USE CASE STUDY REPORT : Milestone 2

Group No.: Group 15

Student Names: Vraj Diyora and Amruta Hombali

Problem Definition:

Design and develop a cross-platform job search tool that aggregates job listings from different websites and provides a one-stop-shop for job seekers to search and apply for jobs. The Job Search Tool aims to provide a seamless job search experience for potential candidates. The tool should enable job seekers to filter job listings based on company name, job ID, job title, location, and salary. The job listings should be refreshed every other day, and the tool should prevent duplicate listings from appearing.

The Job Search Tool should gather job listings from different job listing websites, each with its own unique Website ID and Website Name. The tool should refresh the job listings every other day to ensure that job seekers have access to the latest job listings.

The tool should allow job seekers to create a user profile with a user ID, name, experience, and location, and apply for job listings with ease. Each job listing should display the number of applicants who have applied to the job. The User will also currently have a Role which has a Role ID.

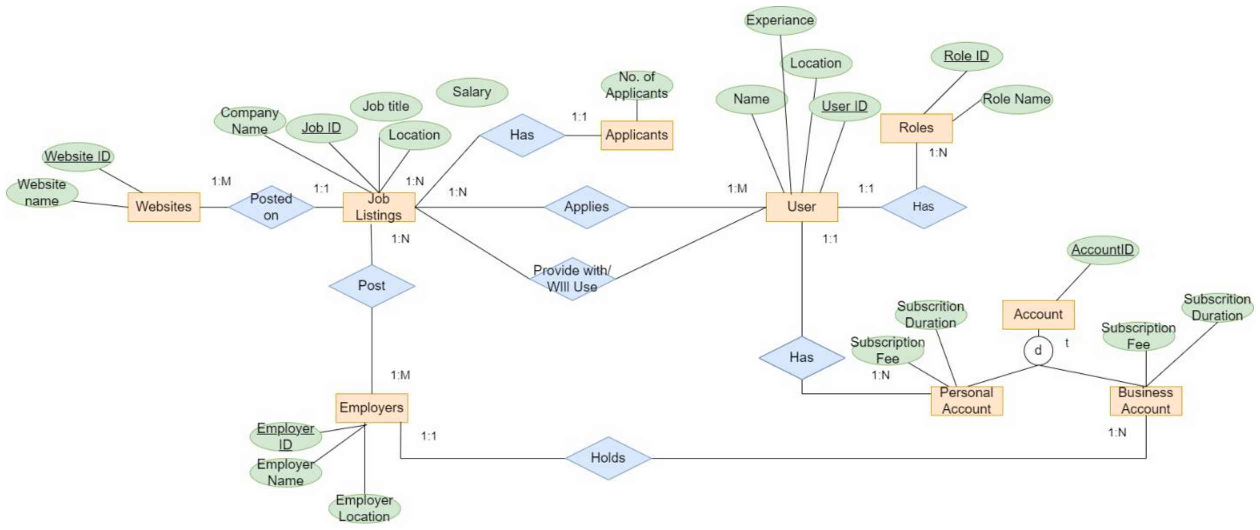
Employers should be able to post job listings, with each job listing having an employer ID, employer name, and location. Both the employer and User will have an account, but they can either have a business account or a personal account.

The job search tool requires a subscription-based model that allows users to create either a business or personal account. Each account type has a different subscription fee and duration. The subscription fee varies based on the account type, and users are required to pay the fee to access the services provided by the tool.

The tool should ensure that the employer and user information is kept confidential, and only the relevant parties can access it. Additionally, the tool should provide employer information to job seekers and vice versa.

