JOB PORTAL DATABASE SYSTEM

Team Number 17

Harshal Sanjiv Patil
UBID: hpatil2

Prathamesh Kishor Gadgil
UBID: pgadgil

MILESTONE 1

I. Introduction:

In today's world, job search and job recruitment processes have become complex and time consuming for both applicants and recruiters as there is no centralized system available.

Handling these processes manually will require significant time investment and come up with the risk of errors. To address this challenge, a centralized database system is necessary for streamlining job applications, achieving accuracy along with enhancing experience.

This project aims to create a database system that effectively manages recruitment process, applicants and status. This system will come up with functionality like job postings, locations, applications tracking, skilled based filtering.

II. Problem Statement:

Manually managing applications, applicants' profile and job postings is unproductive, prone to mistakes, and does not scale effectively. It is challenging to track applicants for each candidate, to match people to their skill matching jobs, challenging to maintain job details and maintain data accuracy for each candidate as well as firm that post applications.

The process can be streamlined with the help of a database system, which can handle large datasets effectively and efficiently, enable sophisticated searches, and ensure data integrity through organized relationships. A database offers scalability, precision, and automation in contrast to manual approaches, increasing the effectiveness of hiring for administrators, employers, and job searchers.

III. Reason for choosing a database:

Scalability: Large datasets can be handled using databases, as opposed to Excel, which may have performance problems and make managing large datasets more difficult.

Data Integrity: To ensure correct and consistent data, databases impose limitations (such as primary keys and foreign keys)

Complex Queries: Advanced SQL queries, such as JOIN and GROUP BY, are supported by databases to facilitate effective data retrieval and analysis.

Normalization: By normalizing data, databases enable effective storage and remove redundancies.

Backup and recovery: To stop data loss, databases provide automated backup and recovery features.

Integration: Web apps, mobile apps, and analytics tools all easily interface with databases.

IV. Intended user of the database and administration:

The database system will serve the following users:

Job Seekers: Those who look for work and submit applications.

Use Case:

1. Search for Job by skill, location or category
2. Apply for job and track status
3. Update profile and resume.
4.

Employers: Firms that advertise job vacancies and evaluate applicants.

Use Case

1. Post new job
2. Review application and status update
3.

Administrator: platform managers who maintain data integrity and supervise the system

Use Case:

1. Monitor database for data integrity and consistency along with performance
2. Resolve issues like duplicate entries, fake posting.
3. Manage user accounts and profiles.

A real-life scenario:

A job portal for growing tech industry

Job seekers: Prathamesh, a data engineer, log into a portal to search for jobs in New York. He found a job at Newbolt, applied and tracked his application.

Employers: Newbolt, a tech company, posted for job based on New York who required skills like 'Python', 'SQL' etc.

Administrator: Patil, the system administrator, monitors database for performance issues. He found duplicate job postings and removed them.

