.user_id as post_user_id
    FROM User as U, Post as P
    WHERE U.user_id = new_row.user_id
        AND P.post_id = new_row.post_id
)
INSERT INTO Notification (user_id, notification_date,
notification_content)
SELECT post_user_id, NOW(),
        CONCAT(commentor_name, ' commented on your post')
FROM post_temp
END;

```

### **3.2.3 Application Notification Trigger**

```

CREATE TRIGGER applicationNotificationTrigger
AFTER INSERT ON ApplicationReview
REFERENCING NEW ROW AS new_row
FOR EACH ROW
BEGIN
WITH application_temp AS (
    SELECT A.user_id, A.application_type,
        CONCAT(U.first_name, " ", U.last_name) AS
admin_name
    FROM Application as A, User as U
    WHERE application_id = new_row.application_id
        AND U.user_id = new_row.admin_id
)
INSERT INTO Notification (user_id, notification_date,
notification_content)
SELECT user_id, NOW(),
        CASE WHEN new_row.response IS TRUE THEN CONCAT(
            'Your ', application_type, ' application has
been approved by ',
            admin_name
        )
        ELSE CONCAT(
            'Your ', application_type, ' application has
been rejected by ',
            admin_name
        )
        END
FROM application_temp;

```

```
END;
```

### 3.2.4 Connection Request Notification Trigger

```
CREATE TRIGGER connectionRequestNotificationTrigger
AFTER INSERT ON Connection
REFERENCING NEW ROW AS new_row
FOR EACH ROW
BEGIN
WITH connection_temp AS (
    SELECT
        CONCAT(U.first_name, " ", U.last_name) AS requester_name,
        FROM
            User AS U
        WHERE
            U.user_id = new_row.sender_id
)
INSERT INTO Notification (user_id, notification_date,
notification_content)
SELECT new_row.receiver_id, NOW(),
CONCAT(requester_name, ' sent you a connection
request')
FROM connection_temp;
END;
```

### 3.2.5 Connection Response Notification Trigger

```
CREATE TRIGGER connectionResponseNotificationTrigger
AFTER UPDATE ON Connection REFERENCING NEW ROW AS new_row
FOR EACH ROW
BEGIN
WITH connection_temp AS (
    SELECT
        CONCAT(U.first_name, " ", U.last_name) AS respondent_name,
        FROM User AS U
        WHERE U.user_id = new_row.receiver_id
)
INSERT INTO Notification (user_id, notification_date,
notification_content)
SELECT new_row.sender_id, NOW(),
CASE WHEN new_row.response IS TRUE
    THEN CONCAT(requester_name,
        ' accepted your connection request'
    )
    ELSE CONCAT( requester_name,
        ' rejected your connection request'
    )
)
```

```

        END
FROM connection_temp;
END;

```

### **3.2.6 Add User to AppUser Trigger**

```

CREATE TRIGGER add_user_to_appuser
AFTER INSERT ON User
FOR EACH ROW
BEGIN
    INSERT INTO AppUser (user_id, user_role)
    VALUES (NEW.user_id, 'Professional');
END;

```

## **3.3 Assertions**

### **3.3.1 Check if Each User Has at Most One Master Skill**

```

CREATE ASSERTION num_of_master_skill
CHECK (1 >= all(SELECT COUNT(*)
                  FROM Skill S, Profile P
                  WHERE S.profile_id = P.profile_id AND
                  S.is_master_skill = 1));

```

### **3.3.2 Check if Number of Mentees of Each CareerExpert Does Not Exceed Mentee Limit**

```

CREATE ASSERTION num_of_mentees CHECK (
    NOT EXISTS (
        SELECT
            mentor_id,
            COUNT(DISTINCT mentee_id)
        FROM
            Mentorship
        HAVING
            COUNT(DISTINCT mentee_id) > (
                SELECT
                    mentee_limit
                FROM
                    CareerExpert
                WHERE
                    user_id = mentor_id
            )
        GROUP BY
            mentor_id
    );

```

### **3.3.3 Check if the Receiver and Creator of a Thread are not the Same Person**

```
CREATE ASSERTION thread_distinct_person CHECK (
    NOT EXISTS(
        SELECT
            thread_id
        FROM
            Thread
        WHERE
            creator_id = receiver_id
    )
);
```

## **3.4 Procedures**

### **3.4.1 Procedure for Increasing the Application Count of an Advertisement by One**

```
CREATE PROCEDURE increase_application_count(IN ad_id_param INT)
BEGIN
    UPDATE JobAdvertisement
    SET application_count = application_count + 1
    WHERE ad_id = ad_id_param;
END;
```

### **3.4.2 Procedure for Increasing the View Count of an Advertisement by One**

```
CREATE PROCEDURE increase_view_count(IN ad_id_param INT)
BEGIN
    UPDATE JobAdvertisement
    SET view_count = view_count + 1
    WHERE ad_id = ad_id_param;
END;
```

## 4 User Interface Design and SQL Statements

### 4.1 Login Page

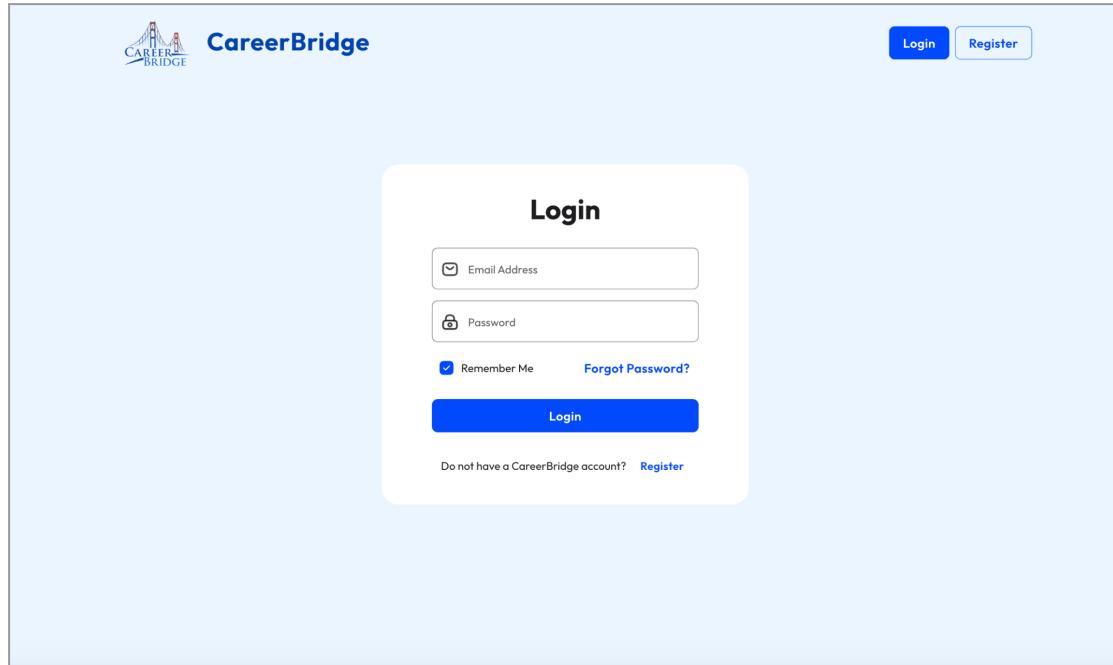


Figure 2

**Inputs:** @email, @password

**On Login button pressed:**

```
SELECT
    user_id,
    first_name,
    last_name,
    user_role,
    can_approve_applications,
    is_admin
FROM
    UserLogin
WHERE
    email = @email
    AND password = @password;
```

If the query above returns successfully, then the result of this query will be saved as a cookie to ease the future queries (i.e., get profile data using user\_id). Otherwise, an error message will be displayed to the user.

```
SELECT user_role, is_admin
FROM GetUserRole
WHERE user_id = @user_id;
```

Also, we have another view called GetUserRole. The user\_role and is\_admin fields from this view will be gathered using the saved user\_id and they will be used to display different pages for different types of users (for example the post job advertisement screen for recruiters or the admin panel for admins). Hence, we do not need separate login screens for different user roles. We do not save the role information as a cookie to prevent manipulation and increase the security when rendering the pages.

## 4.2 Register Page

The screenshot shows the 'Register' page of the CareerBridge website. At the top left is the CareerBridge logo. To the right of the logo is the word 'CareerBridge'. Further to the right are two buttons: 'Login' (in a light blue box) and 'Register' (in a dark blue box). Below the header is a large white rectangular form with rounded corners. The form has a title 'Register' at the top center. It contains four input fields: 'First Name' (with a person icon), 'Last Name' (with a person icon), 'Email Address' (with an envelope icon), and 'Password' (with a lock icon). Below these fields is a checkbox labeled 'I have read the [Users & Roles Policy](#) of CareerBridge.' At the bottom of the form is a large blue button labeled 'Create Account'. Below this button, in smaller text, is the message 'Already have a CareerBridge account? [Login](#)'.

Figure 3

**Inputs:** @first\_name, @last\_name, @email, @password

**On Create Account button pressed:**

```
INSERT INTO User (
    first_name, last_name, email, password
)
VALUES (@first_name, @last_name, @email, @password);
```

If the inputs are valid, the user will be notified that the account is created and they will be redirected to the login page (Figure 2).

### 4.3 Policy Page

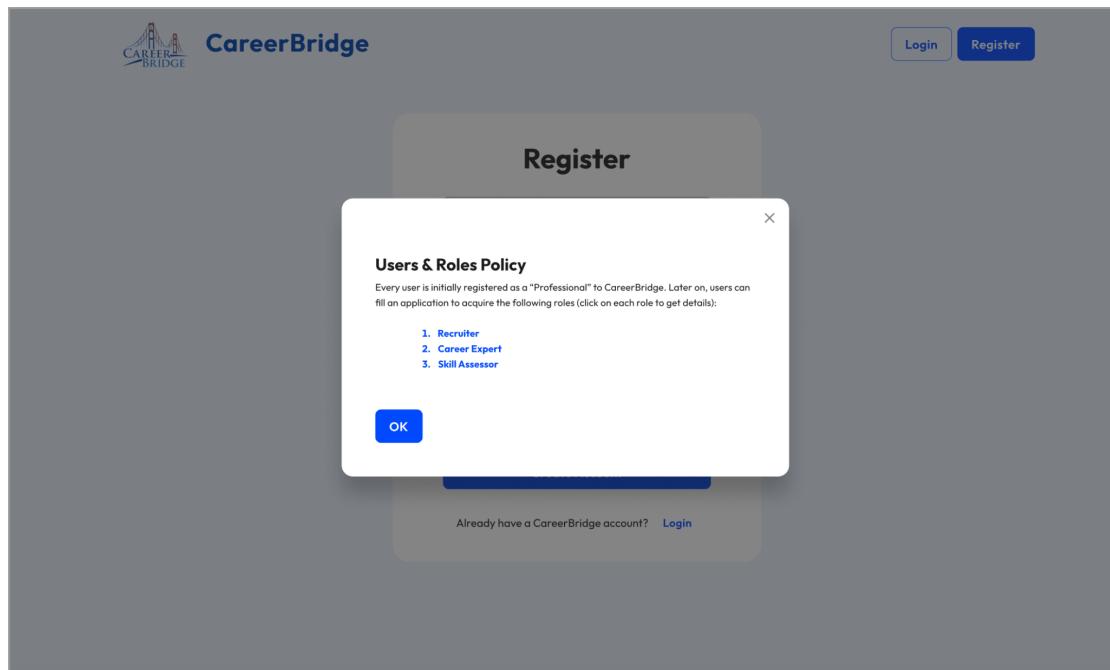


Figure 4

There are no related SQL queries for this page. It is embedded into HTML and the purpose is to notify the users that they will be signed up with the "Professional" role. We created a trigger to assign every user to the "Professional" role after registration.

#### 4.4 Forgot Password Page

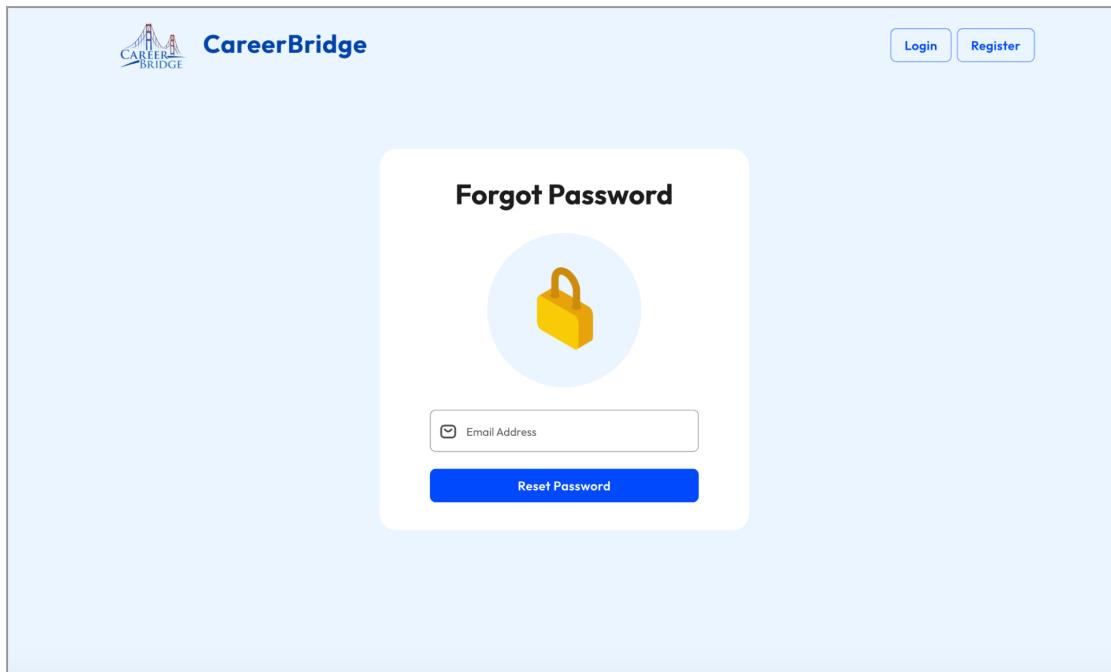


Figure 5

**Inputs:** @email

First of all, we check if the user is registered in the system.