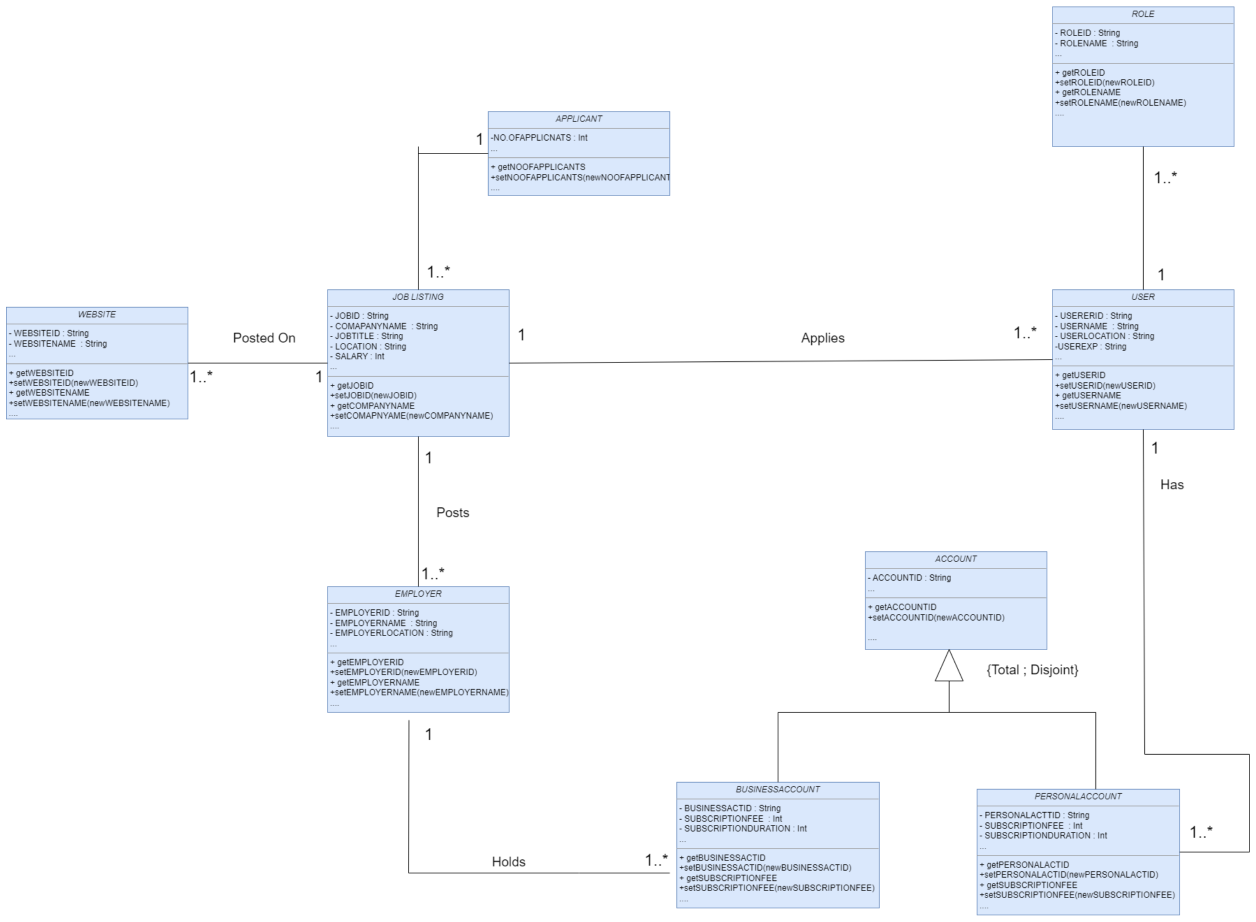
Reference data include Website ID, Website Name, User ID, Role ID, and Employer ID. Transactional data include Job listings, Applicants, and Account details.

ER DIAGRAM:


Experiance
Location
Role ID
No. of Applicants
Name
User ID
Role Name
Salary
Job title
Company Name
Job ID
Location
Has
1:1
Applicants
Roles
1:N
Website ID
Website name
Websites
1:M
Posted on
1:1
1:N
Job Listings
1:N
Applies
1:M
User
1:1
Has
1:N
1:1
AccountID
Post
Provide with/ WIll Use
Subscrition Duration
Account
Subscrition Duration
Subscription Fee
Subscription Fee
d
t
Has
1:N
Personal Account
Business Account
1:M
Employers
1:1
Holds
1:N
Employer ID
Employer Name
Employer Location

UML class Diagram :

In this section, we will present a UML class diagram for the Job Search Tool across platform project. The diagram will show the different entities involved in the project and their relationships, along with the attributes associated with each entity



Mapping the Relational Model based on the conceptual data model:

After developing a conceptual data model using UML class diagrams, the next step is to map it to a relational model. A relational model consists of tables with columns representing attributes and relationships between tables. In the following section, we will describe each of these tables and their attributes in detail, as well as the relationships between them.