V. E/R Diagram

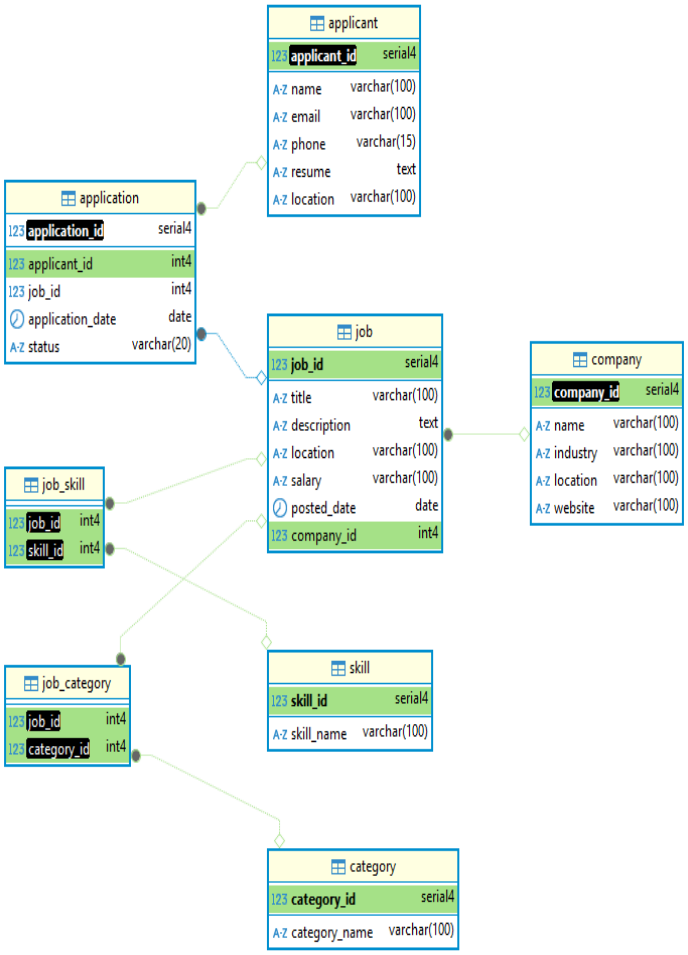

Fig 5.1

VI. Relations and Attributes

In order to properly comprehend the dataset and create the database, we tried to gather real world datasets along with adding newly generated data and determined which table and relationships were required. We have generated several tables for Milestone 1 and specified their properties and connections.
The following are the tables and relationships.

Table 1: Applicant Table
Attributes: applicant_id, name, email, phone, resume, location
Relations: This is the main table for storing information related to job applicants.
The *applicant_id* is the primary key, which helps to identify each applicant uniquely.

Table 2: Company Table
Attribute: company_id, name, industry, location, website
Relations: This table is for storing information related to the company/firms who's going to post the job.
Company_id is a primary key, which is unique for each individual company. Same *company_id* is used as foreign key in Job table to link jobs with its company who posted a job.

Table 3: Job Table
Attribute:
job_id, title, description, location, salary, posted_date, company_id
Relations: This table stores details about job postings. And here is *job_id* is primary key which is ensuring each job posting should be unique.
Company_id is a foreign key which is linked to a company which is posting that job.
This relationship is maintaining one too many relationships meaning one company can post multiple jobs.

Table 4: Application Table
Attribute:
application_id, applicant_id, job_id, application_date, status
Relations: This table store information regarding applicants who applied for which jobs.
The *application_id* is the primary key, which is defined for each unique application made for posted jobs.
And in this table *application_id* and *job_id* are foreign keys that are linked to Applications and Applicants and job respectively for reference and relations.
This table has many to many relationships between Applicants and Job table, which means that one unique applicant can apply for multiple jobs, and one job can have multiple applications.

Table 5: Skill Table
Attribute: skill_id, skill_name
Relations: This table is all about the storing skills required for jobs and acquired by the applicants.
Here *skill_id* is a primary key, for each unique skill
Same *skill_id* us used as foreign key in Job_Skill and Applicant_Skill table for linking specific skills to both job and applicants

Table 6: Job_Skill Table
Attribute: job_id, skill_id
Relations: This table has many to many relationships between jobs and skills.
Job_id is a foreign key that is linked to Job table and *skill_id* is a foreign key that linked to Skill table.

Table 7: Applicant_Skill Table
Attributes: applicant_id, skill_id
Relations: Same as with Job_Skill table this table also has many to many relationships between applicants and skills table.

The *applicant_id* is a foreign key that links to Applicant table. And *skill_id* is the foreign key for Skill table
This table allows applicants to have multiple skills and skills can be acquired by multiple applicants.

Table 8: Category Table
Attribute: category_id, category_name
Relations: Category table stores categories of job or company.
The *category_id* is a primary key which is unique to each category. And it is also used as a foreign key in the Job_category table for linking jobs with their specific categories.

Table 9: Job_category Table
Attribute: job_id, category_id
Relations: This is also a many to many relationships between job and categories. Here *Job_id* and *category_id* both served as primary keys.
Along with that *job_id* is a foreign key for Job table and *category_id* is a foreign key linked to Category table.
This table can have jobs with multiple categories and a category can have multiple job posted.

VII. Description of attributes

The following table represents the detailed description of each attribute including the purpose, data type as well as default values.

1. Applicant Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
applicant_id (Auto generated)	SERIAL	Unique numeric identifier assigned to each applicant. Making sure each applicant has unique reference in database.
Name	VARCHAR (100)	Full name of applicant. Generally, this stores 1st and last name of applicant
email	VARCHAR (100)	Email address of applicant. This must be unique to avoid duplicating accounts.
Phone	VARCHAR (15) (Nullable)	Contact number of applicants. This field can be optional and can be used for communication
resume	TEXT (Nullable)	Detailed summary of applicant's qualification, experience, skills, etc. Optional to filled by applicant.
location	VARCHAR (100) (Nullable)	Geographical location like City, state. Optional to be filled by applicant

2. Company Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
company_id (Auto generated)	SERIAL	Unique numerical value assigned to each company. Making sure each company is referred to by unique identifier.
name	VARCHAR (100)	Registered name of company/firm.
industry	VARCHAR (100) (Nullable)	Industry in which company is working. (e.g. IT, Finance). This field is used for filtered based search.
location	VARCHAR (100) (Nullable)	Geographical location of company. This field help applicant location-based job search.
website	VARCHAR (100) (Nullable)	Official website URL of company. Optional and will provide extra info regarding company.