```
SELECT EXISTS(SELECT * FROM User WHERE email = @email) AS  
email_exists;
```

If this query returns 0, we abort and print an error message to screen, indicating that the email is not registered. Otherwise, we send a link to the user via email which will trigger:

```
SET @new_password = CONCAT(SUBSTRING(MD5(RAND()) FROM 1 FOR  
8));
```

```
UPDATE User  
SET password = @new_password  
WHERE email = @email;
```

Then, we expose the @new\_password to the user. They can change it afterwards from the user settings, if they wish.

```
SELECT @new_password AS new_password;
```

## 4.5 Browse Ads Page

The screenshot shows the CareerBridge platform's 'Browse Ads' section. On the left, a sidebar includes 'Feed', 'Browse Ads' (selected), 'All Ads' (highlighted in blue), 'Ads For You', 'Applied Ads', 'Profile', and 'Settings'. The main area is titled 'All Ads' with 328 results and a 'Sort: Most Recent' dropdown. Six job ads are listed:

- UI / UX Designer** at Google in Mountain View, California, USA. Description: Google is seeking a highly skilled UI/UX Designer to join our team. As a UI/UX Designer, you will be responsible for designing innovative user experiences for our products, including web and mobile applications. You will work collaboratively with cross-functional teams including product managers, engineers, and other designers to create high-quality designs that are user-centered, data-driven, and visually appealing. Requirements: Figma, Photoshop, Angular. Salary: \$15000-20000/Month. Status: Open. Posted 25 minutes ago.
- Assistant Professor of Psychology** at University of Toronto in Toronto, Ontario, Canada. Description: The Department of Psychology at the University of Toronto invites applications for a tenure-stream position in the area of Cognitive Psychology, with a focus on Perception, Attention and Memory. The appointment will be at the rank of Assistant Professor and will begin on July 1, 2023... Requirements: Lecturing, Research, Communication. Academy - Job Ad - On-site - Open. Salary: \$20000-25000/Month. Status: Open. Posted 1 day ago.
- Accountant** at Amazon in Seattle, Washington, USA. Description: Amazon is looking for an energetic and enthusiastic candidate to join the fast-paced world of Payroll operations. We are not an average retailer and this is definitely not your average payroll position. Requirements: Finance - Job Ad - Remote - Closed. Salary: \$12500-15000/Month. Status: Closed. Posted 2 days ago.
- HR Manager** at Coca-Cola in Atlanta, Georgia, USA. Description: Coca-Cola is seeking a talented and experienced Human Resources Manager to join our team. In this role, you will be responsible for leading the development and implementation of HR strategies and initiatives to support the company's business objectives. You will work closely with cross-functional teams to build a high-performance culture that fosters employee engagement, development, and retention. Requirements: Communication, Problem Solving. Human Resources - Job Ad - On-site - Open. Salary: \$7500-10000/Month. Status: Open. Posted 3 days ago.
- Software Development Intern** at Microsoft in Redmond, Washington, USA. Description: Are you a talented software developer looking for an opportunity to gain hands-on experience working with cutting-edge technologies? Do you have a passion for creating innovative software solutions that solve real-world problems? If so, we want to hear from you! As a Software Development Intern at Microsoft, you will work alongside a team of experienced developers, designers, and product managers to help build the future of Microsoft. Requirements: C#, DevOps, Team Work. Software Development - Internship Ad - On-site - Closed. Salary: \$4000-4500/Month. Status: Closed. Posted 6 days ago.
- Medical Laboratory Intern** at Guest Diagnostic in Tampa, Florida, USA. Description: We are looking for a skilled Medical Laboratory Intern to join our laboratory in Tampa, Florida. In this role, you will be responsible for performing laboratory tests and procedures, analyzing and interpreting results, and ensuring the accuracy and quality of test results. You will work collaboratively with laboratory staff and healthcare providers to provide high-quality patient care. Requirements: Laboratory Skills, Attention, Basic Coding. Medical Services - Internship Ad - On-site - Open. Salary: \$2000-2500/Month. Status: Open. Posted 14 days ago.

Each ad has a 'View Ad Details' button and a box showing application and view counts. At the bottom, there are navigation links for pages 1 through 55.

Figure 6

**When a user enters to the All Ads page under Browse Ads tab, all available ads will be displayed with the following query:**

```
SELECT * FROM JobAdvertisement;
```

If the "Sort Most Recent" or a similar sorting option is selected then the following query will be executed instead, considering the "Sort Most Recent" case.

```
SELECT *
FROM JobAdvertisement
ORDER BY created_at DESC;
```

## 4.6 Browse Ads Page with Filtering Options

### Scenario 1.1:

The screenshot shows the CareerBridge platform's browse ads interface. On the left, there's a sidebar with options like Feed, Browse Ads (selected), All Ads, Ads For You, Applied Ads, Profile, and Settings. The main area has a search bar at the top right. Below it are various filtering options: Added (Last day, Last week, Last month, Any time), Ad Type (Internship, Part-time Job, Job, Contract, Project-based), Pay Range (\$7500 to \$25000), Status (Open, Closed), Location (USA), Domain (Not Specified), Skills (Communication), and Required Degrees (Bachelor of Science). The results section is titled "Filtered Ads" and shows two job listings: "Assistant Professor of Psychology" at the University of Toronto and "HR Manager" at Coca-Cola. Each listing includes a thumbnail, title, company, location, skills required, pay range, and a "View Ad Details" button.

Figure 7

**The ads will be filtered based on the following query:**

```
SELECT *
FROM
    JobAdvertisement,
WHERE
    DATEDIFF(month, created_at, GETDATE()) = 1
    AND type = "Job"
    AND pay_range_min >= 7500
    AND pay_range_max <= 25000
    AND location LIKE "%USA%"
    AND setting = "On-site"
    AND status = "Open"
    AND ad_id IN (
        (SELECT
            ad_id
        FROM
```

```

DegreeInJobAdvertisement NATURAL
JOIN Degree
WHERE
    name = "Bachelor of Science")
INTERSECT
(SELECT
    ad_id
FROM
    SkillInJobAdvertisement NATURAL JOIN Skill
WHERE
    name = "Communication"));

```

**If a sorting method is selected then a similar query with ORDER BY clause will be executed accordingly:**

```

SELECT *
FROM
    JobAdvertisement,
WHERE
    DATEDIFF(month,created_at,GETDATE()) = 1
    AND type = "Job"
    AND pay_range_min >= 7500
    AND pay_range_max <= 25000
    AND location LIKE "%USA%"
    AND setting = "On-site"
    AND status = "Open"
    AND ad_id IN (
        (SELECT
            ad_id
        FROM
            DegreeInJobAdvertisement NATURAL
            JOIN Degree
        WHERE
            name = "Bachelor of Science")
        INTERSECT
        (SELECT
            ad_id
        FROM
            SkillInJobAdvertisement NATURAL JOIN Skill
        WHERE
            name = "Communication"))
ORDER BY created_at DESC;

```

## Scenario 1.2 (When search is applied)

The screenshot shows the CareerBridge interface. On the left, there's a sidebar with 'Feed', 'Browse Ads' (selected), 'All Ads', 'Ads For You', 'Applied Ads', 'Profile', and 'Settings'. The main area has a search bar with 'Coca-Cola'. Below it are various filters: 'Added' (Last day, Last week, Last month, Any time), 'Ad Type' (Internship, Part-time Job, Job, Contract, Project-based), 'Pay Range' (\$7500 to \$25000), 'Setting' (On-site, Hybrid, Remote), 'Skills' (Communication), and 'Required Degrees' (Bachelor of Science). The results section shows 'Filtered Ads' with 2 results. One result is for an 'HR Manager' at Coca-Cola Atlanta, Georgia, USA, with a pay range of \$7500-10000/month. The result card includes a preview of the job description, application count (4), and view count (864). The status is 'Open'.

Figure 8

**The following query will be used for search:**

```

SELECT *
FROM
(SELECT *
  FROM JobAdvertisement
 WHERE
    title LIKE "%Coca-Cola%"
    OR organization LIKE "%Coca-Cola%"),
WHERE
    DATEDIFF(month,created_at, GETDATE()) = 1
    AND type = "Job"
    AND pay_range_min >= 7500
    AND pay_range_max <= 25000
    AND location LIKE "%USA%"
    AND setting = "On-site"
    AND status = "Open"
    AND ad_id IN (
        (SELECT ad_id
         FROM
             DegreeInJobAdvertisement NATURAL JOIN Degree
         WHERE
             name = "Bachelor of Science")
        INTERSECT
        (SELECT ad_id
         FROM
             SkillInJobAdvertisement NATURAL JOIN Skill
         WHERE
             name = "Communication"))
);
  
```

## Scenario 2:

The screenshot shows the CareerBridge platform interface. On the left, there's a sidebar with options like Feed, Browse Ads (selected), All Ads, Ads For You, Applied Ads, Profile, and Settings. The main area has a search bar at the top right with placeholder text "Type to search ads by position or organization name...". Below the search bar are several filter buttons: "Hide Filtering Options" (disabled), "Apply Filters" (highlighted in blue), "Added" (radio buttons for Last day, Last week, Last month, Any time), "Ad Type" (radio buttons for Internship, Part-time Job, Job, Contract, Project-based, Internship is selected), "Pay Range" (a slider from \$4000 to \$5000), "Location" (USA), "Status" (radio buttons for Open, Closed, Closed is selected), "Domain" (Software Development). Under "Skills", there are checkboxes for C# (selected) and DevOps. Under "Required Degrees", there is a dropdown menu set to Bachelor of Science. The results section is titled "Filtered Ads" and shows one result: "Software Development Intern" at Microsoft Redmond, Washington, USA, posted 6 days ago, with a pay range of \$4000-4500/month. It includes tags for C#, DevOps, Team Work, and a status of Closed. A "View Ad Details" button is available. The bottom right of the results section shows application and view counts: Application Count: 932, View Count: 1647.

Figure 9

**The ads will be filtered based on the following query:**

```
SELECT *
FROM JobAdvertisement
WHERE
    DATEDIFF(month,created_at,GETDATE()) = 1
    AND type = "Internship"
    AND pay_range_min >= 4000
    AND pay_range_max <= 5000
    AND location LIKE "%USA%"
    AND setting = "On-site"
    AND status = "Closed"
    AND ad_id IN (
        (SELECT ad_id
        FROM
            DegreeInJobAdvertisement NATURAL JOIN Degree
        WHERE
```

```

        name = "Bachelor of Science")
INTERSECT
    (SELECT ad_id
     FROM
        SkillInJobAdvertisement SIJ,
        Skill S1,
        Skill S2
    WHERE
        SIJ.skill_id = S1.skill_id
        AND SIJ.skill_id = S2.skill_id
        AND S1.name = "C#" AND S2.name = "DevOps"));

```

**Again, when a sorting method is selected then a similar query with ORDER BY clause will be executed accordingly:**

```

SELECT *
FROM JobAdvertisement
WHERE
    DATEDIFF(month, created_at, GETDATE()) = 1
    AND type = "Internship"
    AND pay_range_min >= 4000
    AND pay_range_max <= 5000
    AND location LIKE "%USA%"
    AND setting = "On-site"
    AND status = "Closed"
    AND ad_id IN (
        (SELECT ad_id
         FROM
            DegreeInJobAdvertisement NATURAL JOIN Degree
         WHERE
            name = "Bachelor of Science")
INTERSECT
    (SELECT ad_id
     FROM
        SkillInJobAdvertisement SIJ,
        Skill S1,
        Skill S2
    WHERE
        SIJ.skill_id = S1.skill_id
        AND SIJ.skill_id = S2.skill_id
        AND S1.name = "C#" AND S2.name = "DevOps"))
ORDER BY created_at DESC;

```

However, as can be seen some filtering or searching options might be left empty depending on the user's decision, or, for example, the user can supply more than one filtering constraint for skill. In that case, we will construct the queries depending on those selected options benefiting the if-else logic from the high level implementation language (i.e., Python). We have just demonstrated some example scenarios above.

## 4.7 Sample Ad Description

The screenshot shows the CareerBridge platform interface. On the left, there's a sidebar with navigation options: Feed, Browse Ads (selected), All Ads, Ads For You, Applied Ads, Profile, and Settings. The main content area displays a job listing for a "Data Science Intern" at Facebook. The listing includes the following details:

- Pursuing Or Having One Of The Following Degrees Is Required:** Bachelor of Science, Master of Science
- Required Skills:** Analytical Thinking, Attention, Python, R, Tableau, Team Work
- Posted on:** 25 March 2023 (22 days ago)
- Software Development - Internship Ad - Hybrid - Open**
- Salary:** \$4000-5000/Month
- Apply** button
- Recruiter:** Felicia Jackson, Recruiter, HR Manager @ Facebook
- Send Message** button
- Responsibilities:**
  - Work on data-driven projects from start to finish while meeting technical and creative requirements.
  - Collaborate with data scientists, engineers, and product managers to iterate rapidly and deliver innovative solutions.
  - Analyze data using statistical methods to derive insights and make data-driven decisions.
  - Work with large datasets and develop algorithms for predictive modeling, data classification, and data clustering.
  - Participate in regular data science reviews and other team-wide data science efforts; contribute to a great data science team culture.
  - Stay up-to-date with the latest data science technologies and techniques, and share knowledge with team members.
  - Document data science workflows and share best practices with team members.
- Qualifications and Skills:**
  - Pursuing or having a Bachelor's or Master's degree in Data Science, Statistics, Computer Science, or a related field.
  - Strong analytical skills and attention to detail.
  - Experience working with programming languages such as Python or R, and data science libraries such as pandas, numpy, or scikit-learn.
  - Familiarity with data visualization tools such as Tableau, Power BI, or D3.js.
  - Ability to work independently and as part of a team.
  - Strong communication and collaboration skills.
- Preferred Qualifications and Skills:**
  - Experience with machine learning algorithms, such as supervised learning, unsupervised learning, or reinforcement learning.
  - Knowledge of databases, SQL, and big data technologies.
  - Experience with cloud computing platforms, such as AWS or Google Cloud.
  - Strong problem-solving skills and ability to think creatively to develop innovative solutions.
  - Experience working with social media data or other unstructured data sources.
- Facebook Software Development** logo and company information: Facebook is an online social media and social networking service owned by American technology giant Meta Platforms. Created in 2004 by Mark Zuckerberg with fellow Harvard College students and roommates Eduardo Saverin, Andrew McCollum, Dustin Moskovitz, and Chris Hughes. Its name derives from... [see more](#)

Figure 10

**Inputs:** @ad\_id

**The related attributes of an ad will be fetched based on the following query, these include the description, required skills, required degrees, application count, view count and more:**

