

CONCORDIA UNIVERSITY
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

Files and Databases

(COMP#5531)

FINAL PROJECT

PREPARED BY:

Group# hxc55311

Anagh Mehran (40137948)
Donatus Ezeabasili (40137424)
Amol Kumar (40083475)
Mohamed Rashed (40038347)

SUBMITTED TO:

PROFESSOR KHALED JABABO

August 2020

Originality Form

“We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality”

Name	Signature	ID
Anagh Mehran	Anagh Mehran	40137948
Donatus Ezeabasili	Don Ezeabasili	40137424
Amol Kumar	Amol Kumar	40083475
Mohamed Rashed	Mohamed Rashed	40038347

Table of Contents

Overview.....	4
Design Phases	4
1. Initial Phase: User Requirements.....	5
2. Conceptual-Design Phase: The E-R Model.....	7
3. Logical-Design Phase:.....	9
I. The Relation Schemas.....	9
II. Normalization.....	11
Introduction.....	11
Functional Dependencies.....	12
Normal Form	13
4. SQL Declarations.....	16
i. DB Creation: Relations, Primary Keys and Foreign keys.....	16
ii. Data Insertion: DB Instances	24
iii. Queries and Transactions	26
5. Sample Artifacts	33
i. Job Portal: Main Page	33
ii. Job Portal: Signup Page	33
iii. Job Portal: Sign-in Page	34
iv. Job Portal: Create a job listing.....	34
v. Job Portal: Sample Job Listing	35

Overview

We consider the design of the given enterprise database with the use of Entity-Relationship data model (E-R). The E-R model provides a means of identifying entities to be represented in the database and how those entities are related. The final database design will be expressed in terms of a relational database design and an associated set of constraints.

Design Phases

1. Initial Phase:

Characterize fully the data needs of the prospective database users. The outcome of this phase is a specification of user requirements.

2. Conceptual-Design Phase:

Apply the concepts of the chosen data model (in our case, the E-R model) to translate these requirements into a conceptual schema of the database. The entity-relationship model specifies the entities that are represented in the database, the attributes of the entities, the relationships among the entities, and constraints on the entities and relationships.

3. Logical-Design Phase:

Map the conceptual schema defined using the entity-relationship model into a relation schema.

4. Physical-Design Phase:

Use the resulting system-specific database schema in the subsequent physical-design phase, in which the physical features of the database are specified.

1. Initial Phase: User Requirements

Your client is a service provider company, they would like you and your team to design and implement an online Web Career Portal for them to facilitate their day to day tasks and provide a better service for their clients. The online Job Portal website enable recruiters to post new job vacancies for any specialization for the hiring process. Also, it enables job seekers to search and apply for any number of jobs. You must create a dashboard that provide the ability to have access for all activities related to the recruiters and job seekers.

The requirements data to be modeled are as follows:

1. Employer side dashboard:

- 1) Should have an admin login and a mechanism to verify credentials or forgot password.
- 2) Should be able to maintain new users and update the user table.
- 3) Should be able to post jobs.
- 4) Should be able to maintain current jobs and categories.
- 5) Should be able to maintain status of applied jobs.
- 6) Should be able to see a summary of current application at a glance.
- 7) Two user categories should be provided:
 - **Prime:** Employer can post up to five jobs. A monthly charge of \$50 will be applied.
 - **Gold:** Employer can post as many jobs as he/she wants. A monthly charge of \$100 will be applied.
- 8) Users have the ability to upgrade or downgrade their category. Charges should be updated based on the new category.
- 9) When users upgrade their category, it should update the user category status.
- 10) Employer dashboard should have a contact us section to help user with contact information/helpline.

2. User side dashboard:

- 1) Should be able to sign up as a user.
- 2) Three user categories should be provided:
- 3) Basic: Employee can only view jobs but cannot apply. No charge.
 - **Prime:** Employee can view jobs as well as apply for up to five jobs. A monthly charge of \$10 will be applied.
 - **Gold:** Employee can view and apply to as many jobs as he/she wants. A monthly charge of \$20 will be applied.
- 4) Should be able to choose categories.
- 5) Users have the ability to upgrade or downgrade their category. Charges should be updated based on the new category.

- 6) When users upgrade their category, it should update the user category status.
- 7) Should have a user login and a mechanism to verify credentials or forgot password.
- 8) Should be able to search for jobs.
- 9) Should be able to search by job category.
- 10) Should be able to apply for jobs.
- 11) Should be able to maintain status of applied jobs.
- 12) Should be able to delete user profile.
- 13) Should be able to update user profile details.
- 14) Should be able to withdraw from an applied job.

3. **Administration side dashboard:**

- 1) Should have an admin login and a mechanism to verify credentials or forgot password.
- 2) Should be able to activate or deactivate a user.
- 3) Should be able to see all activities in the system.

All Users must have the ability to provide a method of payment to their account and provide the needed information to be charged. Method of payments could be either credit card charge or authorization from an existing checking account. In each case the proper information for withdrawal must be supplied by the user. The user has the option to choose between an automatic withdrawal or manual withdrawal. An automatic withdrawal is done automatically by the system at the beginning of each month by applying a charge to the user from a preselected method of payment. Users can have many methods of payments associated with their accounts and should be able to add/remove or edit their methods of payments. An account that is negative will be frozen until it is settled and a payment is made. The user of a frozen account will be able to login to the system but will not be able to access the services provided by the dashboard until the user account is settled. When a charge is applied to an account, an automatic email is sent to the user informing of the charge and account status. A suffering account will receive a warning message once a week until the account is settled or deactivated. A suffering account for a year will be deactivated automatically by the system.