3. Job Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
job_id (Auto generated)	SERIAL	Unique numerical value assigned to each unique job posted. This filed is making sure that there is no duplicate job posted.

title	VARCHAR (100)	It's a title (role) of Job posted (e.g. SDE1, Intern). It will provide brief description of job role
description	TEXT (Nullable)	This provides detailed descriptions of job responsibilities, their requirements as well as qualifications. It will help applicants to know the requirements of the job.
location	VARCHAR (100) (Nullable)	Geographical location of job. Help applicants to search job based on location.
salary	NUMERIC (Nullable)	Its salary that offered. This field can be optional and will provide info regarding how much compensation will be provided for this role
posted_date (Current date)	DATE	It's a date on which job was posted.
company_id	INT	A foreign key reference to Company table. It's a reference link for the company who posted that job.

4. Application Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
application_id (Auto generated)	SERIAL	Unique numerical value for identifying each application make. It makes sure each application is unique and avoids duplicate entries
applicant_id	INT	It's a foreign key reference to Applicant table. Its application links to its applicant.
job_id	INT	Foreign key referencing to Job table. Linking each application to respective jobs.
application_date (Current date)	DATE	It's a date on which application was submitted.
Status ('Pending')	VARCHAR (20)	This field shows current state of application (e.g. 'Pending', 'Accepted'. This helps to track progress of application.

5. Skill Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
skill_id (Auto generated)	SERIAL	Unique numerical value assigned to each skill. Making sure each skill is referencing uniquely
skill_name (None)	VARCHAR (100)	Its names of skills (e.g. Java, Python, SQL). This filed stored names of skilled required for job as well as acquired by applicants.

6. Job_Skill Table

Attribute & its Default Value	**Data Type & Nullable?**	**Purpose**
job_id	INT	A foreign key for reference of Job table. This links to the skills that are required for the job.
skill_id	INT	A foreign key referenced to Skill table. And this liked to job to skill required for it.

7. *Applicant_Skill Table*

Attribute & its Default Value	Data Type & Nullable?	Purpose
applicant_id (None)	INT	A foreign key for reference of Applicant table. This field links the skills with applicants who have acquired it.
skill_id (None)	INT	Again, a foreign key referencing to Skill table. This links applicant to skill they have.

8. *Category Table*

Attribute & its Default Value	Data Type & Nullable?	Purpose
category_id (Auto generated)	SERIAL	It's a unique numerical value to each category. Making sure each category is uniquely identified
category_name	VARCHAR (100)	This field has categories names. This field categorizes jobs for better filtering.

9. *Job_Category Table*

Attribute & its Default Value	Data Type & Nullable?	Purpose
job_id	INT	A foreign key referring to Job table which links category to job it belong.
category_id	INT	A foreign key referring to the Category table which links Job to category it belongs.

VIII. What happens when Primary key is deleted.

a. No action: On delete action of Primary key it won't delete foreign key and give an error.
b. Delete cascade: All foreign keys connected to that primary key will get deleted.
c. Set null: All foreign keys related to the primary key will be set to null.
d. Set default: It will set foreign keys value to a default value.

If we delete a company from our job portal database, it also deletes all jobs posted by that company. This is an example of a delete cascade.

IX. Query Application

1. Fig 8.1 shows a query which will find jobs that match a specific skill, location, or category using JOINs.

Query | Query History

```
SELECT j.job_id, j.title, j.location, c.category_name, s.skill_name, j.description
FROM job j
JOIN job_category jc ON j.job_id = jc.job_id
JOIN category c ON jc.category_id = c.category_id
JOIN job_skill js ON j.job_id = js.job_id
JOIN skill s ON js.skill_id = s.skill_id
WHERE s.skill_name = 'Python' OR j.location = 'New York, NY' OR c.category_name = 'Data Science';
```

Data Output | Messages | Notifications

	job_id bigint	title text	location text	category_name text	skill_name text	description text
1	151	Social worker	New Hannah, PW	IT	Python	Blue growth represen
2	150	Dietitian	Lake Michael, PR	IT	Python	Course anything fly p
3	72	Chartered management accountant	New Donaldfurt, IA	IT	Python	Since rather store su
4	22	Media buyer	New Rachel, ID	IT	Python	Senior address check
5	1	Environmental health practitioner	Greeneton, MD	IT	Python	Coach local beat or.
6	110	Agricultural engineer	East Christopher, FM	Finance	Python	Little focus central s
7	15	Biochemist, clinical	Port Ronnie, MO	Finance	Python	By despite big state.
8	165	Research scientist (medical)	North Jennifer, MN	Tech	Python	Discuss keep add ca
9	55	Psychologist, educational	Rodriguezton, MI	Tech	Python	Second shake mome
10	199	Games developer	West Zachary, MI	Data Science	Machine Learning	Worker amount black
11	178	Teacher, music	North Jeremy, GA	Data Science	Node.js	Score environmental

Fig 8.1

2. Fig 8.2 shows how employers can use this to track how many applications each job received, sorted from highest to lowest.