```
SELECT *
FROM JobAdvertisement
WHERE ad_id = @ad_id;
```

**The related attributes of the recruiter will be fetched based on the following query:**

```
SELECT
  User.first_name,
  User.last_name,
  AppUser.user_role,
```

```

Experience.title,
WorkExperience.company_name
FROM
User
INNER JOIN JobAdvertisement ON User.user_id =
JobAdvertisement.user_id
INNER JOIN AppUser ON User.user_id = AppUser.user_id
INNER JOIN Profile ON User.user_id = Profile.user_id
INNER JOIN Experience ON Profile.profile_id =
Experience.profile_id
LEFT OUTER JOIN WorkExperience ON Experience.experience_id =
WorkExperience.experience_id
WHERE
Profile.is_application_specific IS FALSE
AND JobAdvertisement.ad_id = @ad_id
AND end_date IS NULL
AND status = "Working"
ORDER BY
start_date DESC
LIMIT 1;

```

**The related attributes of the company giving the ad will be fetched based on the following query:**

```

SELECT *
FROM
Company C,
JobAdvertisement J
WHERE
J.ad_id = @ad_id
AND C.company_name = J.organization;

```

**Update the application view count when users enters to the ad description page:**

```
CALL increase_view_count(@ad_id);
```

## 4.8 Ad Application Pages

### 4.8.1 Personal Information

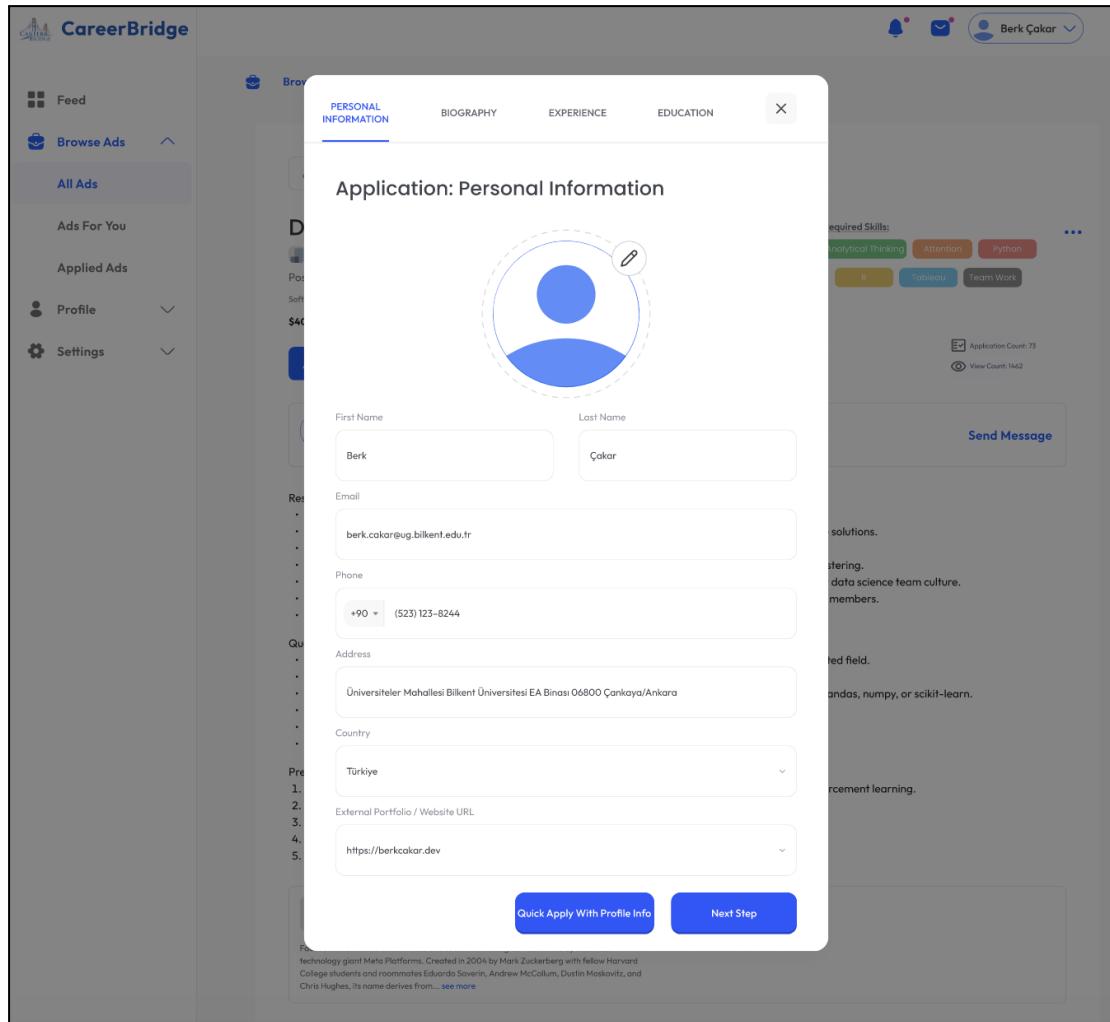


Figure 11

**Inputs:** @avatar, @first\_name, @last\_name, @phone\_number, @email, @address, @country, @external\_portfolio\_url

## 4.8.2 Biography

The screenshot shows the CareerBridge platform interface. On the left, there's a sidebar with options like Feed, Browse Ads, All Ads, Ads For You, Applied Ads, Profile, and Settings. The main area shows a job listing for a 'Data Science Intern' with a salary of '\$40k - \$50k'. The job description mentions 'Pursuing Or Having One Of The Following Degrees Is Required:' and lists 'Required Skills:'. A modal window titled 'Application: Biography' is open in the center, containing a rich text editor with a toolbar (A, B, I, U, etc.) and a placeholder 'Type something'. Below the editor, there's a list of requirements: 1. Knowledge of Python, 2. Knowledge of databases, SQL, and big data technologies, 3. Experience with cloud computing platforms, such as AWS or Google Cloud, 4. Strong problem-solving skills and ability to think creatively to develop innovative solutions, 5. Experience working with social media data or other unstructured data sources. At the bottom of the modal is a blue 'Next Step' button. In the bottom right corner of the main page, there's a 'Send Message' button. The top right corner shows a user profile for 'Berk Çakar'.

Figure 12

**Inputs:** @biography

### 4.8.3 Experience

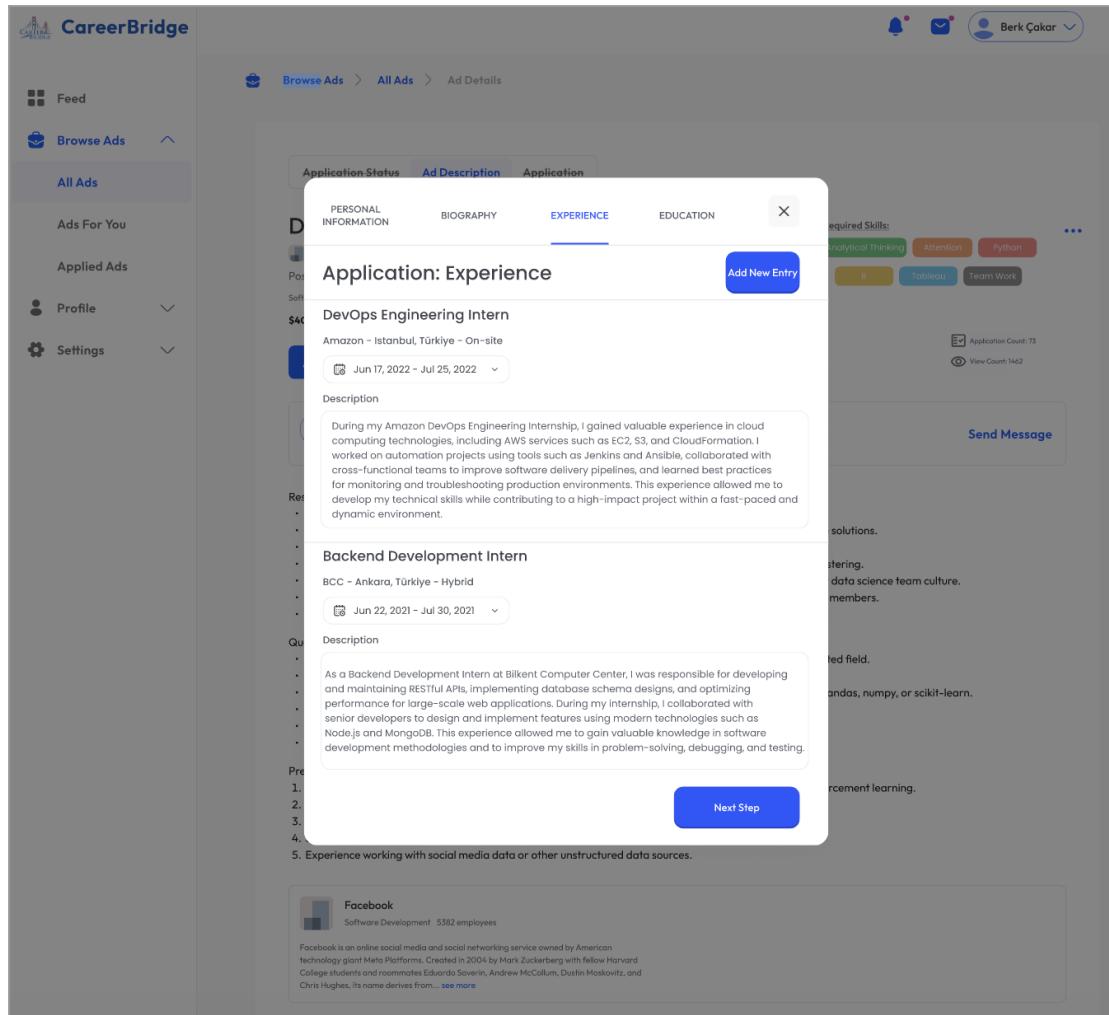


Figure 13

**Inputs:** @work\_title1, @work\_start\_date1, @work\_end\_date1,  
 @work\_current\_status1, @work\_description1, @work\_location1,  
 @work\_type1, @work\_status1, @work\_setting1, @work\_company\_name1  
 @work\_title2, @work\_start\_date2, @work\_end\_date2, @current\_status2,  
 @work\_description2, @work\_location2, @work\_type2, @work\_status2,  
 @work\_setting2, @work\_company\_name2

#### 4.8.4 Education

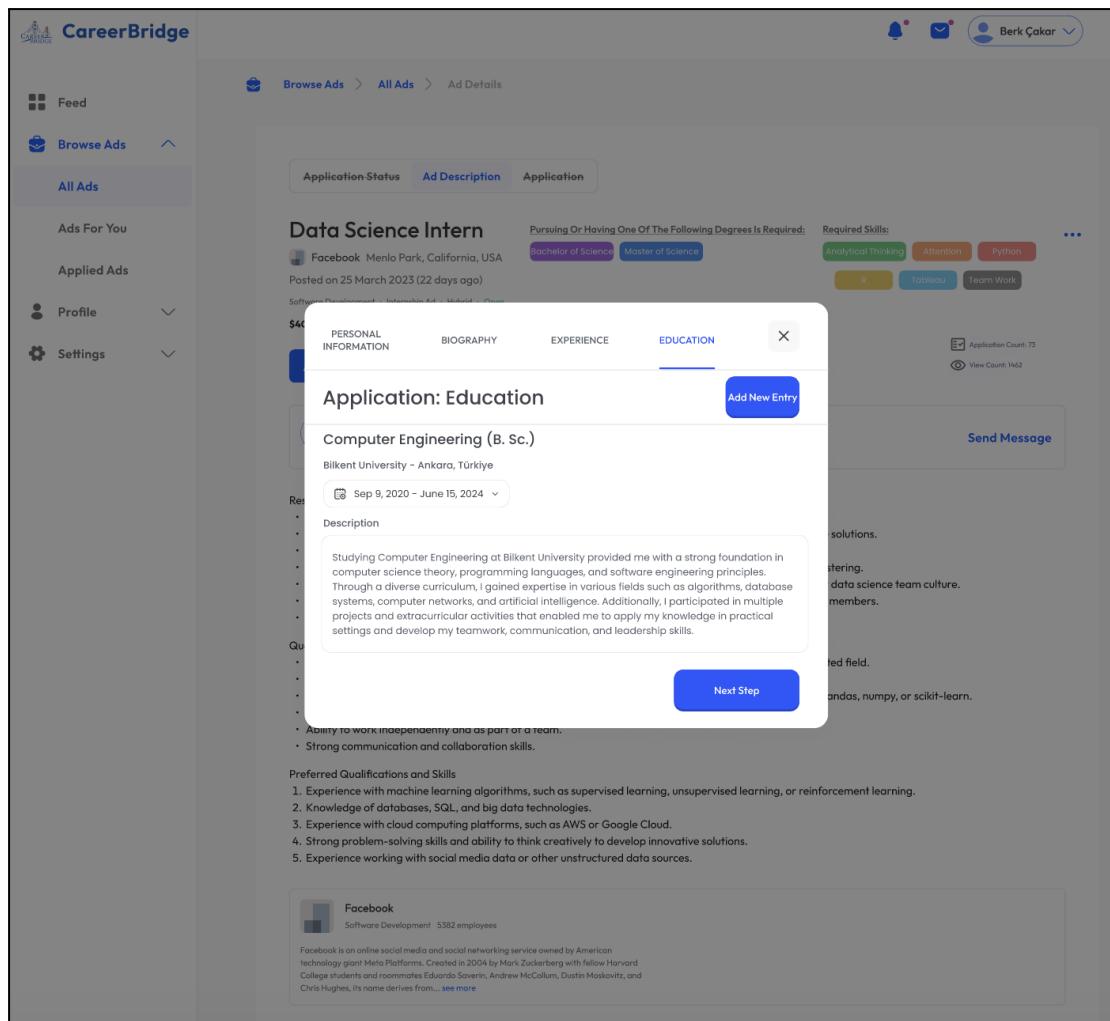


Figure 14

**Inputs:** @education\_title, @education\_start\_date, @education\_end\_date, @education\_current\_status, @education\_degree\_name, @education\_description, @education\_location, @education\_field\_of\_study, @education\_school

#### 4.8.5 Voluntary Work

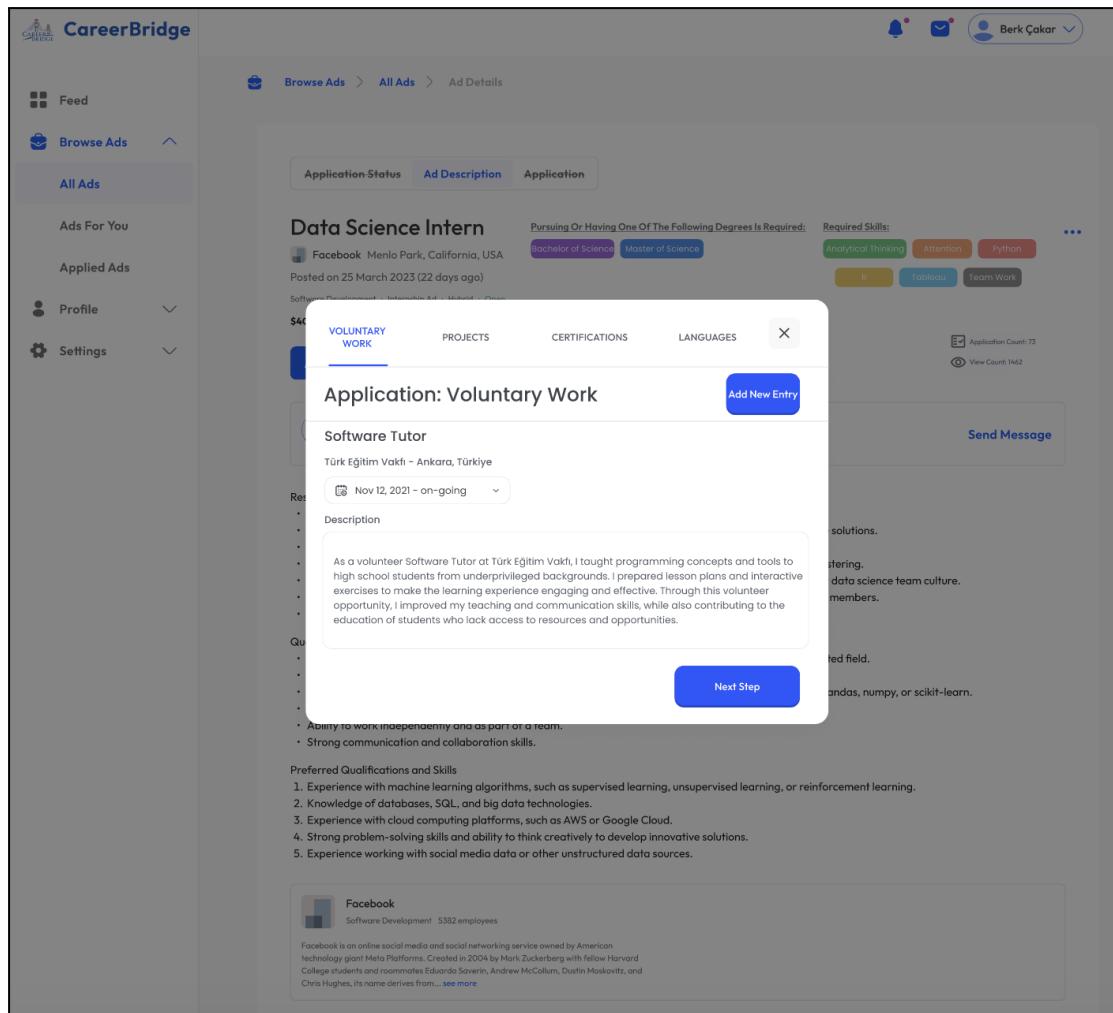


Figure 15

**Inputs:** @voluntary\_title, @voluntary\_start\_date, @voluntary\_end\_date, @voluntary\_current\_status, @voluntary\_description, @voluntary\_location, @voluntary\_organization\_name, @voluntary\_responsibility

#### 4.8.6 Projects

The screenshot shows the CareerBridge application interface. On the left, there's a sidebar with 'Feed', 'Browse Ads' (selected), 'All Ads', 'Ads For You', 'Applied Ads', 'Profile', and 'Settings'. The main area has a breadcrumb navigation: 'Browse Ads > All Ads > Ad Details'. A modal window titled 'Application: Projects' is open, showing two project entries: 'ERSMS' and 'LabConnect'. Each entry includes a URL, a date range, and a detailed description. Below the entries is a 'Preferred Qualifications and Skills' section with a numbered list. At the bottom of the modal is a 'Next Step' button. The top right corner of the main window shows a user profile for 'Berk Çakar'.

