3/28/2025

Final Project
Job Application Tracker

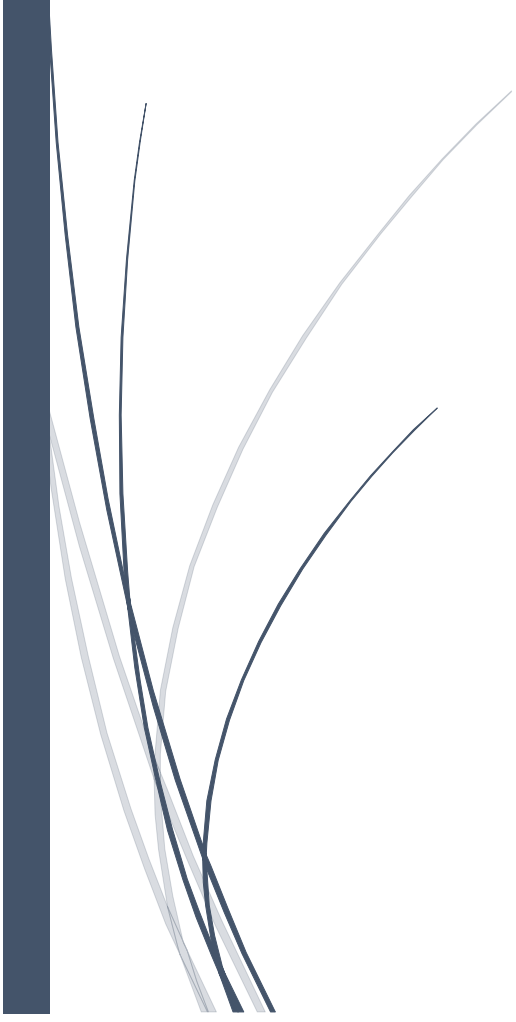


Table of Contents

1.Introduction

In today's competitive job market, keeping track of multiple job applications, interviews, and offers can become an overwhelming task, especially for international students and job seekers navigating new systems. Like many others, I found myself struggling to manage application statuses, interview schedules, follow-up reminders, and documents using scattered notes or basic spreadsheets. This disorganized process often led to missed deadlines, forgotten tasks, or duplicated efforts.

To address this challenge, I decided to develop a **Job Application Tracker**, a centralized platform to help job seekers efficiently organize their job search process. The system is designed to provide users with a structured way to record job applications, track interview progress, store offer details, upload related documents like resumes or cover letters, and set timely reminders for follow-ups.

This project was not just a course requirement, it reflects a personal need. As an international student actively seeking internships and part-time opportunities, I realized how helpful such a tool could be. The system aims to reduce stress and improve decision making by allowing users to see their overall progress, filter job outcomes, and learn from past application patterns.

The purpose of this project is to design and implement a relational database that supports the functionalities of the tracker and enables data analysis through metrics and reports. The ultimate goal is to offer a simple, user-friendly backend database that can support a front-end interface in the future, making job tracking easier, smarter, and more data-driven.

2. Business Analysis:

2.1 Business Problem

Job seekers, especially students or early-career professionals, often struggle to manage the complex process of searching for jobs. Tracking applications manually using spreadsheets or notes can lead to missed interviews, forgotten follow-ups, and disorganized records of job offers and rejections. There's a clear need for a tool that helps users organize this process in one place, offering structure, reminders, and insights.

2.2 User Personas

- ❖ Emma (New Graduate): Recently graduated and actively applying to multiple entry-level roles. Needs a system to track the companies she's applied to and set reminders for follow-ups and interviews.
- ❖ Michael (Experienced Professional): Switching careers and wants to keep detailed notes on companies, interview outcomes, and compare offers side by side.
- ❖ Sarah (International Student): Balancing studies with a job search, and struggling to track documents, deadlines, and interviews while applying for part-time or co-op positions.

2.3 Primary Use Cases

- A user registers, creates an application for a job.
- The user gets notified/interviewed and logs it.
- The user receives a job offer and accepts or declines it.
- The user uploads documents for each application.
- Reminders are set for follow-ups or interviews.