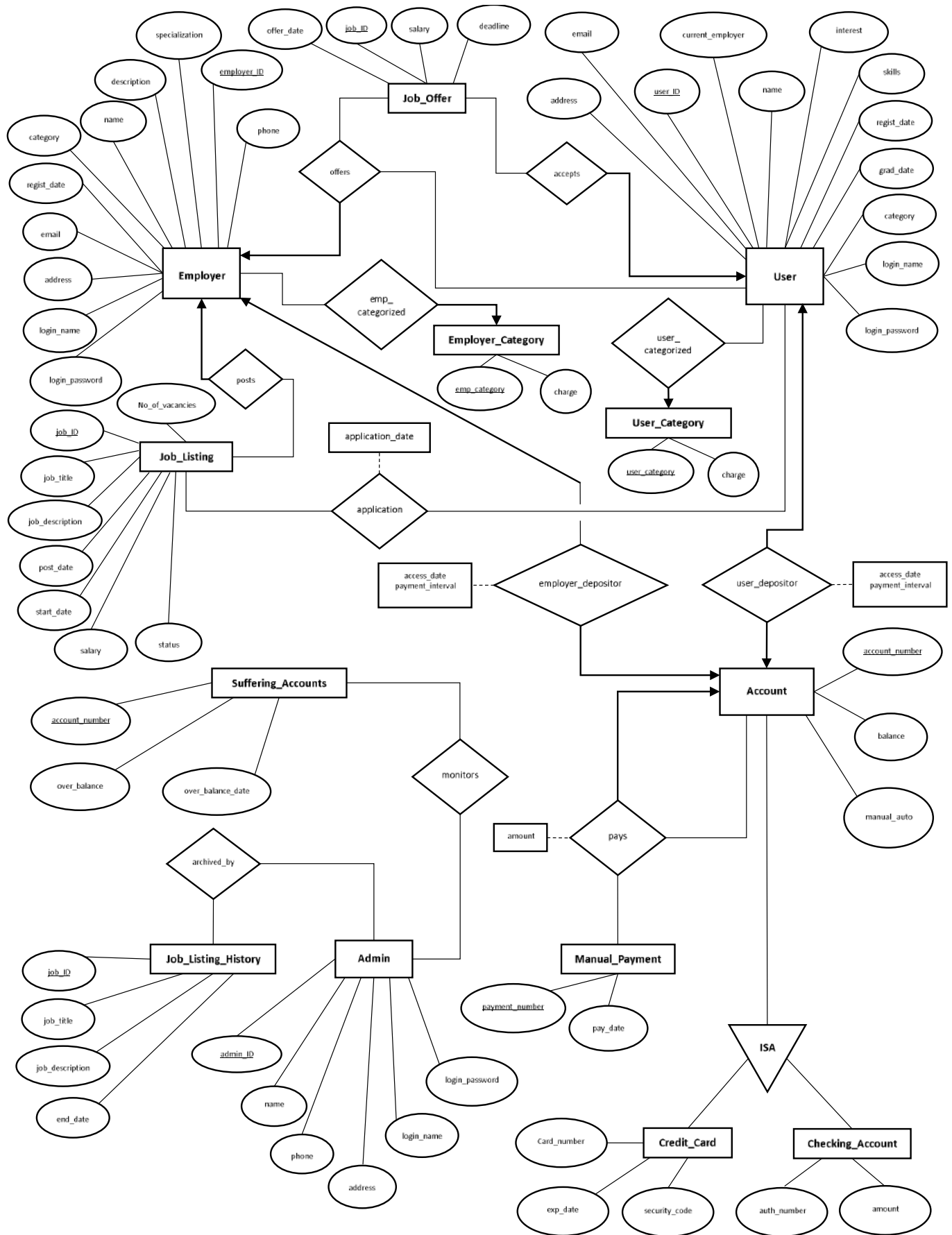
All users must have access to different reports that will enable them to obtain any needed information whether related to their account, posted jobs, applied jobs, accepted jobs, history, etc.

These are the minimum requirements for your application. More details could be added through more research and investigations from your part.

2. Conceptual-Design Phase: The E-R Model

The E-R data model employs three basic concepts: entity sets, relationship sets, and attributes. The E-R model also has an associated diagrammatic representation; the E-R diagram.

Figure
E-R Model – Recruitment Website



3. Logical-Design Phase:

I. The Relation Schemas

We can represent a database that conforms to an E-R database schema by a collection of relation schemas. For each entity set and for each relationship set in the database design, there is a unique relation schema to which we assign the name of the corresponding entity set or relationship set.

Representation of Relationship Sets:

We represent each relationship set by a relation schema with one attribute for each primary key from the forming entity sets.

The primary key for the resulting schema is chosen as follows:

- For a binary many-to-many relationship, the union of the primary-key attributes from the participating entity sets becomes the primary key.
- For a binary one-to-one relationship set, the primary key of either entity set can be chosen as the primary key. The choice can be made arbitrarily.
- For a binary many-to-one or one-to-many relationship set, the primary key of the entity set on the “many” side of the relationship set serves as the primary key.
- For an n-ary relationship set without any arrows on its edges, the union of the primary key-attributes from the participating entity sets becomes the primary key.
- For an n-ary relationship set with an arrow on one of its edges, the primary keys of the entity sets not on the “arrow” side of the relationship set serve as the primary key for the schema.

Foreign-Key Constraints:

We also create foreign-key constraints on the relation schema R as follows:

For each entity set E_i related to relationship set R , we create a foreign-key constraint from relation schema R , with the attributes of R that were derived from primary-key attributes of E_i referencing the primary key of the relation schema representing E_i .

Relation Schemas

Employer (employer_ID, name, description, specialization, address, phone, email, regist_date, login_name, login_password).