Figure 16

**Inputs:** @project\_title1, @project\_start\_date1, @project\_end\_date1,  
@project\_url1, @project\_description1, @project\_title2,  
@project\_start\_date2, @project\_end\_date2, @project\_url2,  
@project\_description2

#### 4.8.7 Certifications

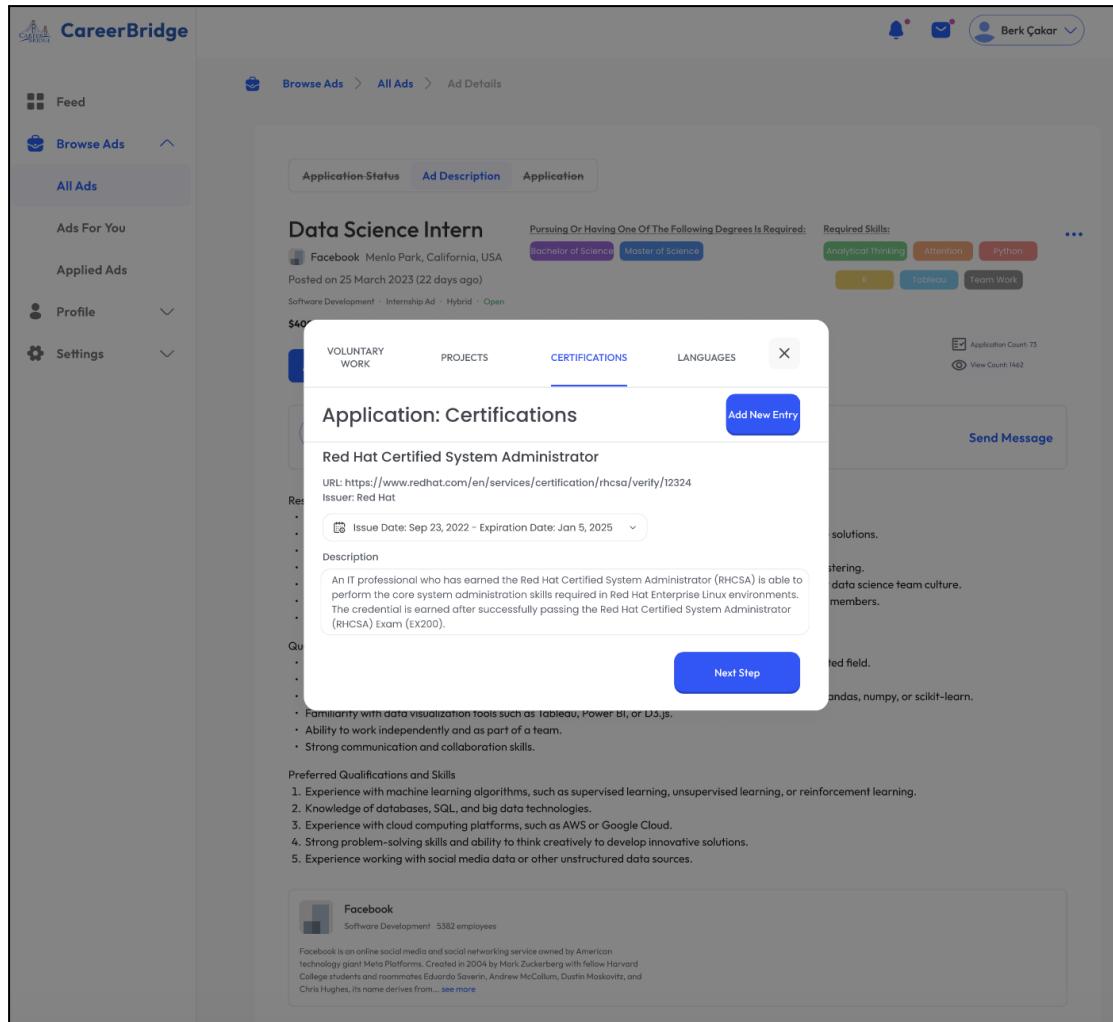


Figure 17

**Inputs:** @certification\_name, @certification\_description, @certification\_credential\_url, @certification\_issue\_date, @certification\_issuer, @certification\_expiration\_date

#### 4.8.8 Awards

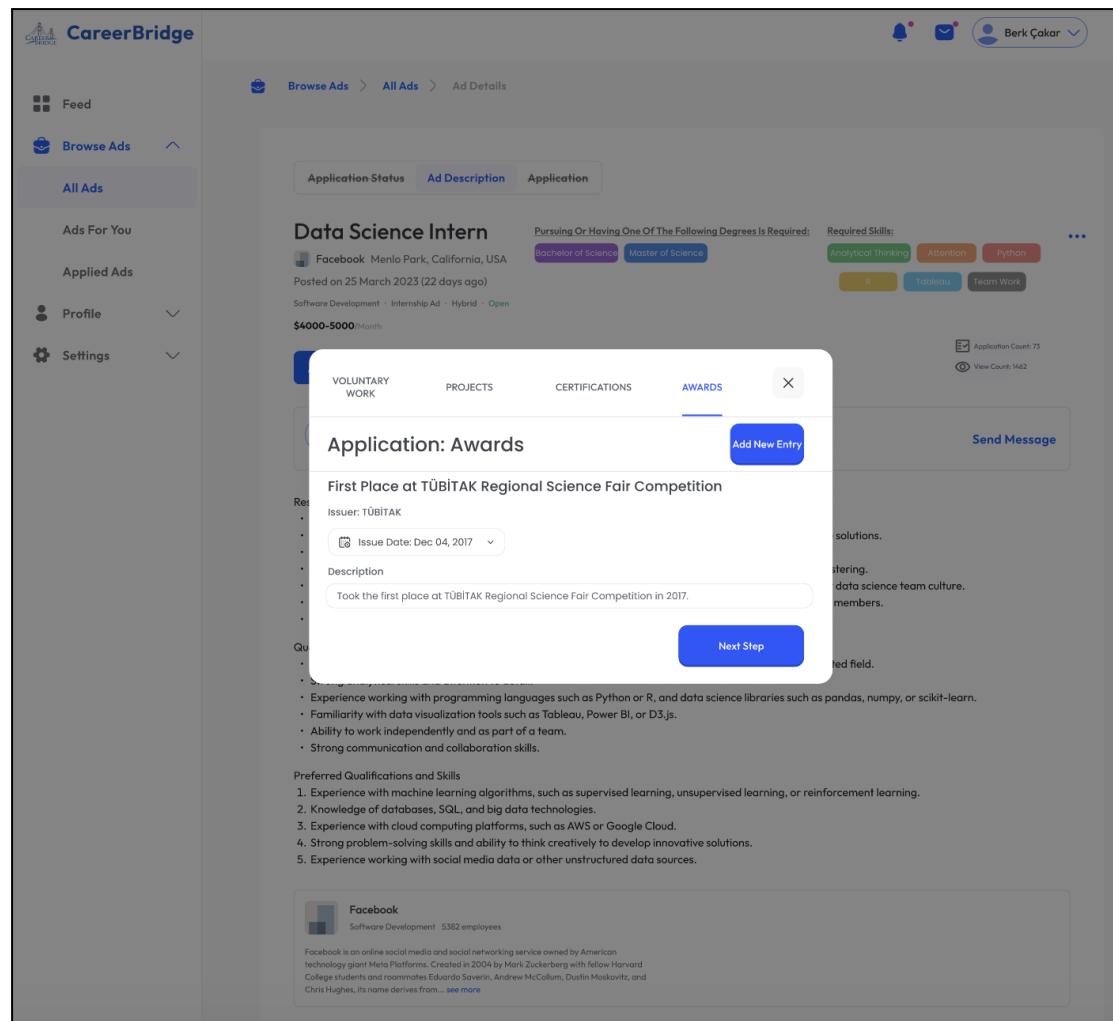


Figure 18

**Inputs:** @award\_title, @award\_description, @award\_issuer, @award\_issue\_date

#### 4.8.9 Test Scores

The screenshot shows the CareerBridge application interface. On the left, there's a sidebar with options like Feed, Browse Ads, All Ads, Ads For You, Applied Ads, Profile, and Settings. The main area displays a job listing for a Data Science Intern at Facebook in Menlo Park, California, USA. The listing specifies a salary range of \$4000-5000/Month. Below the job listing, a modal window is open titled "Application: Test Scores". The modal has tabs for TEST SCORES, LANGUAGES, SKILLS, and RESUME. The TEST SCORES tab is selected, showing a section for TOEFL with a score of 108, an attachment named "toefl\_result.pdf", and a note about the issue date being December 04, 2017. There's also a text input field containing the text "I scored 108 in TOEFL English proficiency exam.". A blue "Add New Entry" button is visible in the top right of the modal. In the bottom right corner of the modal, there's a "Next Step" button. The background of the main page shows other job listings and user profile information.

Figure 19

**Inputs:** @test\_name, @test\_date, @test\_score, @test\_attachment, @test\_date, @test\_description

#### 4.8.10 Languages

The screenshot shows the CareerBridge platform interface. On the left, there's a sidebar with options like Feed, Browse Ads, All Ads (selected), Ads For You, Applied Ads, Profile, and Settings. The main area displays a job listing for a 'Data Science Intern' at Facebook. The job requires a Bachelor's or Master's degree in fields like Mathematics, Computer Science, or a related field. It lists 'Analytical Thinking', 'Attention', 'Python', 'R', 'Tableau', and 'Team Work' as required skills. The salary is \$4000-5000/Month. Below the job description, a modal window is open, titled 'Application: Languages'. It shows two entries: 'English' with 'Proficiency: C1' and 'Turkish' with 'Proficiency: C2'. There are tabs for TEST SCORES, LANGUAGES (selected), SKILLS, and RESUME. A blue button labeled 'Add New Entry' is visible. To the right of the modal, there's a 'Send Message' button. At the bottom of the modal, there's a 'Next Step' button.

Figure 20

**Inputs:** @language\_name1, @language\_proficiency1,  
 @language\_name2, @language\_proficiency2

#### 4.8.11 Skills

The screenshot shows the CareerBridge application interface. On the left, there's a sidebar with navigation options: Feed, Browse Ads (selected), All Ads, Ads For You, Applied Ads, Profile, and Settings. The main content area shows a job listing for a Data Science Intern at Facebook. The listing requires Bachelor of Science or Master of Science degrees and lists required skills like Analytical Thinking, Attention, Python, R, Tableau, and Team Work. It also shows application statistics: Application Count 75 and View Count 1462. A modal window titled "Application: Skills" is open, showing a search bar and a list of added skills: Statistics, Team Work, Python, Numpy, AI, and Scikit. There are buttons for "Add New Entry" and "Next Step". Below the modal, there's a section for Preferred Qualifications and Skills, which includes a list of requirements and a note about experience with Python and data science libraries. At the bottom, there's a Facebook company card.

Figure 21

**Inputs:** @skill\_name1, @skill\_name2, @skill\_name3, @skill\_name4, @skill\_name5, @skill\_name6

#### 4.8.12 Upload Resume

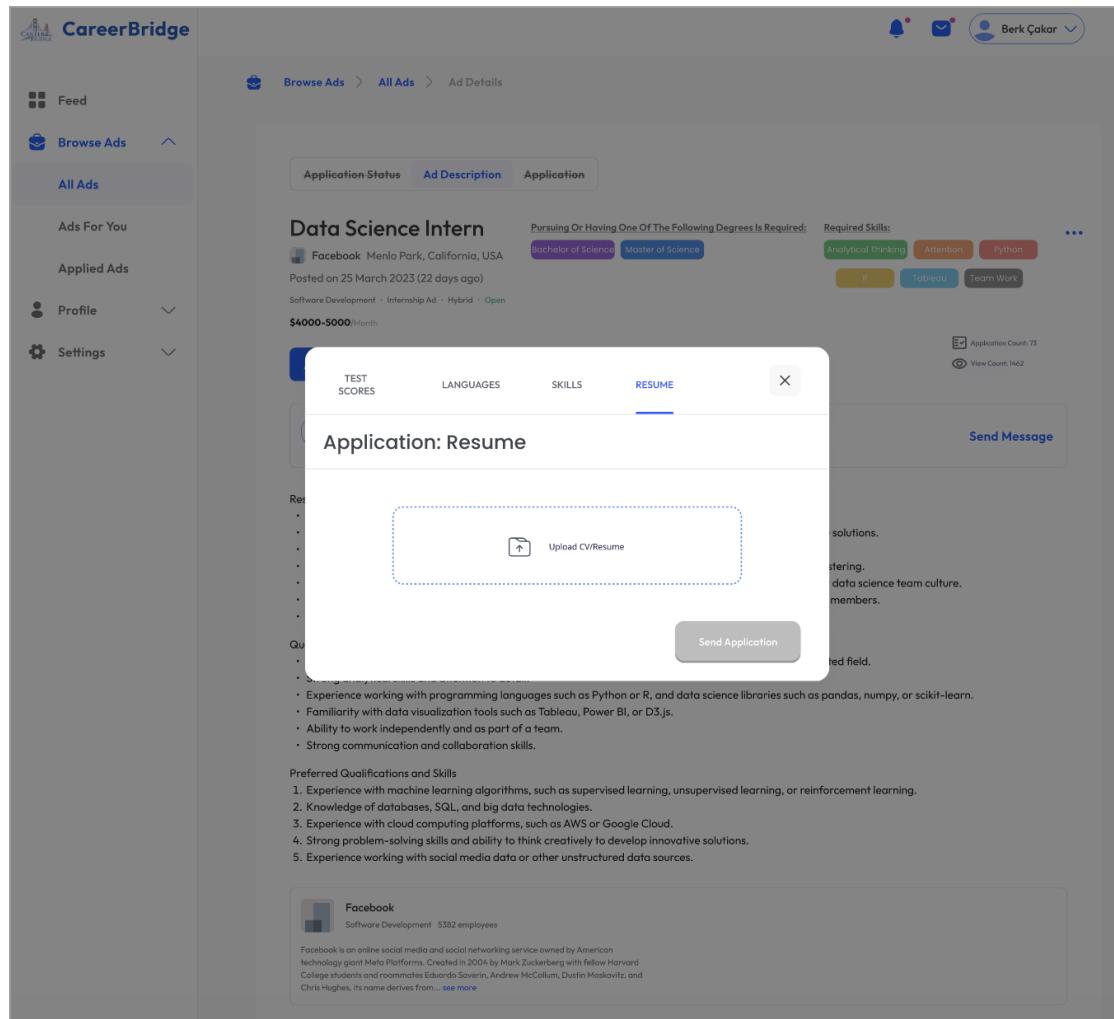


Figure 22

#### 4.8.13 Uploaded Resume

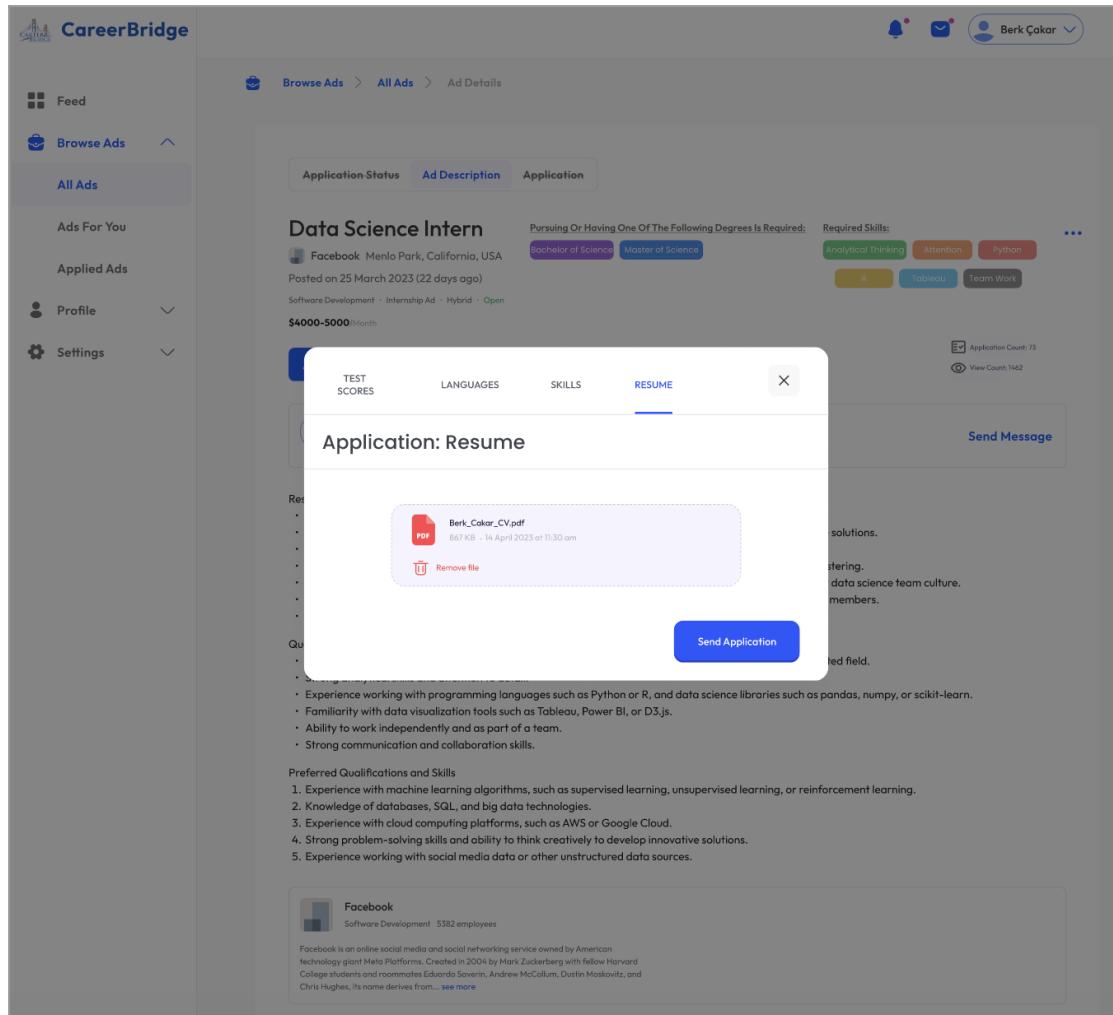


Figure 23

**Inputs:** @resume

**SQL query on Quick Apply with User Profile Info:**