2.4 Major Data Entities:

- Users
- Job Applications
- Jobs
- Interviews
- Companies

- Job Offers
- Documents
- Reminders

3. Table Design and Analysis:

3.1 Business Rules

The database design for the Job Application Tracker project is represented through an Entity-Relationship (ER) diagram that models the core components of the system. The design follows a normalized approach, ensuring data integrity, minimizing redundancy, and maintaining flexibility for future enhancements.

Entity	Key Attributes	Description
Users	UserID (PK), Name, Email, Password	Stores user account details
Companies	CompanyID (PK), CompanyName, Industry	Holds information about the organizations
Jobs	JobID (PK), CompanyID (FK), JobTitle, Location, SalaryRange	Job postings from different companies
JobApplications	ApplicationID (PK), UserID (FK), JobID (FK), Status, DateApplied	Tracks applications submitted by users
Interviews	InterviewID (PK), ApplicationID (FK), InterviewDate, InterviewType, Outcome, RejectionReason	Interview details linked to applications
JobOffers	OfferID (PK), ApplicationID (FK), SalaryOffered, OfferDate, OfferStatus	Job offer details and final outcomes
Documents	DocumentID (PK), UserID (FK), ApplicationID (FK), FileType, FilePath, UploadDate	Uploaded resumes, cover letters, or related files
Reminders	ReminderID (PK), ApplicationID (FK), ReminderType, ReminderDate	Follow-up reminders for job applications

3.2 Key Relationships

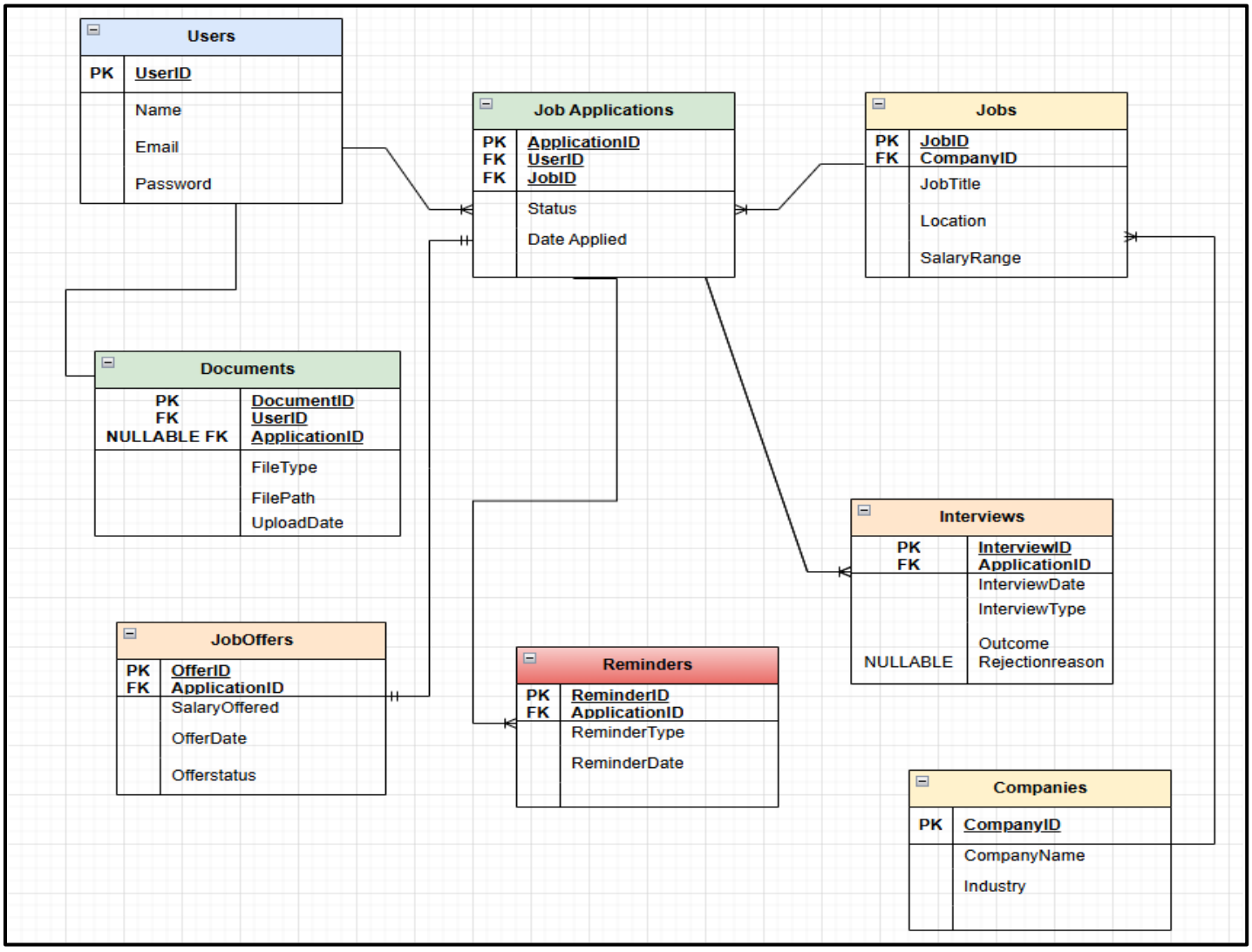
- A User can submit many Job Applications, upload multiple Documents, and set Reminders.
- A Company can post multiple Jobs.
- Each Job can receive multiple Job Applications.
- A Job Application can lead to an Interview, a Job Offer, and be linked with multiple Documents and one Reminder.
- The Interview and Job Offer tables are tied one-to-one with JobApplications, ensuring clarity in the process flow.