User (user_ID, first_name, last_name, address, phone, email, interest, skill, current_employer, grad_date, regist_date, login_name, login_password).

Admin (admin_ID, first_name, last_name, address, phone, email, login_name, login_password).

Employer_Category (emp_category, charge).

User_Category (user_category, charge).

Job_Listing (job_ID, job_title, job_description, no_of_vacancies, post_date, start_date, salary, status, category_id).

Job_Category (category_id, category_name)

Job_Listing_History (job_ID, job_title, job_description, filling_date).

Job_Offer (job_ID, salary, offer_date, deadline).

Account (account_number, balance, manual_or_auto).

Credit_Card_Account (account_number, card_number, exp_date, security_code).

Checking_Account (account_number, auth_number, amount).

Suffering_Accounts (account_number, over_balance, over_balance_date).

Manual_Payment (payment_number, payment_date)

Offers (job_ID, user_ID).

Accepts (job_ID, user_ID).

Emp_Categorized (employer_ID, emp_category).

User_Categorized (user_ID, user_category).

Posts (job_ID, employer_ID).

Application (job_ID, user_ID, application_date).

User_Depositor (account_number, user_ID, access_date, payment_interval).

Employer_Depositor (account_number, employer_ID, access_date, payment_interval).

pays (payment_number, account_number, amount).

II. Normalization

Introduction

We can represent a database that conforms to an E-R database schema by a collection of relation schemas. For each entity set and for each relationship set in the database design, there is a unique relation schema to which we assign the name of the corresponding entity set or relationship set.

In general, the goal of relational database design is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily. This is accomplished by designing schemas that are in an appropriate normal form. To determine whether a relation schema is in one of the desirable normal forms, we need information about the real-world enterprise that we are modeling with the database. Some of this information exists in a well-designed E-R diagram, but additional information about the enterprise may be needed as well.

In this section, we introduce a formal approach to relational database design based on the notion of functional dependencies. We then define the normal form of our enterprise in terms of functional dependencies and other types of data dependencies.

Functional Dependencies

Keys, and more generally, functional dependencies are constraints on the database that require relations to satisfy certain properties. Relations that satisfy all such constraints are legal relations.

We shall use functional dependencies in two ways:

1. To test relations to see whether they are legal under a given set of functional dependencies.
2. To specify constraints on the set of legal relations.

Normal Form

```
Let  $F_c$  be a canonical cover for  $F$ ;  
i=0;  
For each functional dependency  $\alpha \rightarrow \beta$  in  $F_c$  do  
    if none of the schemas  $R_j$ ,  $j = 1, 2, \dots, i$  contains  $\alpha \beta$   
        Then begin  
            i=i+1;  
             $R_i = \alpha \beta$ ;  
        end  
if none of the schemas  $R_j$ ,  $j = 1, 2, \dots, i$  contains a candidate key for  $R$   
    Then begin  
        i=i+1;  
         $R_i =$  any candidate key for  $R$ ;  
    End  
return
```

Dependency-preserving, lossless decomposition into 3NF

Employer (employer_ID, employer_name, emp_description, emp_specialization, country, phone, email, regist_date, login_name, login_password).

employer_ID → all other attributes

User (user_ID, first_name, last_name, country, phone, email, interest, skill, current_employer, grad_date, regist_date, login_name, login_password).

user_ID → all other attributes

admin (admin_ID, first_name, last_name, country, phone, email, login_name, login_password).

admin_ID → all other attributes

employer_category (emp_category, charge).

emp_category → charge

User_Category (user_category, charge).

user_category → charge

job_listing (job_ID, job_title, job_description, no_of_vacancies, post_date, start_date, salary, status, category_id).

job_ID → all other attributes

Job_Category (category_id, category_name)

Category_id → category_name

Job_Listing_History (job_ID, job_title, job_description, filling_date).

job_ID → job_title, job_description, filling_date

Job_Offer (job_ID, salary, offer_date, deadline).

job_ID → salary, offer_date, deadline

account (account_number, balance, manual_or_auto).

account_number → balance, manual_or_auto

Credit_Card_Account (account_number, card_number, exp_date, security_code).

account_number → card_number, exp_date, security_code

Checking_Account (account_number, auth_number, amount).

account_number → auth_number, amount

suffering_accounts (account_number, over_balance, over_balance_date).

account_number → over_balance, over_balance_date

manual Payment (payment_number, payment_date)

$\text{payment_number} \rightarrow \text{payment_date}$

offers (job_ID, user_ID).

accepts (job_ID, user_ID).

emp_Categorized (employer_ID, emp_category).

$\text{employer_ID} \rightarrow \text{emp_category}$

User_Categorized (user_ID, user_category).

$\text{user_ID} \rightarrow \text{user_category}$

Posts (job_ID, employer_ID).

$\text{job_ID} \rightarrow \text{employer_ID}$

Application (job_ID, user_ID, application_date).

$\text{job_ID}, \text{user_ID} \rightarrow \text{application_date}$

User_Depositor (account_number, user_ID, access_date, payment_interval).

$\text{account_number} \rightarrow \text{user_ID}, \text{access_date}, \text{payment_interval}$

Employer_Depositor (account_number, employer_ID, access_date, payment_interval).

$\text{account_number} \rightarrow \text{employer_ID}, \text{access_date}, \text{payment_interval}$

pays (payment_number, account_number, amount).

$\text{Payment_number}, \text{account_number} \rightarrow \text{amount}$

Because the LHS of every FD is the schema's key, the schemas all satisfy BCNF.