```
INSERT INTO JobAdvertisementResponse(profile_id, ad_id,
apply_date)
SELECT
    profile_id,
    @ad_id,
    NOW()
FROM
    Profile
WHERE
    user_id = @user_id
    AND is_application_specific IS FALSE;
```

**SQL query on Submit of Job Application (all inputs from the Section 4.8 are used):**

```
INSERT INTO PROFILE(
    avatar, country, external_portfolio_url,
    address, biography, is_private, resume,
    phone_number, is_application_specific,
    created_at
)
SELECT
    @avatar,
    @country,
    @external_portfolio_url,
    @address,
    @biography,
    0,
    @resume,
    @phone_number,
    1,
    NOW();
```

**Now assume the ID of the profile created is @profile\_id:**

```
INSERT INTO Experience(
    profile_id, start_date, end_date,
    title, description, current_status,
    location
)
VALUES
    (
        @profile_id, @work_start_date1, @work_end_date1,
        @work_title1, @work_description1,
        @work_status1, @work_location1
    ),
    (
        @profile_id, @work_start_date2, @work_end_date2,
        @work_title2, @work_description2,
        @work_status2, @work_location2
    ),
    (
        @profile_id, @education_start_date,
        @education_end_date, @education_title,
        @education_description, @education_current_status,
        @education_location
    ),
    (
        @profile_id, @voluntary_start_date,
        @voluntary_end_date, @voluntary_title,
        @voluntary_description, @voluntary_current_status,
        @voluntary_location);
```

**Now assume the ID of the experiences created are @experience\_id1, @experience\_id2, @experience\_id3 and @experience\_id4, respectively. Also, the corresponding entries of the NonProfitOrganization, Company and School already exist in the database:**

```
INSERT INTO WorkExperience(experience_id, setting, type)
VALUES
(
    @experience_id1, @work_setting1,
    @work_type1, @work_company_name1
),
(
    @experience_id2, @work_setting2,
    @work_type2, @work_company_name2
);
```

#### **SQL query to insert into Degree:**

```
INSERT INTO Degree(name) VALUES (@education_degree_name);
```

**Now assume the ID of the degree created is @degree\_id. SQL query to insert into EducationalExperience:**

```
INSERT INTO EducationalExperience(
    experience_id, grade, field_of_study,
    school_name, degree_id
)
VALUES
(
    @experience_id3, @education_field_of_study,
    @education_school, @degree_id
);
```

#### **SQL query to insert into VoluntaryExperience:**

```
INSERT INTO VoluntaryExperience(
    experience_id, responsibility, organization_name
)
VALUES
(
    @experience_id4, @voluntary_responsibility,
    @voluntary_organization_name
);
```

#### **SQL query to insert into Project:**

```
INSERT INTO Project(
    profile_id, title, description, start_date,
    end_date, project_url
```

```

)
VALUES
(
    @profile_id, @project_title1, @project_description1,
    @project_start_date1, @project_end_date1,
    @project_url1
),
(
    @profile_id, @project_title2, @project_description2,
    @project_start_date2, @project_end_date2,
    @project_url2
);

```

**SQL query to insert into Certification:**

```

INSERT INTO Certification(
    profile_id, certification_name, description,
    credential_url, issue_date, issuer,
    expiration_date
)
VALUES
(
    @profile_id, @certification_name,
    @certification_description,
    @certification_credential_url,
    @certification_issue_date, @certification_issuer,
    @certification_expiration_date
);

```

**SQL query to insert into Award:**

```

INSERT INTO Award(
    profile_id, title, description, issue_date,
    issuer
)
VALUES
(
    @profile_id, @award_title, @award_description,
    @award_issuer, @award_issue_date
);

```

**SQL query to insert into TestScore:**

```

INSERT INTO TestScore(
    profile_id, test_name, description,
    test_date, score, attachment
)
VALUES
(
    @profile_name, @test_name, @test_description,

```

```
@test_date, @test_score, @test_attachment);
```

### **SQL query to insert into LanguageProficiency:**

```
INSERT INTO LanguageProficiency(
    profile_id, language_name, proficiency
)
VALUES
(
    @profile_id, @language_name1, @language_proficiency1
),
(
    @profile_id, @language_name2, @language_proficiency2
);
```

### **SQL query to insert into Skill:**

```
INSERT INTO Skill(profile_id, name)
VALUES
    (@profile_id, @skill_name1),
    (@profile_id, @skill_name2),
    (@profile_id, @skill_name3),
    (@profile_id, @skill_name4),
    (@profile_id, @skill_name5),
    (@profile_id, @skill_name6);
```

### **SQL query to insert into JobAdvertisementResponse:**

```
INSERT INTO JobAdvertisementResponse(profile_id, ad_id,
apply_date)
SELECT
    @profile_id,
    @ad_id,
    NOW();
```

### **After the application, update the application\_count:**