3.3 Design Justification

- The use of primary and foreign keys ensures proper referential integrity.
- Normalization helps avoid duplication of data and keeps the structure clean.
- The design supports complex queries such as tracking the application-to-offer timeline, company-wise conversion rates, and rejection trends.
- The structure is scalable and allows for future feature additions, such as recruiter dashboards or AI-based recommendations.

3.4 Entity Relationship Diagram

This ER diagram illustrates the key entities and relationships used in the Job Application Tracker database design, including user accounts, job applications, interviews, job offers, documents, and reminders:



4. Database Implementation:

4.1 Table Overview & Key Fields

The database implementation of the Job Application Tracker is designed using a relational model, with normalized tables to reduce redundancy and maintain data integrity. Each table corresponds to a real-world entity in the job application process, and foreign key constraints enforce consistent relationships between them.

Table Name	Primary Key	Foreign Keys	Purpose
Users	UserID	—	Stores user profile data
Companies	CompanyID	—	Stores company details
Jobs	JobID	CompanyID → Companies(CompanyID)	Stores job postings
JobApplications	ApplicationID	UserID → Users(UserID), JobID → Jobs(JobID)	Tracks user applications
Interviews	InterviewID	ApplicationID → JobApplications(ApplicationID)	Stores interview data
JobOffers	OfferID	ApplicationID → JobApplications(ApplicationID)	Stores offer details
Documents	DocumentID	UserID → Users(UserID), ApplicationID → JobApplications(ApplicationID)	Manages uploaded files
Reminders	ReminderID	ApplicationID → JobApplications(ApplicationID)	Stores follow-up reminders

4.2 Data Types and Field Examples

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Password VARCHAR(100)
);
CREATE TABLE JobApplications (
    ApplicationID INT PRIMARY KEY,
    UserID INT,
    JobID INT,
    Status VARCHAR(50),
    DateApplied DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
);
```

Similar definitions can be created for Interviews, JobOffers, Documents, and Reminders.

4.3 Keys and Constraints

- Primary keys uniquely identify each record in every table.
- Foreign keys maintain referential integrity between tables (e.g., UserID, ApplicationID).
- Unique constraints (such as on Email) prevent duplication.
- Not Null constraints ensure essential fields (e.g., Status, OfferDate) are always filled.

4.4 Normalization

The database follows at least 3rd Normal Form (3NF):

- No repeating groups in tables.
- Non-key attributes depend only on the primary key.
- No transitive dependencies to reduce redundancy.

This ensures clean, scalable data storage and efficient querying.

4.5 Example Use Case: Sara Applies for a Job and Tracks Her Progress

Step 1: View all jobs Sara has applied to, along with job titles and application status

