DEBRE BERHAN UNIVERSITY

COLLEGE OF COMPUTING

Department Of Software Engineering

JOB SEEKING SITE PROJECT

(ADVANCED OOP)

Group Two

Name **ID**

1. Yishaq Dmatew ----------DBU1601755

2. Yirgalem Zegeye ----------DBU1601753

3. Ahmed Seid ----------DBU1601509

4. Abrham Belay ----------DBU1601483

Submitted To: Zerihun T.(Mr)

Submission Date: Dec 23 -2025

Table Of Contents

PROJECT TITLE: JOB SEEKER SITE

1. Introduction

The rapid development of information technology has caused a shift from conventional recruitment systems to web-based systems. This has made job portals significant for providing easy access between employers and job seekers. Conventional recruitment systems are prone to errors, time-consuming, and lack transparency. Therefore, an automated job portal system is necessary.

The Online Job Portal Management System is a desktop application that uses JavaFX as its GUI and PostgreSQL as its database. This system serves as a platform that enables job seekers to search and apply for jobs online. Similarly, the system enables companies to manage job vacancies. Furthermore, the system enables the administration to manage the whole system.

This project incorporates things such as employer approvals, CV handling, job applications, and notification alerts, all aiming to provide safety, transparency, and efficiency.

2. Problem Statement

Indeed, the job market is highly competitive, and job seekers feel overwhelmed due to disorganized job postings, while employers have difficulty finding potential candidates. Traditional job sites have unintuitive user interfaces, lack real-time notification systems, and inadequate admin tools, leading to delays in hiring and poor matching results.

Some of the major issues that

- Job seekers who cannot efficiently search jobs and monitor their submissions.
- Candidates finding it challenging to look for, view, and apply to jobs.
- Admins without the ability to approve or manage system content.
- Lack of notification for critical events, such as new registrations or applications.

It is, therefore, clear that the proposed system, JobSeekerApp, tackles the challenges by offering a seamless and multifaceted platform with role-based access, sophisticated searching, resume organization, and notification systems to improve efficiency and user satisfaction.

Furthermore, the traditional method for recruiting employees has some flaws such as:

- Job posting and application processing by hand.
- There is no collective pool of
- Issues in tracking job applications.
- There will be no

- Poor communication between employers and job seekers.
- The lack of real-time

These problems result in inefficiency, inconsistencies in data, and lack of trust in the recruitment process. Therefore, the need for a secure, automated, and well-managed job portal system that offers a guarantee of genuine employers, organized employment applications, and up-to-date information is imperative.

3. Objectives

Job SeekerApp has the following main goals as part of the Job SeekerApp project:

- To design a job portal system that is central, secure, and user-friendly for job seekers, employers, and administrators.
- To empower job seekers to conveniently search and apply for suitable job vacancies based on sophisticated parameters such as job name, geographic position, and remuneration.
- For employers to be able to post, edit, and manage job listings.
- To ensure that employers must be approved and verified in order to be allowed to post job advertisements on the site.
- For job seekers, providing adequate capabilities for managing resumes such as uploading, editing, and monitoring resumes during applications.
- To apply severe role-based access control, whereby users can only view functions and information related to their role, either Job Seeker, Employer, or Admin role.
- The purpose of creating alerts for critical system events such as registrations, postings, and applications with the aim of keeping all the stakeholders notified in real time.
- In order to provide data integrity, security, and privacy through the use of password hashing and data validation.

These objectives help in developing an efficient, trustworthy, and engaging platform with respect to the primary needs of all users in the recruitment process**.**

4. Scope of the System

JobSeekerApp focuses on essential job portal features for desktop use, emphasizing core functionalities for its three main user roles while excluding advanced or non-essential elements to maintain a focused Minimum Viable Product (MVP).

In Scope:

- User registration and login with role selection (Job Seeker, Employer, Admin), including secure authentication and role-based redirection.
- Role-based dashboards tailored to each user type, providing personalized overviews and quick access to relevant features.
- Job searching with advanced filters (e.g., title, location, salary range, job type), viewing detailed job listings, and applying for jobs with CV submission.
- Job posting, editing, and deletion by approved employers, along with viewing and managing incoming applications.
- Admin approvals for employer registrations, user management (e.g., role updates, deletions), and CV template preparation.
- Real-time notifications for major events, such as new job postings, applications, and user registrations.
- Attractive JavaFX UI design with icons, gradients, shadows, and responsive elements for an intuitive user experience.

Out of Scope:

- Mobile or web versions of the application, as the system is optimized for desktop use with JavaFX.
- Extra features like video interviews, payment systems for premium listings, or AI-driven job recommendations.
- Integration with external APIs (e.g., LinkedIn authentication, email sending services, or third-party payment gateways).
- User ratings, reviews, or real-time chat functionality between users.