Relational Model :

Job Listing (JobID, Company Name, Job Title, Location, Salary)

Website (WebsiteID**,** *JobID***,** WebsiteName**)**
Foreign Key – *JobID* refers to JobID in Job listing. NULL not allowed.

Applicants (*JobID***,** Number of Applicants**)**
Foreign Key – *JobID* refers to JobID in Job listing. NULL not allowed.

Employer (EmployerID**,** Employer Name, Employer Location**)**

Job Posts (*JobID, EmployerID***)**
Foreign Key – *JobID* refers to JobID in Job listing. NULL not allowed.
Foreign Key – *EmployerID* refers to EmployerID in Employer. NULL not allowed.

User (UserID**,** UserName, UserLocation, UserExperiance**)**

AppliesforJob (*JobID, UserID***)**
Foreign Key – *JobID* refers to JobID in Job listing. NULL not allowed.
Foreign Key – *UserID* refers to UserID in User. NULL not allowed.

Role (*RoleID***,** *JobID*, Role Name**)**
Foreign Key – *JobID* refers to JobID in Job listing. NULL not allowed.

Account (AccountID**)**

Personal Account (PersonalActID, *UserID*, SubscriptionFee, SubscriptionDuration**)**
Foreign Key – *UserID* refers to UserID in User. NULL not allowed.

Business Account (BusinessActID, *EmployerID*, SubscriptionFee, SubscriptionDuration**)**
Foreign Key – *EmployerID* refers to EmployerID in Employer. NULL not allowed.

- The SQL statements used to create the tables in the database are :

```
create table Job_listing(
JobID char(4) NOT NULL PRIMARY KEY,
Company_Name varchar(60) NOT NULL,
Job_Title varchar(40),
Location varchar(20),
Salary Integer);
create table Website(
WebsiteID char(4) NOT NULL PRIMARY KEY,
JobID char(4) NOT NULL,
Foreign Key (JobID) References Job_listing(JobID),
WebsiteName Varchar(40));

DROP TABLE IF EXISTS applicants;
create table Applicants(
JobID char(4) NOT NULL,
Foreign Key (JobID) References Job_listing(JobID),
NumberOfApplicants Integer);

create table Employer(
EmployerID char(6) not null Primary key,
EmployerName varchar(40),
EmployerLocation varchar(40));

create table JobPosts(
JobID char(4) NOT NULL,
Foreign Key (JobID) References Job_listing(JobID),
EmployerID char(6) NOT NULL,
foreign key (EmployerID) References Employer(EmployerID));

create table User(
UserID char(6) NOT NULL Primary key,
UserName varchar(40) NOT NULL,
UserLocation varchar(40),
UserExperiance Varchar(40));

create table AppliesForJob(
primary key (JobID, UserID),
JobID char(4) NOT NULL,
Foreign Key (JobID) References Job_listing(JobID),
UserID char(6) NOT NULL,
foreign key (UserID) References User(UserID));
create table Role(
RoleID char(6) NOT NULL PRIMARY KEY,
UserID char(4) NOT NULL,
Foreign Key (UserID) References User(UserID),
RoleName varchar(40) NOT NULL);

create table Account(
AccountID char(6) NOT NULL PRIMARY KEY);

create table PersonalAccount(
PersonalActID char(6) NOT NULL PRIMARY KEY,
Foreign Key (PersonalActID) References Account(AccountID),
UserID char(6) NOT NULL,
Foreign Key (UserID) References User(UserID),
SubscriptionFee Integer,
SubscriptionDuration Integer);

create table BusinessAccount(
BusinessActID char(6) NOT NULL PRIMARY KEY,
Foreign Key (BusinessActID) References Account(AccountID),
EmployerID char(6) NOT NULL,
Foreign Key (EmployerID) References Employer(EmployerID),
SubscriptionFee Integer,
SubscriptionDuration Integer);
```