4. SQL Declarations

i. DB Creation: Relations, Primary Keys and Foreign keys

```
CREATE TABLE employer (  
    employer_ID int NOT NULL AUTO_INCREMENT,  
    employer_name varchar(50) NOT NULL,  
    emp_description varchar(100),  
    emp_specialization varchar(100),  
    country varchar(50),  
    phone varchar(15),  
    email varchar(50),  
    regist_date DATE,  
    login_name varchar(50),  
    login_password varchar(50),  
    PRIMARY KEY (employer_ID)  
);
```

```
CREATE TABLE user (  
    user_ID int NOT NULL AUTO_INCREMENT,  
    first_name varchar(50) NOT NULL,  
    last_name varchar(50) NOT NULL,  
    country varchar(50),  
    phone varchar(15),  
    email varchar(50),  
    interest varchar(50),  
    skill varchar(50),  
    current_employer varchar(50),  
    graduation_date DATE,  
    regist_date DATE,  
    login_name varchar(50),
```



```
login_password varchar(50),  
PRIMARY KEY (user_ID)  
);
```

```
CREATE TABLE admin (  
    admin_ID int NOT NULL AUTO_INCREMENT,  
    first_name varchar(50) NOT NULL,  
    last_name varchar(50) NOT NULL,  
    country varchar(50),  
    phone varchar(15),  
    email varchar(50),  
    login_name varchar(50),  
    login_password varchar(50),  
    PRIMARY KEY (admin_ID)  
);
```

```
CREATE TABLE employer_category (  
    emp_category varchar(50) NOT NULL,  
    charge INT,  
    PRIMARY KEY (emp_category)  
);
```

```
CREATE TABLE user_category (  
    user_category varchar(50) NOT NULL,  
    charge INT,  
    PRIMARY KEY (user_category)  
);
```

```
CREATE TABLE job_category (  
    category_id INT NOT NULL,
```

```
category_name varchar(50),  
PRIMARY KEY (category_id)  
);
```

```
CREATE TABLE job_listing (  
    job_ID int NOT NULL AUTO_INCREMENT,  
    job_title varchar(50) NOT NULL,  
    job_description varchar(50) NOT NULL,  
    no_of_vacancies varchar(50),  
    post_date DATE,  
    start_date DATE,  
    salary INT,  
    status varchar(50),  
    category_id INT,  
    PRIMARY KEY (job_ID),  
    FOREIGN KEY (category_id) REFERENCES job_category(category_id)  
);
```

```
CREATE TABLE job_listing_history (  
    job_ID int NOT NULL,  
    job_title varchar(50) NOT NULL,  
    job_description varchar(50) NOT NULL,  
    filling_date DATE,  
    PRIMARY KEY (job_ID),  
    FOREIGN KEY (job_ID) REFERENCES job_listing(job_ID)  
);
```

```
CREATE TABLE application (  
    job_ID int NOT NULL,  
    user_ID int NOT NULL,
```

```
application_date DATE,  
PRIMARY KEY (job_ID, user_ID),  
FOREIGN KEY (job_ID) REFERENCES job_listing(job_ID),  
FOREIGN KEY (user_ID) REFERENCES user(user_ID)  
);
```

```
CREATE TABLE job_offer (  
    job_ID int NOT NULL,  
    salary int,  
    offer_date DATE,  
    deadline DATE,  
    PRIMARY KEY (job_ID),  
    FOREIGN KEY (job_ID) REFERENCES application(job_ID)  
);
```

```
CREATE TABLE account (  
    account_number int NOT NULL AUTO_INCREMENT,  
    balance int,  
    manual_or_auto varchar(50),  
    PRIMARY KEY (account_number)  
);
```

```
CREATE TABLE credit_card_account (  
    account_number int NOT NULL,  
    card_number varchar(15),  
    exp_date DATE,  
    security_code int,  
    PRIMARY KEY (account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number)  
);
```

```
CREATE TABLE checking_account (  
    account_number int NOT NULL,  
    auth_number varchar(15),  
    amount int,  
    PRIMARY KEY (account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number)  
);
```

```
CREATE TABLE suffering_account (  
    account_number int NOT NULL,  
    over_balance int,  
    over_balance_date DATE,  
    PRIMARY KEY (account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number)  
);
```

```
CREATE TABLE manual_payment (  
    paymant_number int NOT NULL AUTO_INCREMENT,  
    paymant_date DATE,  
    PRIMARY KEY (paymant_number)  
);
```

```
CREATE TABLE offers (  
    job_ID int NOT NULL,  
    user_ID int NOT NULL,  
    PRIMARY KEY (job_ID, user_ID),  
    FOREIGN KEY (job_ID) REFERENCES job_offer(job_ID),  
    FOREIGN KEY (user_ID) REFERENCES application(user_ID)
```

);

```
CREATE TABLE accepts (  
    job_ID int NOT NULL,  
    user_ID int NOT NULL,  
    PRIMARY KEY (job_ID, user_ID),  
    FOREIGN KEY (job_ID) REFERENCES offers(job_ID),  
    FOREIGN KEY (user_ID) REFERENCES offers(user_ID)  
);
```

```
CREATE TABLE rejects (  
    job_ID int NOT NULL,  
    user_ID int NOT NULL,  
    PRIMARY KEY (job_ID, user_ID),  
    FOREIGN KEY (job_ID) REFERENCES offers(job_ID),  
    FOREIGN KEY (user_ID) REFERENCES offers(user_ID)  
);
```

```
CREATE TABLE emp_categorized (  
    employer_ID int NOT NULL,  
    emp_category varchar(50) NOT NULL,  
    PRIMARY KEY (employer_ID),  
    FOREIGN KEY (employer_ID) REFERENCES employer(employer_ID)  
);
```

```
CREATE TABLE user_categorized (  
    user_ID int NOT NULL,  
    user_category varchar(50) NOT NULL,  
    PRIMARY KEY (user_ID),  
    FOREIGN KEY (user_ID) REFERENCES user(user_ID)
```