```
SELECT ja.ApplicationID, j.JobTitle, ja.Status, ja.DateApplied
FROM JobApplications ja
JOIN Jobs j ON ja.JobID = j.JobID
JOIN Users u ON ja.UserID = u.UserID
WHERE u.Email = 'sara@example.com';
```

Step 2: See upcoming interviews for Sara's applications

```
SELECT j.JobTitle, i.InterviewDate, i.InterviewType, i.Outcome
FROM Interviews i
JOIN JobApplications ja ON i.ApplicationID = ja.ApplicationID
JOIN Jobs j ON ja.JobID = j.JobID
JOIN Users u ON ja.UserID = u.UserID
WHERE u.Email = 'sara@example.com' AND i.InterviewDate >= CURRENT_DATE;
```

Step 3: Check reminders Sara has set for follow-ups or interviews

```
SELECT r.ReminderType, r.ReminderDate, j.JobTitle
FROM Reminders r
JOIN JobApplications ja ON r.ApplicationID = ja.ApplicationID
JOIN Jobs j ON ja.JobID = j.JobID
JOIN Users u ON ja.UserID = u.UserID
WHERE u.Email = 'sara@example.com' AND r.ReminderDate > CURRENT_DATE;
```

Step 4: View companies that gave Sara job offers and their salary

```
SELECT c.CompanyName, j.JobTitle, jo.SalaryOffered, jo.OfferDate,
jo.OfferStatus
FROM JobOffers jo
JOIN JobApplications ja ON jo.ApplicationID = ja.ApplicationID
JOIN Jobs j ON ja.JobID = j.JobID
JOIN Companies c ON j.CompanyID = c.CompanyID
JOIN Users u ON ja.UserID = u.UserID
WHERE u.Email = 'sara@example.com';
```

Step 5: Update an application status to reflect that Sara accepted an offer