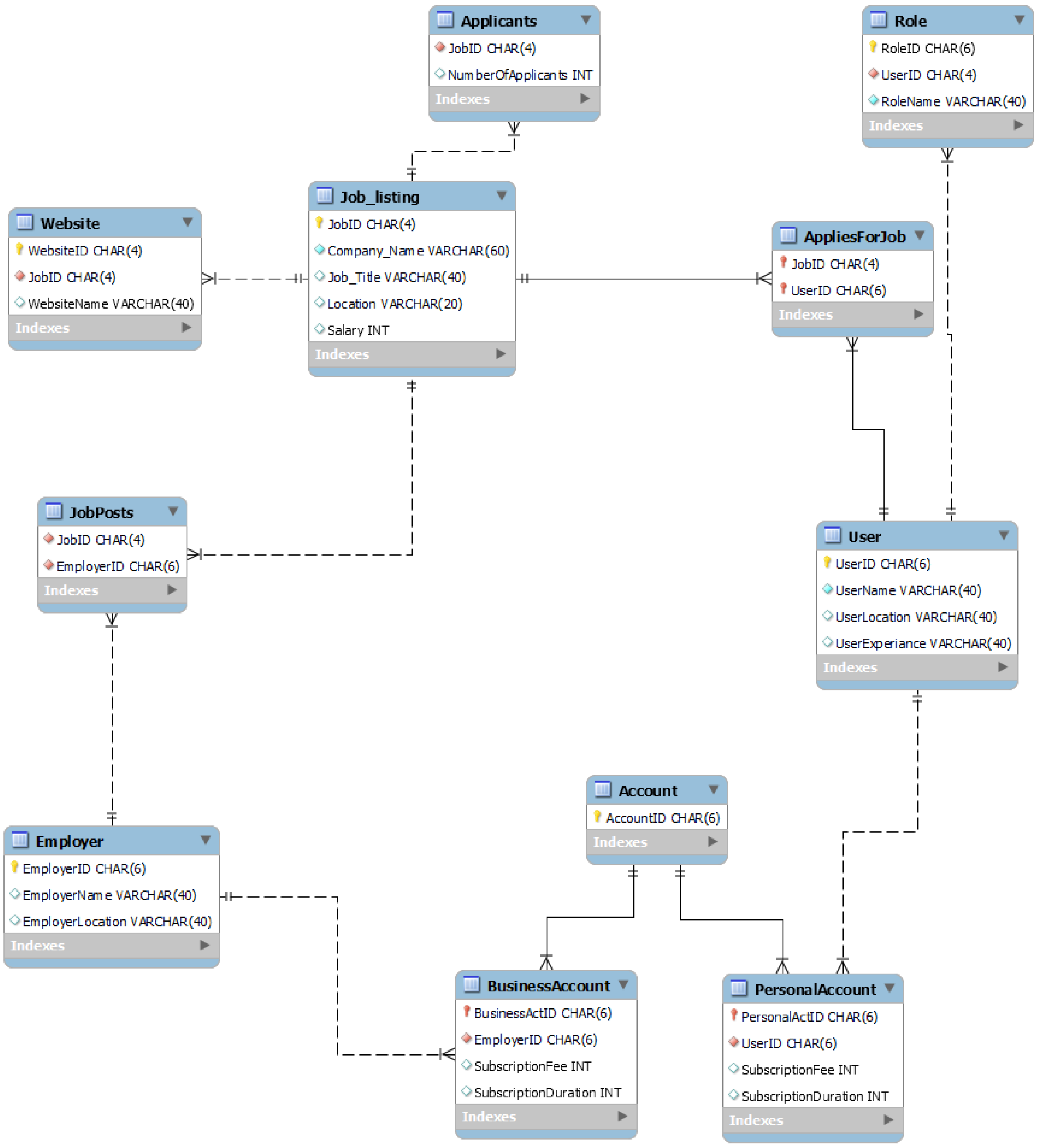
Normalization steps:

The above relational model is in the first normal form (1NF) as each table has a primary key and each attribute has a single value, and each column has atomic values. There are no repeating groups in any column.