The system is designed for scalability, with PostgreSQL handling data storage and retrieval, supporting up to 10,000 users in its initial MVP phase, while allowing for future expansions.

5. Functional Requirements

The functional requirements of JobSeekerApp are clearly defined for each user role and the notification system.

Admin Functional Requirements

- The system shall provide secure admin login authentication.
- The admin shall be able to view all registered users in the system.
- The admin shall have the ability to approve or reject employer accounts.
- The admin shall be able to view employer subscription documents in PDF format.
- The admin shall have the capability to manage CV templates.
- The admin shall be able to view all system notifications.
- The admin shall be able to generate reports in CSV and PDF formats.

Employer Functional Requirements

- Employers shall be able to register and create an account.
- Employers shall be able to upload subscription documents for verification.
- Employers shall have restricted access until admin approval is granted.
- Approved employers shall be able to post and manage job listings.
- Employers shall be able to view all applications received for each job posting.
- Employers shall be able to download applicant CVs.
- Employers shall be able to manage application statuses (e.g., shortlist, reject).

Job Seeker Functional Requirements

- Job seekers shall be able to register and log in to the system.
- Job seekers shall be able to manage their profile, including updating photo, email, and password.
- Job seekers shall be able to upload and update their CV.
- Job seekers shall be able to search and filter jobs based on title, location, job type, and salary range.
- Job seekers shall be able to apply for jobs by submitting their CV.
- Job seekers shall be able to view their applied job history.

Notification Functional Requirements

- The admin shall be notified automatically when a new employer registers.
- The admin shall be notified automatically when a new job is posted.
- The admin shall be notified automatically when a job seeker applies for a job.

- Job seekers shall be notified automatically when a new job is posted that matches their interests or profile.
- Notification counters shall update dynamically on the dashboards to reflect unread notifications.

6. Non-Functional Requirements

The non-functional requirements ensure the quality, security, and performance of JobSeekerApp.

- **Security:** The system shall implement strict role-based access control and secure user authentication to protect sensitive data.
- **Performance**: The system shall provide fast job search and filtering operations with response times under 2 seconds.
- **Scalability**: The system shall support a large number of users and job postings without performance degradation.
- **Reliability**: The system shall prevent duplicate job applications from the same user for the same job.
- **Usability**: The system shall offer a user-friendly JavaFX interface that is intuitive and visually appealing.
- **Maintainability**: The system shall follow a modular MVC architecture for easy updates and maintenance.
- **Portability**: The system shall run on any platform that supports Java and JavaFX.

These functional and non-functional requirements form the foundation for a secure, efficient, and user-centric job portal system.

7. System Architecture Diagram

JobSeekerApp adopts a **Layered Architecture** based on the **Model-View-Controller (MVC) pattern**, ensuring clear separation of concerns, maintainability, and scalability.

The system is structured into four distinct layers:

Presentation Layer (View)

- Implemented using JavaFX with FXML files for layout and Controllers for handling user interactions.
- Provides role-based graphical user interfaces, including dashboards tailored for Admin, Employer, and Job Seeker.
- Features modern styling, icons, notifications, and responsive navigation.

Business Logic Layer (Controller/Services)

- Contains Service classes (e.g., UserService, JobService, ApplicationService, NotificationService) that encapsulate core business rules and workflows.
- Acts as an intermediary between the Presentation Layer and Data Access Layer.
- Manages operations such as user authentication, job posting, application submission, employer approval, and notification generation.

Data Access Layer (Model/DAO)

- Comprises **Data Access Object (DAO) classes** responsible for database interactions.
- Utilizes **JDBC** to connect to the PostgreSQL database.
- Handles CRUD operations, queries, and transaction management while abstracting database details from upper layers.

Database Layer

- **PostgreSQL** relational database management system.
- Stores all persistent data including users, jobs, applications, CVs, notifications, and CV templates.
- Ensures data integrity through primary keys, foreign keys, constraints, and indexes.

Data Flow

User interactions originate in the **UI** → processed by **Controllers** → business logic executed in **Services** → data operations performed via **DAOs** → queries sent to **PostgreSQL Database** → results returned through the same path in reverse → **UI** updated accordingly.

This layered MVC architecture promotes modularity, facilitates testing and maintenance, and supports future enhancements such as additional features or integration with external systems.