```
CALL increase_application_count(@ad_id);
```

Similar to the section where we perform filtering, the number of possible queries is infinite. In this series of queries we handled a specific case with the given inputs to apply an advertisement. Again, with the help of a high level programming language's capabilities we can construct the corresponding queries based on the filled fields or number of elements in a field (for example multiple skills or work experiences may appear). In a similar way, users will be warned to fill in the fields which do not have a default value or set not to be NULL.

## 4.9 Applied Ads Page

The screenshot shows the 'CareerBridge' application interface. On the left, there's a sidebar with navigation options: 'Feed', 'Browse Ads' (which is currently selected), 'All Ads', 'Ads For You', and 'Applied Ads'. Below these are 'Profile' and 'Settings' sections. The main content area is titled 'Applied Ads' and shows a single job listing for a 'Data Science Intern' at Facebook in Menlo Park, California, USA. The listing indicates the user applied on April 14, 2023. It includes tabs for 'Application Status', 'Job Description', and 'Application'. The 'Job Description' tab is active, displaying details like the required degree ('Bachelor of Science, Master of Science'), required skills ('Analytical thinking, Attention, Python, R, Tableau, Team Work'), responsibilities, qualifications, and preferred qualifications. A message button is also present. The right side of the header shows notifications and a user profile for 'Berk Çakar'.

Figure 24

**Inputs:** @user\_id, @ad\_id

**SQL query to get all ads that user applied ordered by their application date:**

```
SELECT *
FROM
    JobAdvertisementResponse
WHERE
    user_id = @user_id
ORDER BY
    apply_date DESC;
```

**SQL query to get the info of the selected ad:**

```
SELECT *
FROM
    JobAdvertisement JA,
    JobAdvertisementResponse JAR
WHERE
    JA.ad_id = JAR.ad_id
    AND ja.ad_id = @ad_id
    AND JAR.user_id = @user_id;
```

**SQL query to get the recruiter info of the selected ad:**

```
SELECT
    User.first_name,
    User.last_name,
    AppUser.user_role,
    Experience.title,
    WorkExperience.company_name
FROM
    User
    INNER JOIN JobAdvertisement ON User.user_id =
JobAdvertisement.user_id
    INNER JOIN AppUser ON User.user_id = AppUser.user_id
    INNER JOIN Profile ON User.user_id = Profile.user_id
    INNER JOIN Experience ON Profile.profile_id =
Experience.profile_id
    LEFT OUTER JOIN WorkExperience ON Experience.experience_id =
WorkExperience.experience_id
WHERE
    Profile.is_application_specific IS FALSE
    AND JobAdvertisement.ad_id = @ad_id
    AND end_date IS NULL
    AND status = "Working"
ORDER BY
    start_date DESC
LIMIT
    1;
```

## 4.10 Published Ads Page

The screenshot shows the 'CareerBridge' platform's 'Published Ads' section. On the left, there's a sidebar with 'Feed', 'Browse Ads', and 'Manage Your Ads' (which is currently selected). Below that are 'Profile' and 'Settings'. The main area displays 'Your Published Ads' with a search bar at the top. There are two ads listed:

- Data Science Intern** (Facebook, Menlo Park, California, USA) - Python, R, Attention. Status: Open. Description Preview: As a Data Science Intern at Facebook, you will have the opportunity to work on some of the most challenging and impactful problems in the tech industry. You will be part of a team that is responsible for leveraging data and analytics to drive business decisions and improve the user experience for billions of people around the world. You will work closely with our data science tea... Application Count: 73 View Count: 1462.
- Software Engineer** (Facebook, Menlo Park, California, USA) - Backend Dev., Database Systems, Cloud Platforms. Software Development: Job Ad - On-site - Closed. Description Preview: As a Software Engineer on Facebook's Ads Ranking team, you will be at the forefront of innovation in the advertising industry. You will work on cutting-edge machine learning models and algorithms that power Facebook's advertising platform, and you will be responsible for building and improving these models to deliver the most relevant ads to our users. You will work closely with other engine... Application Count: 33 View Count: 467.

At the bottom right of the main area, there is a small number '1' inside a blue-bordered box.

Figure 25

**Inputs:** @user\_id, @ad\_id

**SQL query to get all ads that is published by the user logged in:**

```
SELECT *
FROM JobAdvertisement
WHERE
    user_id = @user_id
ORDER BY
    created_at DESC;
```

**SQL query to get 3 skills for each published ad:**

```
SELECT name
FROM SkillInJobAdvertisement NATURAL JOIN Skill
WHERE ad_id = @ad_id
LIMIT 3;
```

## Delete of a published ad by clicking on the delete button:

```
DELETE FROM JobAdvertisement  
WHERE ad_id = @ad_id;
```

## 4.11 Candidates (Applicants) of an Ad Page

The screenshot shows the 'Manage Your Ads' section of the CareerBridge application. The current view is 'Data Science Intern > Candidates'. There are 73 candidates listed, sorted by 'Most Recent'. Each candidate card displays their profile picture, name, status (e.g., 'Not Viewed', 'Approved', 'Interview'), contact information (email and phone number), and four action buttons: 'Application', 'Respond', 'Profile', and 'Message'. The bottom of the page shows a navigation bar with page numbers from 1 to 13.

Figure 26

**Inputs:** @ad\_id, @profile\_id, @response, @search\_input

## SQL query to fetch the applicants and their relevant information:

```
SELECT  
    P.avatar,  
    P.phone_number,  
    U.first_name,  
    U.last_name,  
    U.email,  
    JR.response,  
    JR.user_id  
FROM  
    JobAdvertisementResponse JR,  
    Profile P,  
    User U  
WHERE  
    JR.user_id = U.user_id
```

```
AND JR.ad_id = @ad_id  
AND U.user_id = P.user_id;
```

**SQL query to sort the applicants when a sorting option is selected, for example most recent first:**

```
SELECT  
    P.avatar,  
    P.phone_number,  
    U.first_name,  
    U.last_name,  
    U.email,  
    JR.response,  
FROM  
    JobAdvertisementResponse JR,  
    Profile P,  
    User U  
WHERE  
    JR.user_id = U.user_id  
    AND JR.ad_id = @ad_id  
    AND U.user_id = P.user_id  
ORDER BY  
    JR.apply_date DESC;
```

**SQL query to retrieve the candidates when the candidates are searched by their names:**

```
SELECT  
    P.avatar,  
    P.phone_number,  
    U.first_name,  
    U.last_name,  
    U.email,  
    JR.response,  
FROM  
    JobAdvertisementResponse JR,  
    Profile P,  
    (SELECT *  
     FROM User  
     WHERE  
         CONCAT(first_name, ' ', last_name) LIKE "%@search_input%"  
    ) as U  
WHERE  
    JR.user_id = U.user_id  
    AND JR.ad_id = @ad_id  
    AND U.user_id = P.user_id  
ORDER BY  
    JR.apply_date DESC;
```

## SQL query when a recruiter respond the job application:

```
UPDATE JobAdvertisementResponse  
SET response = @response  
WHERE profile_id = @profile_id AND ad_id = @ad_id;
```

## 4.12 CareerBridge Specific Pages

### 4.12.1 Profile Page

The screenshot shows the CareerBridge profile page for a user named Berk Çakar. The left sidebar has a 'Profile' section selected, showing options like 'Information', 'Work Experience', and 'Skills'. The main content area displays Berk's basic information, work experience, and skills.

**Basic Information:**  
Update profile information  
Email Address: me@berkacakar.dev  
Phone Number: +90 (523) 123-8244  
Country: Türkiye  
Address: Üniversiteler Mahallesi Bilkent Üniversitesi EA Binası 06800 Çankaya/Ankara  
Website: <https://berkacakar.dev>

**Work Experience:**  
Add Experience  
**DevOps Engineering Intern**  
Amazon, İstanbul, Türkiye Jun 17, 2022 - Jul 25, 2022  
During my Amazon DevOps Engineering Internship, I gained valuable experience in cloud computing technologies, including AWS services such as EC2, S3, and CloudFormation. I worked on automation projects using tools such... [See More](#)

**Backend Development Intern**  
BCC, Ankara, Türkiye Jun 22, 2021 - Jul 30, 2021  
As a Backend Development Intern of Bilkent Computer Center, I was responsible for developing and maintaining RESTful APIs, implementing database schema designs, and optimizing performance for large-scale web applications. [See More](#)

Figure 27

**Inputs:** All variables prefixed with '\*' are inputs in this section.

Note that the queries to fetch all of the profile information are below although the mock-up demonstrates some part of it for simplicity.

### SQL query for retrieving basic information:

```
SELECT *  
FROM  
    User U  
JOIN Profile P ON U.user_id = P.profile_id  
WHERE  
    U.user_id = @user_id;
```

**SQL query to fetch the notifications of a user:**

```
SELECT * FROM Notification WHERE user_id = @user_id;
```

**SQL query to fetch the master skill of a user:**

```
SELECT *
FROM
    Skill S,
    Profile P,
    SkillAssesment SA
WHERE
    S.profile_id = P.skill_id
    AND P.user_id = @user_id
    AND S.is_master_skill = 1
    AND SA.skill_id = S.skill_id
LIMIT 1;
```

**SQL query to fetch the work experiences of a user:**

```
SELECT *
FROM
    WorkExperience WE,
    Experience E,
    Profile P
WHERE
    WE.experience_id = E.experience_id
    AND E.profile_id = P.profile_id
    AND P.user_id = @user_id;
```

**SQL query for retrieving biography of a user:**

```
SELECT biography from Profile P where P.user_id = @user_id;
```

**SQL query to fetch the educational experiences of a user:**

```
SELECT *
FROM
    EducationalExperience EE,
    Experience E,
    Profile P
WHERE
    P.user_id = @user_id
    AND P.profile_id = E.profile_id
    AND E.experience_id = EE.experience_id;
```

**SQL query to fetch the voluntary experiences of a user:**

```
SELECT *
FROM
    VoluntaryExperience VE,
    Experience E,
    Profile P
WHERE
    P.user_id = @user_id
    and P.profile_id = E.profile_id
    AND E.experience_id = VE.experience_id;
```

**SQL query to fetch the projects of a user:**

```
SELECT *
FROM
    Project,
    Profile,
WHERE
    Profile.user_id = @user_id
    AND Profile.profile_id = Project.profile_id;
```

**SQL query to fetch the certifications of a user:**

```
SELECT *
FROM
    Certification C,
    Profile P
WHERE
    P.user_id = @user_id
    AND P.profile_id = C.profile_id;
```

**SQL query to fetch the awards of a user:**

```
SELECT *
FROM
    User U
    JOIN Profile P ON U.user_id = P.profile_id
    JOIN Awards A ON P.profile_id = A.profile_id
WHERE
    U.user_id = @user_id;
```

**SQL query to fetch the test scores of a user:**

```
SELECT *
FROM
    TestScore T,
    Profile P
WHERE
    P.user_id = @user_id
```

```
        AND P.profile_id = T.profile_id;
```

**SQL query to fetch the language proficiencies of a user:**

```
SELECT
  *
FROM
  LanguageProficiency L,
  Profile P
WHERE
  P.user_id = @user_id
  AND P.profile_id = L.profile_id;
```

**SQL query to fetch the skills of a user:**

```
SELECT *
FROM
  Skill S,
  SkillAssesment SA,
  Profile P
WHERE
  P.user_id = @user_id
  AND P.profile_id = S.profile_id
  AND S.skill_id = SA.skill_id;
```

**SQL query to fetch the resume of a user:**

```
SELECT resume FROM Profile P WHERE P.user_id = @user_id;
```

**SQL query to delete an experience from a user:**

```
DELETE FROM Experience WHERE experience_id = @experience_id;
```

**SQL query to delete a certification from a user:**

```
DELETE FROM Certification WHERE certification_id =
@certification_id;
```

**SQL query to delete an award from a user:**

```
DELETE FROM Certification WHERE award_id = @award_id;
```

**SQL query to delete a test score from a user:**

```
DELETE FROM TestScore WHERE test_score_id = @test_score_id;
```

**SQL query to delete a publication from a user:**

```
DELETE FROM Publication WHERE publication_id =
@publication_id;
```

**SQL query to delete a language proficiency from a user:**

```
DELETE FROM LanguageProficiency WHERE language_id =  
@language_id;
```

**SQL query to delete a skill from a user:**

```
DELETE FROM Skill WHERE skill_id = @skill_id;
```

**SQL query to delete the resume of a user:**

```
UPDATE Profile SET resume = NULL WHERE Profile.profile_id =  
@profile_id;
```

#### 4.12.2 Messages Page (Request for Skill Assessment)

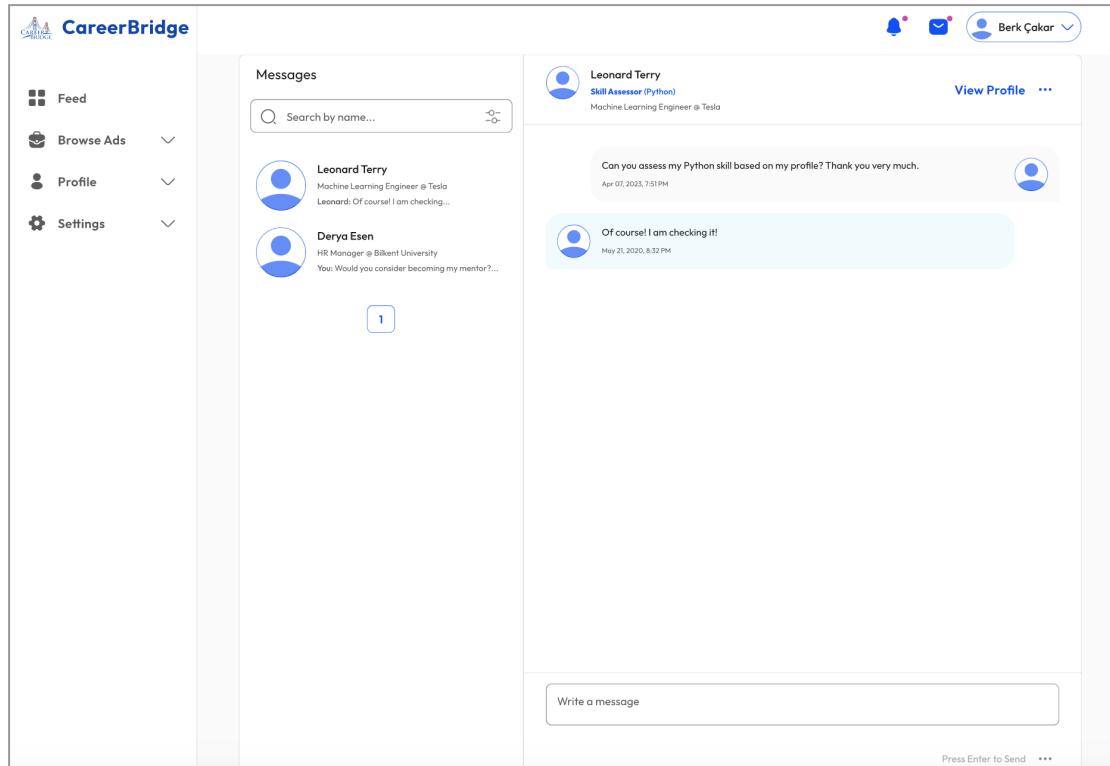


Figure 28

**Inputs:** @user\_id, @receiver\_id, @content, @file, @other\_user\_id, @skill\_id, @rating

**SQL query to send message when the thread already exists between two users:**