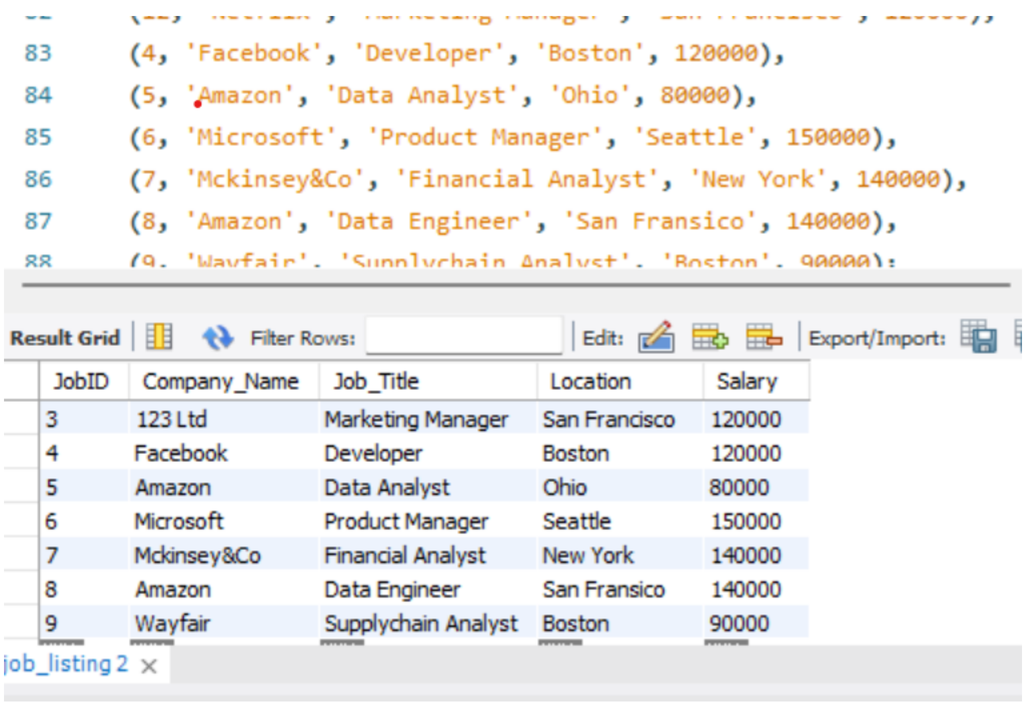
The above relational model is also in the second normal form (2NF) as all the non key attributes are dependent on the primary key. For ex: Website, Applicants, Role, and Personal Account tables were already in 2NF, as they only contain single-valued attributes dependent on their respective primary keys. Job Listing is also already in 2 NF.

The above relational model is also in the third normal form (3NF) there are no transitive dependencies between non-key attributes. The model is in 3.5 NF because There are no join dependencies between composite candidate keys. All tables have a single attribute primary key, so there are no composite candidate keys. Therefore the model is in 3.5NF

The schema diagram after creating the tables from the relational model is :



We also populated some sample data into our tables using Insert into function :



```
(4, 'Facebook', 'Developer', 'Boston', 120000),
(5, 'Amazon', 'Data Analyst', 'Ohio', 80000),
(6, 'Microsoft', 'Product Manager', 'Seattle', 150000),
(7, 'Mckinsey&Co', 'Financial Analyst', 'New York', 140000),
(8, 'Amazon', 'Data Engineer', 'San Fransico', 140000),
(9, 'Wayfair', 'Supplychain Analyst', 'Boston', 90000);
```

JobID	Company_Name	Job_Title	Location	Salary
3	123 Ltd	Marketing Manager	San Francisco	120000
4	Facebook	Developer	Boston	120000
5	Amazon	Data Analyst	Ohio	80000
6	Microsoft	Product Manager	Seattle	150000
7	Mckinsey&Co	Financial Analyst	New York	140000
8	Amazon	Data Engineer	San Fransico	140000
9	Wayfair	Supplychain Analyst	Boston	90000