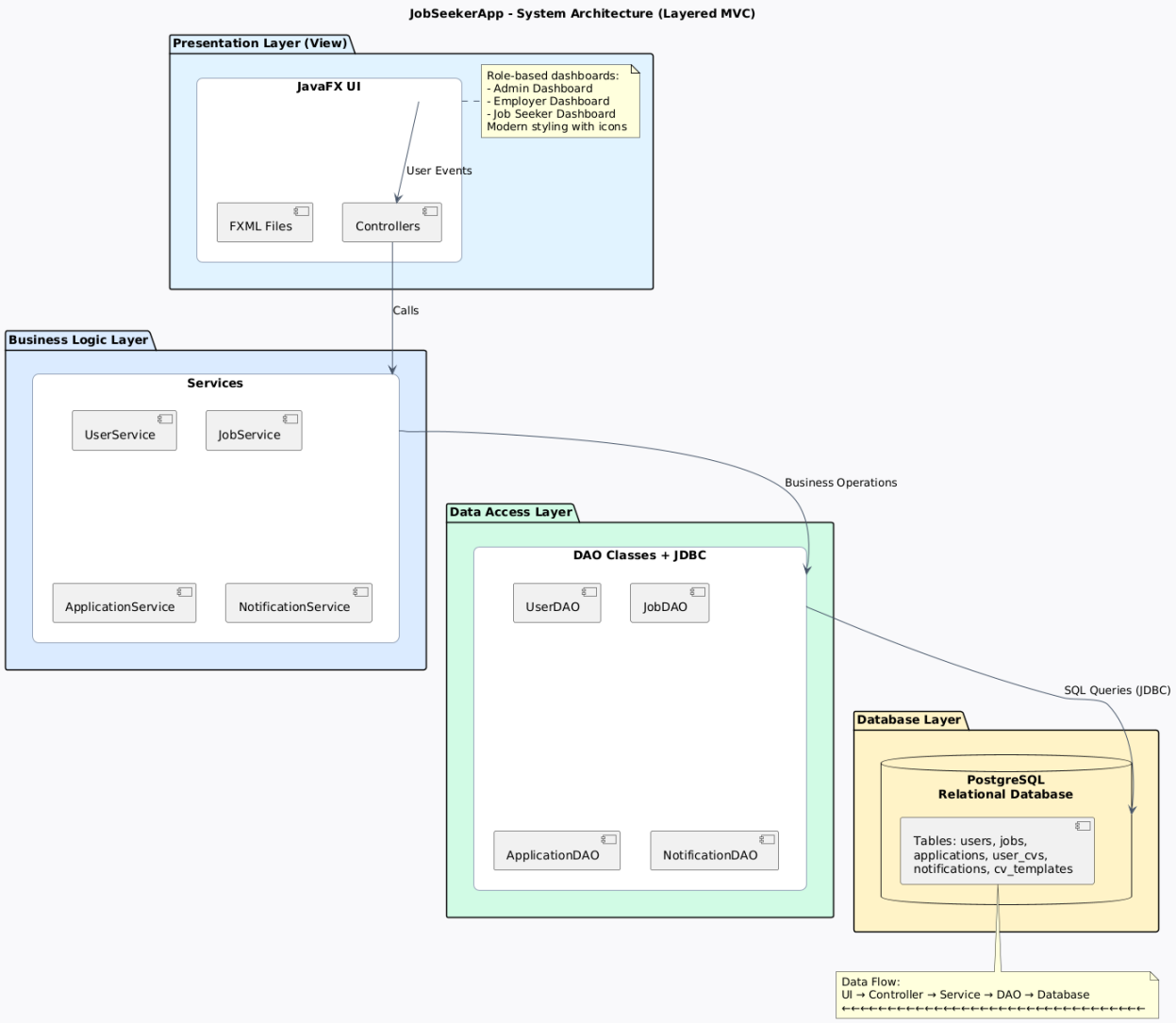

Figure 1: system architecture diagram

8. Use Case Diagram

The Use Case Diagram illustrates the interactions between the system's actors and its primary functionalities.

Actors

- **Admin:** The system administrator responsible for platform oversight and management.
- **Employer:** A company representative who posts jobs and manages applications.
- **Job Seeker:** An individual searching and applying for employment opportunities.