Query Query History

```
SELECT j.title, c.name AS company_name, COUNT(a.application_id) AS total_applications
FROM application a
JOIN job j ON a.job_id = j.job_id
JOIN company c ON j.company_id = c.company_id
GROUP BY j.title, c.name
ORDER BY total_applications DESC;
```

Data Output Messages Notifications

	title text	company_name text	total_applications bigint
1	Geoscientist	Huffman-Underwood	24
2	Publishing rights manager	Webb, Grant and Brown	24
3	Advertising account planner	Porter PLC	23
4	Engineer, structural	Webb, Grant and Brown	22
5	Fitness centre manager	Wilson-Trujillo	21
6	Environmental health practitioner	Davis, Avila and Fleming	20
7	Clinical scientist, histocompatibility and immunogenetics	Wilson-Trujillo	19
8	Mudlogger	Knight, Schmidt and Thomas	17
9	Financial controller	Huffman-Underwood	16
10	Bonds trader	Rodriguez PLC	14

Fig 8.2

3. Fig 8.3 shows how job seekers can track their application status using a subquery to find their applicant_id from the email.

Query Query History

```
SELECT a.application_id, j.title, a.application_date, a.status
FROM application a
JOIN job j ON a.job_id = j.job_id
WHERE a.applicant_id = (SELECT applicant_id FROM applicant WHERE email = 'espinozadeborah@example.org')
```

Data Output Messages Notifications

	application_id bigint	title text	application_date text	status text
1	7	Publishing rights manager	2025-03-01	Rejected
2	11	Advertising account planner	2025-02-17	Accepted
3	45	Financial controller	2025-01-25	Rejected
4	56	Clinical scientist, histocompatibility and immunogenetics	2025-02-05	Accepted
5	61	Clinical scientist, histocompatibility and immunogenetics	2025-02-24	Pending
6	72	Advertising account planner	2025-01-07	Rejected
7	79	Mudlogger	2025-01-28	Pending
8	90	Advertising account planner	2025-01-09	Rejected
9	99	Geoscientist	2025-02-24	Rejected
10	103	Advertising account planner	2025-02-10	Accepted
11	106	Advertising account planner	2025-01-01	Accepted
12	107	Fitness centre manager	2025-01-25	Rejected
13	117	Geoscientist	2025-01-11	Pending

Fig 8.3

4. Fig 8.4 shows how administrators can find duplicate job listings where the same company posts a job with the same title multiple times.

Query Query History

```
SELECT j1.job_id, j1.title, j1.company_id, j1.posted_date
FROM job j1
WHERE EXISTS (
    SELECT 1 FROM job j2
    WHERE j1.title = j2.title
    AND j1.company_id = j2.company_id
    AND j1.job_id <> j2.job_id
);
```

Data Output Messages Notifications

	job_id bigint	title text	company_id bigint	posted_date text
1	26	Theatre stage manager	3	2025-02-04
2	52	Trading standards officer	1	2025-02-19
3	82	Development worker, commu...	2	2025-01-05
4	99	Trading standards officer	1	2025-01-24
5	128	Development worker, commu...	2	2025-02-06
6	129	Operational investment banker	7	2025-01-30
7	153	Research scientist (medical)	5	2025-01-14
8	162	Operational investment banker	7	2025-01-25
9	165	Research scientist (medical)	5	2025-01-11
10	189	Theatre stage manager	3	2025-02-11

Fig 8.4

X. Code for database generation

We have used faker to generate fake data for our project. Below is a snapshot of how we generate data for applicants.

```python
import pandas as pd
from faker import Faker
import random

# Initialize Faker
fake = Faker()

# Function to generate applicant data
def generate_applicants(num=200):
    applicants = []
    for i in range(1, num + 1):
        applicant = {
            "applicant_id": i,
            "name": fake.name(),
            "email": fake.email(),
            "phone": fake.phone_number()[:15],
            "resume": fake.text(max_nb_chars=200),
            "location": fake.city() + ", " + fake.state_abbr()
        }
        applicants.append(applicant)
    return pd.DataFrame(applicants)
```

Fig 9.1