```

INSERT INTO Message(
    thread_id, content, file, sent_time,
    from_thread_creator)
SELECT
    thread_id,
    @content,
    @file,
    NOW(),
    CASE WHEN creator_id = @user_id THEN TRUE ELSE FALSE END
FROM
    (SELECT
        thread_id,
        creator_id
    FROM

```

```

    Thread T
WHERE
    T.creator_id = @user_id
    AND T.receiver_id = @receiver_id
UNION
SELECT
    thread_id,
    creator_id
FROM
    Thread T
WHERE
    T.creator_id = @receiver_id
    AND T.receiver_id = @user_id
);

```

**SQL query to send message when the thread does not exist between two users:**

```

INSERT INTO Thread(creator_id, receiver_id)
VALUES
    (@user_id, @receiver_id);

```

**Assume the ID of the thread created is @thread\_id:**

```

INSERT INTO Message(
    thread_id, content, file, sent_time,
    from_thread_creator
)
SELECT
    @thread_id,
    @content,
    @file,
    NOW(),
    TRUE;

```

**SQL query to load conversation:**

```

SELECT
    M.content,
    M.file,
    M.sent_time,
    M.has_read,
    M.from_thread_creator
FROM
    Message M JOIN Thread T ON M.thread_id = T.thread_id
WHERE
    (T.creator_id = @user_id
    AND T.receiver_id = @receiver_id)
    OR (
        T.creator_id = @receiver_id

```

```

        AND T.receiver_id = @sender_id
    );
UPDATE
    Message
SET
    has_read = TRUE
WHERE
    Message.message_id IN (
        SELECT M.message_id
        FROM
            Message M JOIN Thread T ON M.thread_id = T.thread_id
        WHERE
            (T.creator_id = @user_id
            AND T.receiver_id = @receiver_id)
        OR (
            T.creator_id = @receiver_id
            AND T.receiver_id = @sender_id)
    );

```

**SQL query to load all threads for constructing the list of previous conversations:**

```

SELECT
    U.user_id,
    U.first_name,
    U.last_name
FROM
    Thread T
    JOIN User U ON T.creator_id = U.user_id
WHERE
    T.receiver_id = @user_id
UNION
SELECT
    U.user_id,
    U.first_name,
    U.last_name
FROM
    Thread T
    JOIN User U ON T.receiver_id = U.user_id
WHERE
    T.creator_id = @user_id;

```

**SQL query to retrieve the information about the other side of the thread:**

```

SELECT
    User.first_name,
    User.last_name,
    AppUser.user_role,
    Experience.title,

```

```

WorkExperience.company_name
FROM
    User
        INNER JOIN AppUser ON User.user_id = AppUser.user_id
        INNER JOIN Profile ON User.user_id = Profile.user_id
        INNER JOIN Experience ON Profile.profile_id =
            Experience.profile_id
        LEFT OUTER JOIN WorkExperience ON Experience.experience_id =
            WorkExperience.experience_id
WHERE
    Profile.is_application_specific IS FALSE
    AND User.id = @other_user_id
    AND end_date IS NULL
    AND status = "Working"
ORDER BY
    start_date DESC
LIMIT
    1;

```

### **SQL query for Skill Assessors to assess a skill of the user:**

```

UPDATE SkillAssessment
SET rating = @rating
WHERE skill_id = @skill_id;

```

#### **4.12.3 Messages Page (Request for Mentorship & Appointment)**

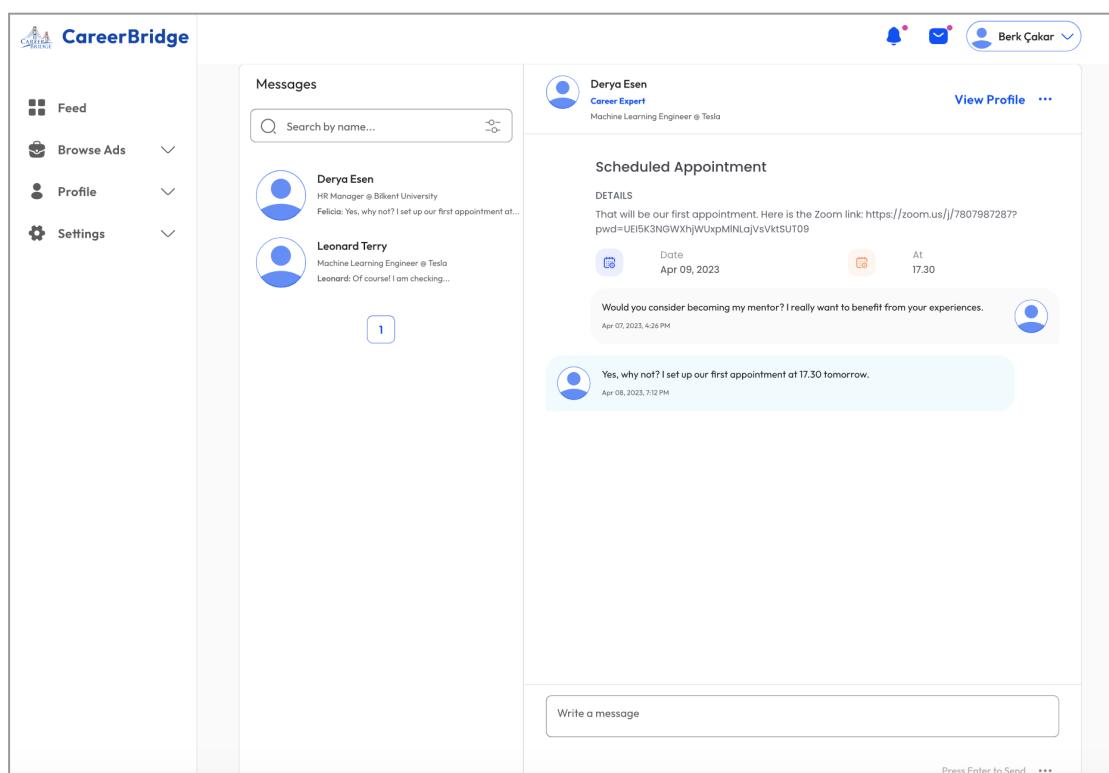


Figure 29

**Inputs:** @user\_id, @receiver\_id, @thread\_id, @date, @details, @response

**SQL query to create appointment (appears when clicked on the three dots):**

```
INSERT INTO Appointment(thread_id, date, details)
VALUES(@thread_id, @date, @details);
```

**SQL query to retrieve existing appointment:**

```
SELECT *
FROM Appointment
WHERE thread_id = @thread_id AND date = @date;
```

**SQL query to send mentorship request:**

```
INSERT INTO Mentorship(mentor_id, mentee_id)
VALUES (@user_id, @receiver_id);
```

**SQL query to respond to mentorship request:**

```
UPDATE Mentorship
SET response = @response
WHERE mentee_id = @user_id AND mentor_id = @receiver_id;
```

#### 4.12.4 Apply For a Role Page

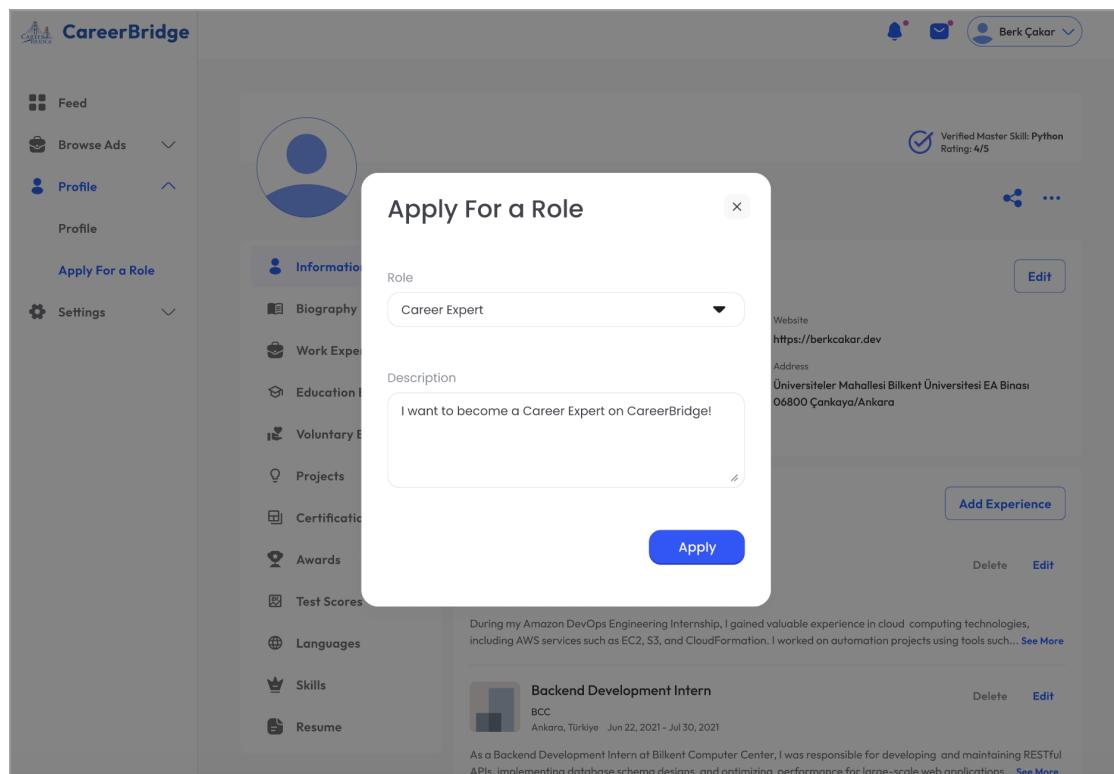


Figure 30

**Inputs:** @user\_id, @role, @description

**SQL query to apply for a role:**

```
INSERT INTO Application(
    description, application_type, application_date)
SELECT @description,
    @role,
    GETDATE();
```

**Assume the ID of application created is @application\_id:**

```
INSERT INTO ApplicationReview(@application_id)
VALUES(@application_id);
```

**SQL query to approve or reject an application for administrators  
(assume the ID of the administrator who makes the decision is @admin\_id, and the response is @response):**

```
UPDATE ApplicationReview
SET admin_id = @admin_id,
    response = @response
WHERE application_id = @application_id;
```

#### 4.12.5 Send a Connection Request

The screenshot shows the CareerBridge platform's user profile interface. On the left is a sidebar with options: Feed, Browse Ads, Profile, and Settings. The main area displays a user profile for "Berk Çakar". The profile picture is a blue placeholder. Below it, the name "Berk Çakar" is shown. To the right, there is a "Verified Master Skill: Python" badge with a rating of 4/5. A "Connect" button is visible. The profile is divided into sections: Information, Basic Information, Work Experience, DevOps Engineering Intern, Backend Development Intern, and a resume section.

Figure 31

This screenshot shows the same user profile for Berk Çakar as in Figure 31, but with a modal dialog box overlaid. The dialog box is titled "Connection Request" and asks, "Are you sure you want to send a connection request to Berk Çakar?" It has two buttons: "Cancel" and "Yes". The background of the profile page is dimmed.

Figure 32

**Inputs:** @user\_id, @receiver\_id

## SQL query to create a connection between on click Yes:

```
INSERT INTO Connection(sender_id, receiver_id)
VALUES (@user_id, @receiver_id);
```

### 4.12.6 Accept or Reject a Connection Request

The screenshot shows the 'CareerBridge' application interface. On the left, there is a sidebar with options: Feed, Browse Ads, Profile (selected), Apply For a Role, Settings, and Notifications. The main area displays a user profile for 'Berk Çakar' with a blue circular icon. Below the profile, there are sections for Information, Basic Information, Work Experience, Awards, Test Scores, Publications, Languages, Skills, and Resume. In the top right corner, there is a notifications bar stating 'You do not have any notifications!' and a 'Connection Requests' section. A notification card for 'Accept Kütay Tire's connection request' is visible, with a '5 min ago' timestamp and a 'Delete' button.

Figure 33

**Inputs:** @user\_id, @sender\_id

## SQL query to accept connection request for a user:

```
UPDATE Connection
SET response = 1
WHERE
    sender_id = @sender_id
    AND receiver_id = @user_id;
```

## SQL query to reject connection request for a user:

```
UPDATE Connection
SET response = 0
WHERE
    sender_id = @sender_id
    AND receiver_id = @user_id;
```

#### 4.12.7 User Feed

The screenshot shows the CareerBridge application's user feed. At the top right, there are notifications, messages, and a profile for 'Berk Çakar'. On the left, a sidebar has 'Feed' selected under 'Browse Ads', with options for 'Profile' and 'Settings'. The main area starts with a text input field 'Write something...'. Below it are buttons for 'Photo', 'Video', and 'Other Attachment', followed by a send arrow icon. A post from 'Derya Esen' (Career Expert, HR Manager @ Bilkent University) dated April 14, 2023, at 7:51 PM is shown. The post contains 10 tips for resume writing:

- 1 Set one-inch margins on all four sides.
- 2 Be consistent with your resume formatting.
- 3 Pick a 11 or 12pt resume font and stick to it.
- 4 Use single or 1.15 line spacing.
- 5 Create a proper resume header format for your contact details.
- 6 Add an extra space before and after each section heading.
- 7 Divide your resume into legible resume sections.
- 8 Make your resume as long as it needs to be.
- 9 Use bullet points to talk about past jobs.
- 10 Don't use photos on your resume.

Below the tips is a comment section with a like button, a reply button, and a text input field 'Write a comment...'. A comment from 'Cenk Yıldız' (Computer Engineer @ Amazon) dated April 14, 2023, at 8:44 PM is shown, saying 'Thanks for the recommendations!'. At the bottom, a message encourages users to 'Elevate your PowerPoint presentation skills to the next level! Check the video below.'

**Ads For You**

- Software Engineer at Google - Mountain View, California, USA [View Ad Details](#)
- Full Stack Developer at Amazon - Seattle, Washington, USA [View Ad Details](#)
- Computer Science Researcher at Bilkent University - Çankaya, Ankara, TR [View Ad Details](#)
- AI Intern at Microsoft - Redmond, Washington, USA [View Ad Details](#)
- Cybersecurity Analyst at IBM - Armonk, New York, USA [View Ad Details](#)
- Embedded Systems Engineer at Intel - Santa Clara, California, USA [View Ad Details](#)

**Applied Ads**

- Data Science Intern at Facebook - Menlo Park, California, USA Applied on 11 April 2023
- Machine Learning Intern at Tesla - Palo Alto, California, USA Applied on 5 April 2023
- Summer Research Intern at National University of Singapore - Singapore Applied on 24 March 2023

Figure 34

**Inputs:** @user\_id, @attachement, @content\_post, @post\_id, @content\_comment

**SQL query to retrieve the user feed:**

```
WITH Connected AS (
    SELECT sender_id
    FROM Connection C
    WHERE
        C.receiver_id = @user_id
        AND C.response IS TRUE
    UNION
    SELECT receiver_id
    FROM Connection C
    WHERE
        C.sender_id = @user_id
        AND C.response IS TRUE
)
SELECT *
FROM Post P,
```

```

Comment C
WHERE
    P.user_id = @user_id
    OR P.user_id IN Connected
    AND C.post_id = P.post_id
ORDER BY
    P.post_date DESC;

```

**SQL query to retrieve the applied ads:**

```

SELECT
    J.title,
    J.organization,
    J.location,
    JR.apply_date
FROM
    JobAdvertisementResponse JR,
    JobAdvertisement J
WHERE
    JR.ad_id = J.ad_id
    AND JR.user_id = @user_id
ORDER BY JR.apply_date DESC
LIMIT 3;

```

**SQL query to retrieve the ads for you:**

```

SELECT
    J.title,
    J.organization,
    J.location
FROM JobAdvertisement J
WHERE
    (SELECT S.skill_id
        FROM Skill S,
             Profile P
        WHERE
            S.profile_id = P.profile_id
            AND P.user_id = @user_id
    ) IN (
        SELECT S2.skill_id
        FROM
            Skill S2,
            SkillInJobAdvertisement SIJ
        WHERE
            S2.skill_id = SIJ.skill_id
            AND J.ad_id = SIJ.ad_id
    );

```

**SQL query to create a post:**

```
INSERT INTO Post (
    user_id, content, attachment, post_date)
VALUES
    (@user_id, @content_post, @attachment,
     NOW())
);
```

**SQL query to create a comment on a post:**

```
INSERT INTO Comment (
    post_id, user_id, content, commented_at)
VALUES
    (@post_id, @user_id, @content_comment,
     NOW())
);
```

#### 4.12.8 Admin Panel

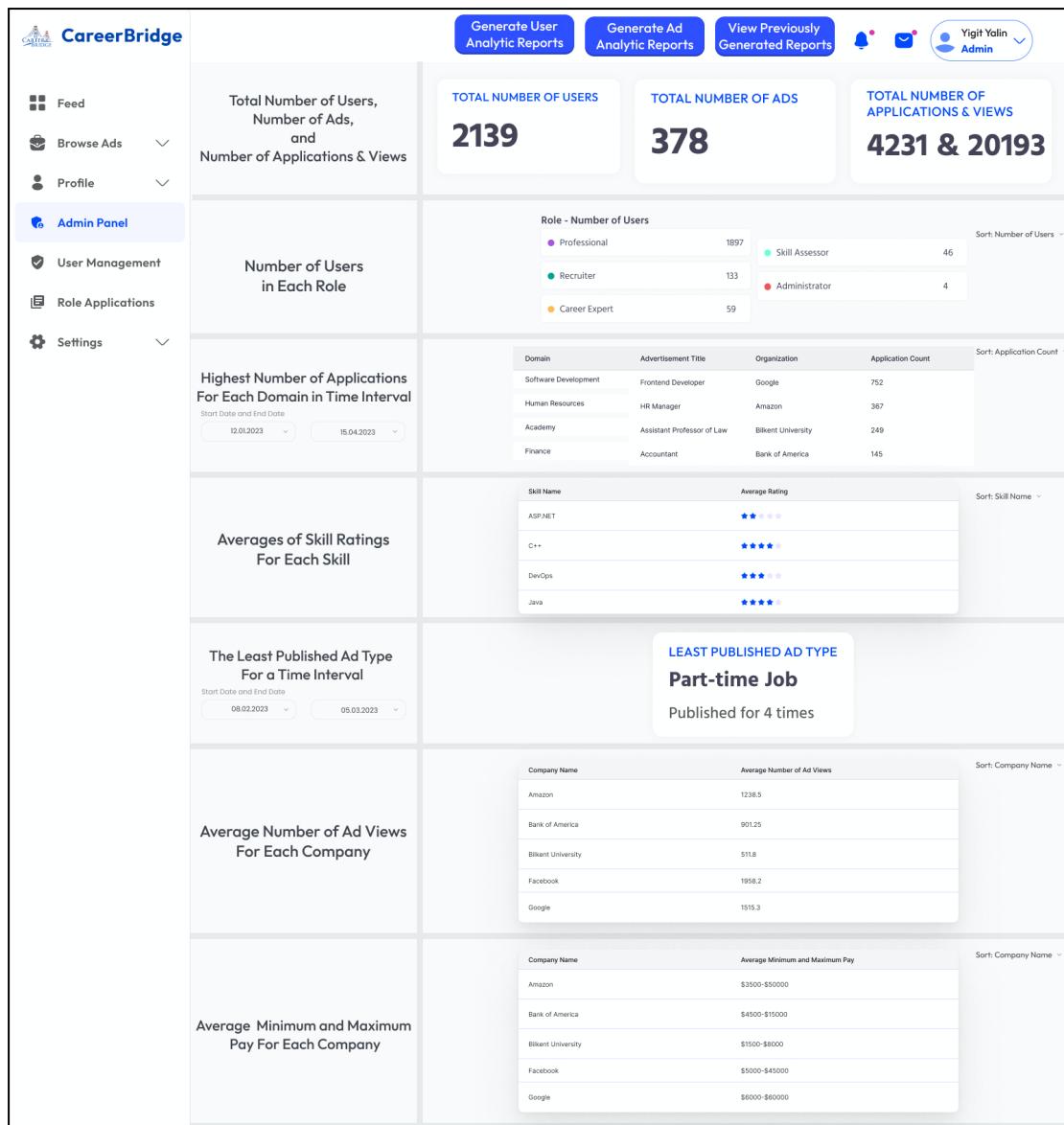


Figure 35

**SQL query to retrieve total number of users, total number of ads, total number of ad applications, and total number of ad views in CareerBridge:**

```

SELECT
    COUNT(*) AS user_count
FROM
    User;
SELECT
    COUNT(*) AS ad_count,
    SUM(view_count) as ad_view_count,
    SUM(application_count) as ad_application_count
FROM

```

```
JobAdvertisement;
```

**SQL query to retrieve the total number of users for each role in CareerBridge (sorting according to role name or number of users is possible):**

```
SELECT
    user_role,
    COUNT(*) AS user_count
FROM
(
    SELECT
        user_role
    FROM
        AppUser
    UNION ALL
    SELECT
        'Admin' AS user_role
    FROM
        Admin
) AS combined_table
GROUP BY
    user_role
ORDER BY
    user_count DESC;
```

**SQL query to retrieve the most applied ad for each domain in a given time interval (sorting according to domain, advertisement title, company name, or application count is possible):**

```
WITH domain_counts AS (
    SELECT
        JA.domain_name AS domain_name,
        JA.ad_id AS ad_id,
        COUNT(*) as application_count
    FROM
        JobAdvertisement JA,
        JobAdvertisementResponse JAR
    WHERE
        JA.ad_id = JAR.ad_id
        AND JAR.apply_date BETWEEN @start_date
        AND @end_date
    GROUP BY
        JA.domain_name,
        JA.ad_id
)
SELECT
    DC.domain_name,
    DC.ad_id,
    JA.title,
```

```

JA.organization,
DC.application_count
FROM
    domain_counts DC,
    JobAdvertisement JA
WHERE
    JA.ad_id = DC.ad_id
    AND DC.application_count = (
        SELECT
            MAX(application_count)
        FROM
            domain_counts
    )
ORDER BY
    DC.application_count DESC;

```

**SQL query to retrieve the average rating of available skills in CareerBridge (sorting according to skill name or average rating is possible):**

```

SELECT
    S.skill_name AS skill_name,
    AVG(SA.rating) AS average_rating
FROM
    Skill S,
    SkillAssessment SA
WHERE
    S.skill_id = SA.skill_id
GROUP BY
    S.skill_name
ORDER BY
    S.skill_name ASC;

```

**SQL query to retrieve the least applied type of ad in a given time interval:**

```

WITH ad_counts AS (
    SELECT
        JA.ad_type AS ad_type,
        COUNT(*) as application_count
    FROM
        JobAdvertisement JA,
        JobAdvertisementResponse JAR
    WHERE
        JA.ad_id = JAR.ad_id
        AND JAR.apply_date BETWEEN @start_date
        AND @end_date
    GROUP BY
        JA.ad_type
)

```

```

SELECT
    AC.ad_type,
    AC.application_count
FROM
    ad_counts AC
WHERE
    AC.application_count = (
        SELECT
            MIN(application_count)
        FROM
            ad_counts
    );

```

**SQL query to retrieve the average number of job advertisement view for each company (sorting according to average view count or company name is possible):**

```

SELECT
    C.company_name,
    AVG(JA.view_count) AS average_view_count
FROM
    Company C,
    JobAdvertisement JA
WHERE
    C.company_name = JA.organization
GROUP BY
    C.company_name
ORDER BY
    C.company_name ASC;

```

**SQL query to retrieve the average pay range for job advertisements for each company (sorting according to company name is possible):**

```

SELECT
    C.company_name,
    AVG(JA.pay_range_min) AS average_min_pay,
    AVG(JA.pay_range_max) AS average_max_pay
FROM
    Company C,
    JobAdvertisement JA
WHERE
    C.company_name = JA.organization
GROUP BY
    C.company_name
ORDER BY
    C.company_name ASC;

```

**In order to store the generated reports for user analytics and ad analytics the following queries will be executed upon the click on**

**“Generate Report” button. A description will be provided by the user which is @description, and assume the ID of the admin who is generating the report is @admin\_id:**

**SQL query to generate a system report of user analytics:**

```
INSERT INTO SystemReport (report_type, content, description,
creation_date, admin_id)
VALUES (
    'User Analytics',
    JSON_OBJECT(
        'total_users', (
            SELECT COUNT(*) AS user_count FROM User
        ),
        'users_by_role', (
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT('role', user_role, 'count', user_count)
            )
            FROM (
                SELECT user_role, COUNT(*) AS user_count
                FROM (SELECT user_role FROM AppUser UNION ALL SELECT
'Admin' AS user_role FROM Admin) AS combined_table
                GROUP BY user_role
            ) AS user_counts
        ),
        'average_skill_ratings', (
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT('skill_name', skill_name,
                'average_rating', average_rating)
            )
            FROM (
                SELECT S.skill_name AS skill_name, AVG(SA.rating) AS
average_rating
                FROM Skill S
                INNER JOIN SkillAssessment SA ON S.skill_id =
SA.skill_id
                GROUP BY S.skill_name
            ) AS skill_ratings
        )
    ),
    @description,
    NOW(),
    @admin_id
);
```

**SQL Query for generating a system report of ad analytics:**

```
INSERT INTO SystemReport (report_type, content, description,
creation_date, admin_id)
VALUES (
```

```

'Ad Analytics',
JSON_OBJECT(
    'most_applied_ads_by_domain', (
        WITH domain_counts AS (
            SELECT JA.domain_name AS domain_name, JA.ad_id AS ad_id, COUNT(*) as application_count FROM JobAdvertisement JA, JobAdvertisementResponse JAR WHERE JA.ad_id = JAR.ad_id AND JAR.apply_date BETWEEN @start_date AND @end_date GROUP BY JA.domain_name, JA.ad_id
        )
        SELECT DC.domain_name, DC.ad_id, JA.title, DC.application_count FROM domain_counts DC, JobAdvertisement JA WHERE JA.ad_id = DC.ad_id AND DC.application_count = (SELECT MAX(application_count) FROM domain_counts)
    ),
    'least_applied_ad_type', (
        WITH ad_counts AS (
            SELECT JA.ad_type AS ad_type, COUNT(*) as application_count FROM JobAdvertisement JA, JobAdvertisementResponse JAR WHERE JA.ad_id = JAR.ad_id AND JAR.apply_date BETWEEN @start_date AND @end_date GROUP BY JA.ad_type
        )
        SELECT AC.ad_type, AC.application_count FROM ad_counts AC WHERE AC.application_count = (SELECT MIN(application_count) FROM ad_counts)
    ),
    'average_job_ad_views_per_company', (
        SELECT C.company_name, AVG(JA.view_count) AS average_view_count FROM Company C, JobAdvertisement JA WHERE C.company_name = JA.organization GROUP BY C.company_name
    ),
    'average_pay_range_per_company', (
        SELECT C.company_name, AVG(JA.min_pay) AS average_min_pay, AVG(JA.max_pay) AS average_max_pay FROM Company C, JobAdvertisement JA WHERE C.company_name = JA.organization GROUP BY C.company_name
    )
),
@description,
NOW(),
@admin_id
);

```