```
UPDATE JobApplications
SET Status = 'Offer Accepted'
WHERE ApplicationID = 1012;
```

5. Analytics, Reports, and Metrics:

The Job Application Tracker database is designed not only for efficient data storage but also to provide meaningful insights that help job seekers make informed decisions during their job search journey. This section outlines **key business questions** that can be answered using structured queries and **data pulled across multiple related tables**.

5.1 10 Key Business Questions (General)

1. How many applications has each user submitted?
2. What is the interview success rate per user?
3. Which companies have posted the highest number of jobs?
4. What is the average salary offered by job title?

5. How many applications have been submitted in the last 30 days?
6. What is the overall offer acceptance rate?
7. Which application statuses are most common (e.g., pending, rejected)?
8. What is the average time from application to interview?
9. Which industries are generating the most job offers?
10. What are the top reasons for interview rejections?

Tables used: Users, JobApplications, Interviews, JobOffers, Companies, Jobs

5.2 5 Advanced Business Questions (Using Joins & Filters)

1. What is the average time taken for an application to move from submission to a job offer?

- Tables: JobApplications, JobOffers
- Logic: Calculate the difference between DateApplied and OfferDate for each ApplicationID.

2.Which companies have the highest interview-to-offer conversion rate?

- Tables: Companies, Jobs, JobApplications, Interviews, JobOffers
- Logic: Join applications → interviews → offers; group by CompanyID.

3.What percentage of applications are withdrawn before reaching the interview stage?

- Tables: JobApplications, Interviews
- Logic: Filter applications where no interview is linked, and status is ‘Withdrawn’.

4.What are the most common rejection reasons during interviews?

- Table: Interviews
- Logic: Group by RejectionReason and count occurrences.

5.What time of year sees the highest job application success rates?

- Tables: JobApplications, JobOffers
- Logic: Join both tables and group offers by OfferDate month/quarter.

5.3 Sample Reports That Could Be Generated

- Monthly Job Application Summary by user
- Company Offer Conversion Rates
- Most Frequent Rejection Reasons
- Salary Distribution Report by job title
- Follow-Up Reminders Dashboard

These reports would help users better understand their progress, optimize their job search strategy, and identify patterns in their application outcomes.

6. Security Concerns:

As the Job Application Tracker deals with sensitive user data—including contact information, application history, and confidential documents such as resumes and job offers—implementing strong security measures is critical. The system must be designed to safeguard user privacy, prevent unauthorized access, and ensure data integrity.

6.1 Key Security Measures

1.Authentication and Authorization

- Only registered users can access their application data.
- Passwords are stored in the database in hashed form (e.g., using bcrypt or SHA-256).
- Future versions may include role-based access control for recruiters or career advisors.

2. Data Privacy

- Uploaded documents (e.g., resumes, offer letters) should be stored securely, either encrypted in the database or linked via a secure file system.
- Personally Identifiable Information (PII) such as name, email, and login credentials is protected using encryption and secure transmission protocols (e.g., HTTPS).

3.Input Validation and SQL Injection Prevention

- All user inputs should be sanitized to avoid SQL injection vulnerabilities.
- Use of prepared statements or ORM frameworks is recommended.

4.Backup and Recovery

- Regular database backups are essential to prevent data loss in case of system failure.
- Backup policies should include automated daily backups and disaster recovery testing.

5.Audit Logs (Future Enhancement)

- Tracking login history, document uploads, or offer changes can help identify unauthorized activity.
- Logs can be stored in a separate secure table for administrative review.

6.2 Potential Risks if Security is Weak

- Data breaches exposing users' job search history and personal information.
- Loss of trust among users if documents or job offers are leaked.
- Regulatory consequences if data privacy laws (e.g., GDPR) are violated in a full-scale product.

7. Architecture:

The Job Application Tracker is designed using a modular, relational architecture to support job seekers in managing their application process from start to finish. The system can be built as a web-based application, supported by a robust backend database, with a potential for future mobile integration.

7.1 High-Level Architecture Components

1.Frontend (User Interface)

- Built using web technologies like HTML, CSS, and JavaScript (or frameworks like React in future).
- Allows users to log in, submit applications, upload documents, view reminders, and track progress.
- Could include dashboard components like charts and reminders.

2.Backend (Application Layer)

- Handles business logic and communication between the frontend and the database.
- Could be implemented using Node.js, Python (Flask/Django), or Java (Spring Boot).
- Responsible for validation, security, data processing, and analytics logic.

3.Database (Data Layer)

- Central relational database (e.g., MySQL, PostgreSQL, or SQLite for local testing).
- Stores all user data, applications, interviews, documents, reminders, and offers.
- Designed with referential integrity and normalization (as per the ER diagram).

4.Document Storage

- Uploaded documents like resumes or offer letters can be stored:
    - In the database as BLOBs (basic prototype)
    - Or in a file storage system (like AWS S3) with secure paths stored in the DB

5.Security Layer

- Authentication (login system) and access control
- Encrypted password handling
- HTTPS for secure communication

6.Optional: Reporting and Visualization Tools

- Analytics reports could be generated using SQL queries, or connected to a visualization layer using Tableau, Power BI, or a custom frontend dashboard.

8. Conclusion:

The Job Application Tracker database project has been thoughtfully designed to address a real-world challenge faced by many job seekers: managing and organizing their application journey. Through structured data modeling, normalization, and relational design, the system effectively stores and tracks essential information such as user applications, interview schedules, job offers, and reminders.

By using SQL queries and joining related tables, the database enables users to generate actionable insights — from identifying top-performing companies to understanding application-to-offer conversion rates. The project adheres to database best practices by enforcing referential integrity and incorporating necessary security considerations for handling sensitive information like resumes and personal contact details.

This project demonstrates how a well-structured database can serve as the foundation for an intelligent and user-centered solution, and sets the stage for future growth, analytics, and integration with broader application ecosystems.