);

```
CREATE TABLE posts (  
    job_ID int NOT NULL,  
    employer_ID int NOT NULL,  
    PRIMARY KEY (job_ID),  
    FOREIGN KEY (job_ID) REFERENCES job_listing(job_ID),  
    FOREIGN KEY (employer_ID) REFERENCES employer(employer_ID)  
);
```

```
CREATE TABLE user_depositor (  
    account_number int NOT NULL,  
    user_ID int NOT NULL,  
    access_date DATE,  
    payment_interval int,  
    PRIMARY KEY (account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number),  
    FOREIGN KEY (user_ID) REFERENCES user(user_ID)  
);
```

```
CREATE TABLE employer_depositor (  
    account_number int NOT NULL,  
    employer_ID int NOT NULL,  
    access_date DATE,  
    payment_interval int,  
    PRIMARY KEY (account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number),  
    FOREIGN KEY (employer_ID) REFERENCES employer(employer_ID)  
);
```

```
CREATE TABLE pays (  
    paymant_number int NOT NULL,  
    account_number int NOT NULL,  
    amount int,  
    PRIMARY KEY (paymant_number, account_number),  
    FOREIGN KEY (paymant_number) REFERENCES manual_payment(paymant_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number)  
);
```

ii. Data Insertion: DB Instances

INSERT INTO employer (employer_name, emp_description, emp_specialization, country, phone, email, regist_date, login_name, login_password)

VALUES ('ABC plc', 'Leader in IT hardwares', '5G Network installation', 'USA', '0017215567891', 'abc@techno.com', '2001-11-15', 'abc_plc', 'abc5GNetwork');

INSERT INTO user (first_name, last_name, country, phone, email, interest, skill, current_employer, graduation_date, regist_date, login_name, login_password)

VALUES ('James', 'Alfred', 'France', '0017435567888', 'JamesAlfred@france.com', 'abc', 'welder', 'Chase Welding', '2010-05-15', '2006-01-10', 'JAlfred', 'JwelAlfred');

INSERT INTO admin (first_name, last_name, country, phone, email, login_name, login_password)

VALUES ('Franklin', 'Andrew', 'Canada', '0017437567324', 'FrankAndd@yahoo.com', 'FrankAndrew', 'FAndrewCan');

INSERT INTO employer_category ()

VALUES ('Prime', 50);

INSERT INTO user_category ()

VALUES ('Prime', 10);

INSERT INTO job_category ()

VALUES (1, 'full time');

INSERT INTO job_listing (job_title, job_description, no_of_vacancies, post_date, start_date, salary, status, category_id)

VALUES ('Mechanic', 'Toyota cars mechanic', 3, '2020-08-01', '2020-09-01', 60000, 'open', 1);


```
INSERT INTO user_categorized ()  
VALUES (15, 'Gold');
```

```
INSERT INTO application ()  
VALUES (1, 11, '2020-08-05');
```

```
INSERT INTO job_offer ()  
VALUES (3, 76000, '2020-09-21', '2020-09-30');
```

```
INSERT INTO posts ()  
VALUES (4, 5);
```

iii. Queries and Transactions

Get user email and login password. Search by 'email'.

```
select email, login_password  
from user  
where email = 'JamesAlfred@france.com';
```

Get all attributes for one user. Search by user_id

```
select *  
from user  
where user_ID = 1;
```

Get name and id of all categories (job category)

```
select *  
from job_category;
```

Get name of a job_category (search by id)

```
select category_name  
from job_category  
where category_id = 2;
```

Get list of all jobs (sort by date posted, descending order)

```
SELECT *  
FROM job_listing  
ORDER BY post_date DESC;
```

Get list of jobs (search by category_id)

```
select *  
from job_listing  
where category_id = 1;
```

Get a single job (search by job_id)

```
select *  
from job_listing  
where job_id = 2;
```

Post a new application. User applies to job. New row in ‘Application’ relation. (insert: job_id, user_id, date)

```
INSERT INTO application()  
VALUES (6, 12, '2020-08-04');
```

Edit name, email etc field for a specific user

```
UPDATE user  
SET first_name = 'Schmidt'  
WHERE user_ID = 12;
```

```
UPDATE user  
SET last_name = 'Joseph'  
WHERE user_ID = 12;
```

```
UPDATE user  
SET country = 'France'  
WHERE user_ID = 12;
```

```
UPDATE user
SET phone = 0015678769856
WHERE user_ID = 12;
```

```
UPDATE user
SET email = 'Joseph@gmail.com'
WHERE user_ID = 12;
```

```
UPDATE user
SET interest = 'traveling'
WHERE user_ID = 12;
```

```
UPDATE user
SET skill = 'Diving'
WHERE user_ID = 12;
```

```
UPDATE user
SET current_employer = 'Microsoft'
WHERE user_ID = 12;
```

```
UPDATE user
SET graduation_date = '2003-12-15'
WHERE user_ID = 12;
```

```
UPDATE user
SET regist_date = '2008-10-15'
WHERE user_ID = 12;
```

```
UPDATE user
SET login_name = 'JoeFredMotors'
WHERE user_ID = 12;
```

```
UPDATE user
SET login_password = 'Joexxzrr'
WHERE user_ID = 12;
```

Change user category. Edit the User_categorized relation.

```
UPDATE user_categorized
SET user_category = 'Gold'
WHERE user_ID = 13;
```

Find list of job a specific user has already applied for

```
select job_listing.job_ID, job_description, post_date, start_date, salary, application_date
from job_listing, application
where job_listing.job_ID = application.job_ID AND user_ID = 13;
```

Find job offer offered to a user. Find all the fields for the job like title, company etc.

```
select employer_name, job_title, job_description, job_offer.salary, offer_date
from offers, job_offer, job_listing, posts, employer
where user_ID = 14 AND offers.job_ID = job_offer.job_ID AND offers.job_ID =
job_listing.job_ID AND offers.job_ID = posts.job_ID AND posts.employer_ID =
employer.employer_ID;
```

Delete job application by a specific user

```
DELETE FROM application WHERE user_ID = 15;
```

Delete job offer by a specific user

```
DELETE FROM job_offer WHERE user_ID = 11;
```

Delete user_account by a specific user

```
DELETE FROM user WHERE user_ID = 12;
```

Delete a job_listing

```
DELETE FROM job_listing WHERE job_ID = 1;
```

Create a job offer. (offered to a single user)

```
INSERT INTO job_offer ()
```

```
VALUES (4, 43000, '2020-10-01', '2020-09-15');
```

```
INSERT INTO offers ()
```

```
VALUES (4, 14);
```

Get a list of all applicants for a single job listing

```
select first_name, last_name, job_title
```

```
from user, application, job_listing
```

```
where job_listing.job_ID = 4 AND user.user_ID = application.user_ID AND  
application.job_ID = job_listing.job_ID;
```

Create a new user

```
INSERT INTO user (first_name, last_name, country, phone, email, interest, skill,  
current_employer, graduation_date, regist_date, login_name, login_password)
```

```
VALUES ('Foreman', 'Tyson', 'Canada', '0018434067734', 'Foreman201@gmail.com', 'trial2',  
'trainer', 'Boxing Federation', '1995-05-12', '1990-01-12', 'ForTyson', 'lIvEbOxInG');
```

Create a new employer

```
INSERT INTO employer (employer_name, emp_description, emp_specialization, country, phone, email, regist_date, login_name, login_password)
```

```
VALUES ('Houston Traders', 'Games dealer', 'sells latest computer games', 'Canada', '0011957463297', 'Games_dealer@gmail.com', '1980-01-20', 'Games_dealer', 'V2Pgthhy');
```

Create a new admin

```
INSERT INTO admin (first_name, last_name, country, phone, email, login_name, login_password)
```

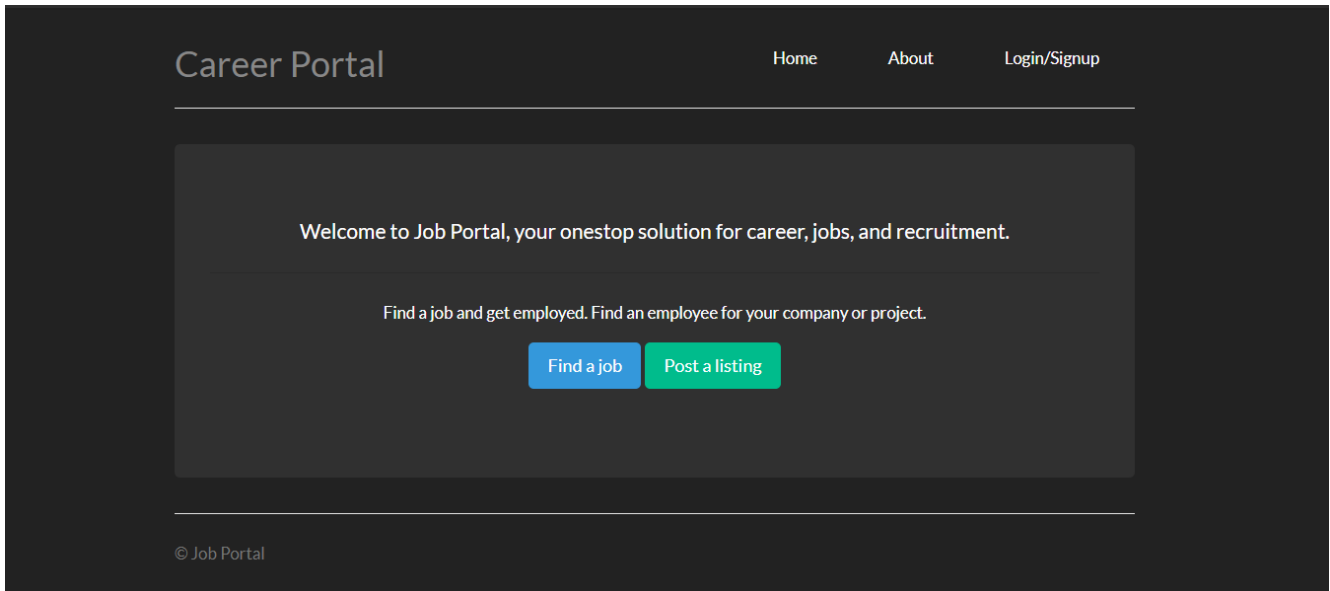
```
VALUES ('Kinsley', 'Raymond', 'Canada', '0018430087363', 'Raymond@ygmail.com', 'Raybam', 'tryhyehdjud');
```

5. Contributions

	Anagh Mehran	Amol Kumar	Don Ezeabasili	Mohamed Rashed
Requirements Elicitation	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
E-R Modeling	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Relation Schemas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Normalization	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
SQL Declarations	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Queries	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Backend (PHP)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Frontend (UI)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Database Testing	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
UI testing	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Report	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

6. Sample Artifacts

i. Job Portal: Landing page



The screenshot shows the landing page of a 'Career Portal'. The header features the title 'Career Portal' on the left and navigation links 'Home', 'About', and 'Login/Signup' on the right. The main content area is a dark gray rectangle with white text. It starts with a welcome message: 'Welcome to Job Portal, your onestop solution for career, jobs, and recruitment.' Below this is a sub-header: 'Find a job and get employed. Find an employee for your company or project.' Two buttons are centered: a blue 'Find a job' button and a green 'Post a listing' button. The footer contains the copyright notice '© Job Portal'.

Career Portal

Home About Login/Signup

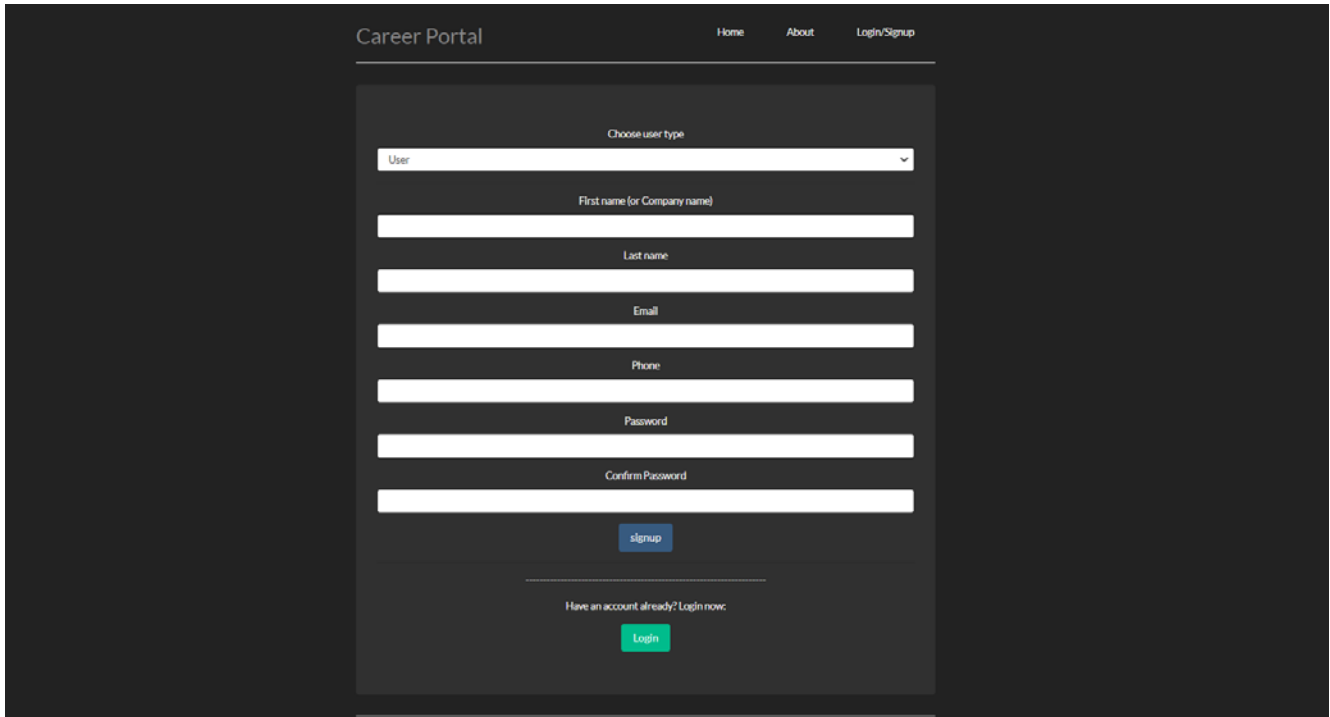
Welcome to Job Portal, your onestop solution for career, jobs, and recruitment.

Find a job and get employed. Find an employee for your company or project.

Find a job Post a listing

© Job Portal

ii. Job Portal: Signup Page



The screenshot shows the signup page of the 'Career Portal'. The header is identical to the landing page. The main content area is a dark gray rectangle with white text and input fields. It starts with a dropdown menu labeled 'Choose user type' with 'User' selected. Below this are six text input fields labeled 'First name (or Company name)', 'Last name', 'Email', 'Phone', 'Password', and 'Confirm Password'. A blue 'signup' button is centered below the fields. At the bottom, there is a link 'Have an account already? Login now.' with a green 'Login' button below it.

Career Portal

Home About Login/Signup

Choose user type

User

First name (or Company name)

Last name

Email

Phone

Password

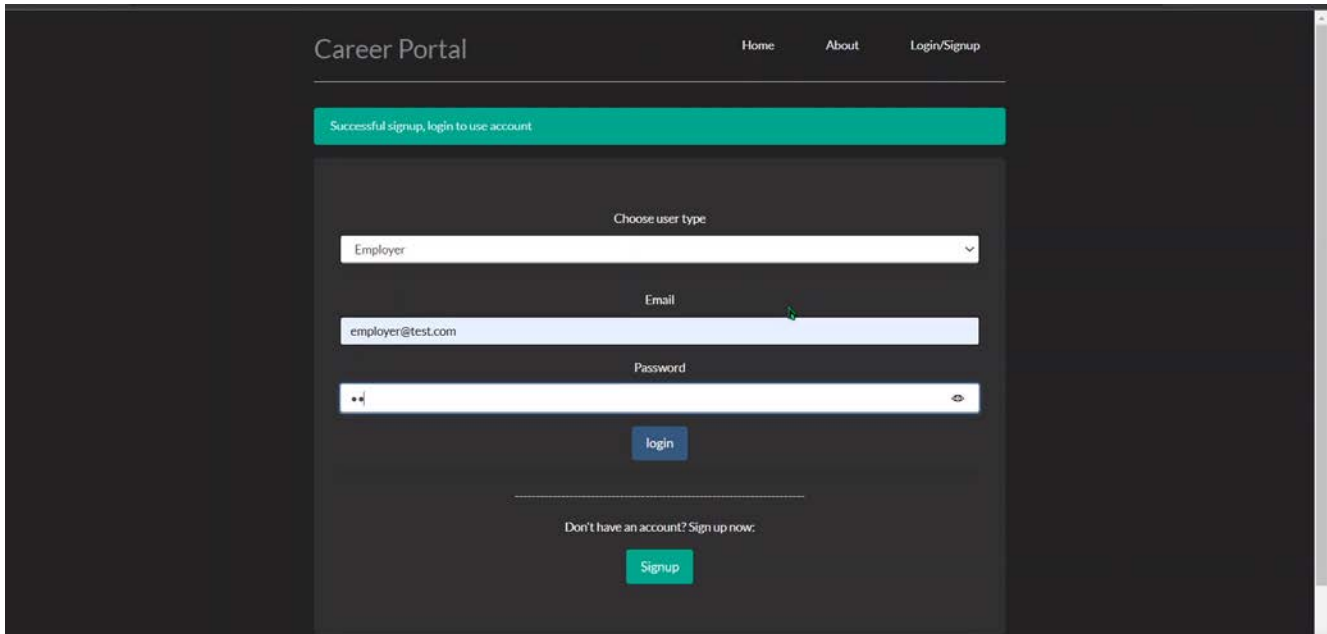
Confirm Password

signup

Have an account already? Login now.

Login

iii. Job Portal: Sign-in Page



The screenshot shows the 'Sign-in Page' of a 'Career Portal'. The page has a dark blue header with the portal name and navigation links. A green success message is displayed at the top. The main form area is white and contains fields for 'Choose user type' (a dropdown menu with 'Employer' selected), 'Email' (containing 'employer@test.com'), and 'Password' (with masked characters and a toggle icon). A blue 'login' button is positioned below the password field. At the bottom of the form, there is a link to 'Sign up now' and a green 'Signup' button.

Career Portal

Home About Login/Signup

Successful signup, login to use account

Choose user type

Employer

Email

employer@test.com

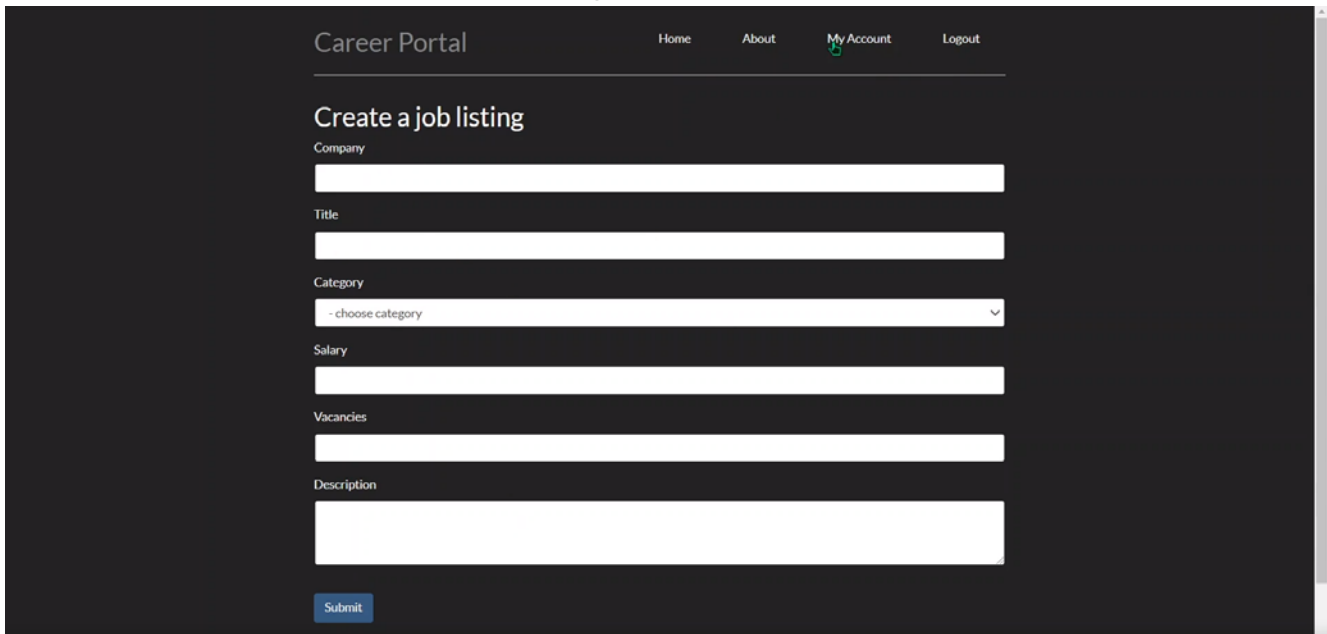
Password

login

Don't have an account? Sign up now:

Signup

iv. Job Portal: Create a job listing



The screenshot shows the 'Create a job listing' page of a 'Career Portal'. The page has a dark blue header with the portal name and navigation links. The main form area is white and contains several input fields: 'Company', 'Title', 'Category' (a dropdown menu with '- choose category' selected), 'Salary', 'Vacancies', and 'Description'. A blue 'Submit' button is located at the bottom left of the form.

Career Portal

Home About My Account Logout

Create a job listing

Company

Title

Category

- choose category

Salary

Vacancies

Description

Submit

v. Job Portal: Sample Job Listing