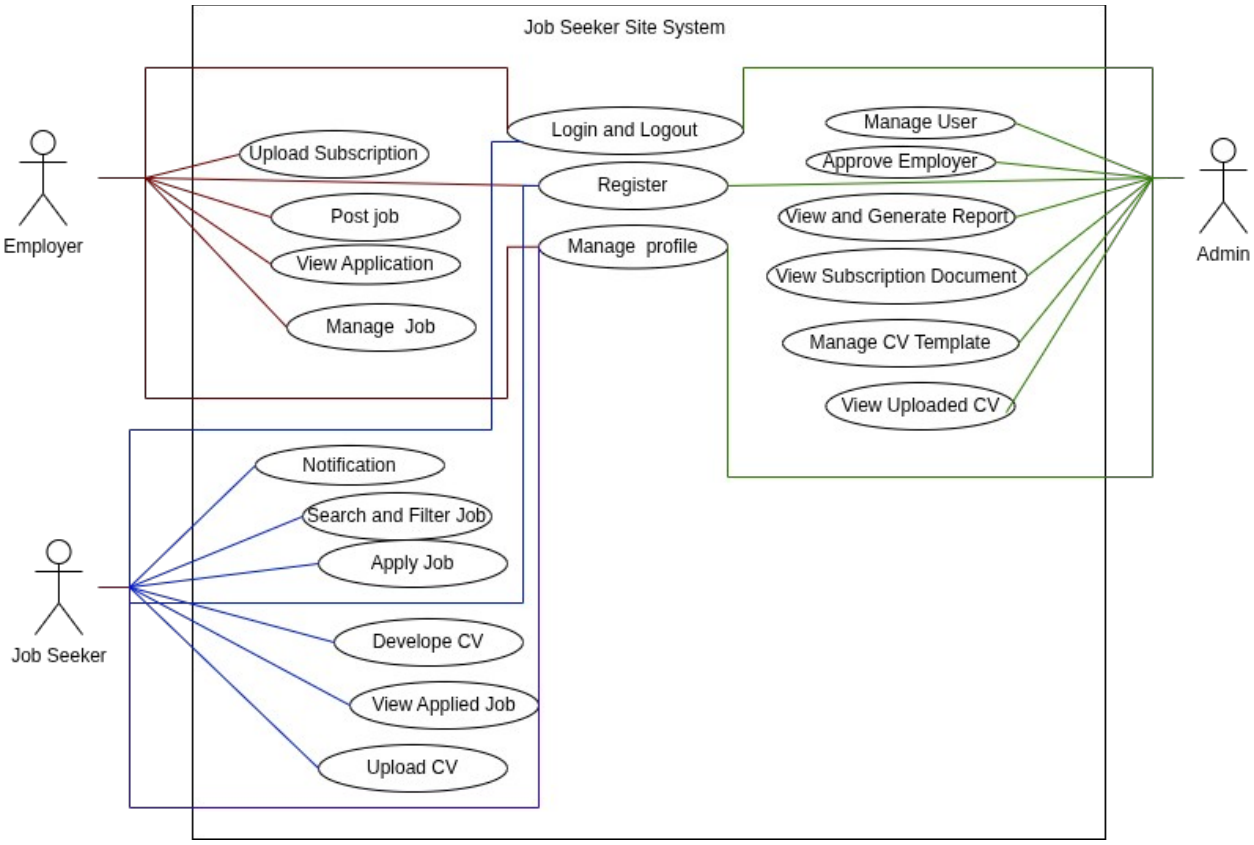

Figure 2: use case diagram

These use cases represent the core interactions within JobSeekerApp, ensuring clear role separation and efficient workflow for all users.

9. Database Design (ER Diagram)

Entities and Attributes

The database schema of JobSeekerApp consists of six main entities, each with specific attributes.

users

The users entity serves as the central table, storing information for all system users regardless of role.

- **id** is the primary key.
- **email** is unique across all users.
- **password_hash** stores the hashed password for security.
- **full_name** holds the user's full name.
- **role defines** the user's role (e.g., JOBSEEKER, EMPLOYER, ADMIN).
- **created_at** records the timestamp of account creation.
- **is_active** indicates whether the account is active.
- **cv_path** stores the path to the user's primary CV (if applicable).
- **approved** indicates whether an employer account has been approved.
- **username** stores an optional username.
- **profile_image** stores the path to the user's profile picture.
- **subscription_doc** stores the path to the employer's verification document.
- **approval_status** tracks the status of employer approval (e.g., PENDING, APPROVED, REJECTED).

user_cvs

The user_cvs entity stores multiple CVs uploaded by job seekers.

- **id** is the primary key.
- **user_id** is a foreign key referencing users.id.
- **file_path** stores the location of the uploaded CV file.
- **file_name** records the original name of the CV file.
- **uploaded_at** records the timestamp of CV upload.
- **is_current** indicates whether this CV is the user's primary one.
- **file_size** stores the size of the uploaded file.

Jobs

The jobs entity represents job postings created by employers.

- **id** is the primary key.
- **employer_id** is a foreign key referencing users.id (the employer).
- **title** contains the job title.
- **description** holds the full job description.
- **job_type** specifies the type of employment (e.g., FULL_TIME, PART_TIME).
- **location** stores the job location.
- **salary** records the offered salary.
- **updated_at** and created_at track modification and creation timestamps.

Applications

The applications entity records job applications submitted by job seekers.

- **id** is the primary key.
- **job_id** is a foreign key referencing jobs.id.
- **user_id** is a foreign key referencing users.id (the job seeker).
- **cv_path** stores the path to the CV submitted with the application.
- **applied_at** records the application timestamp.
- A unique constraint on (job_id, user_id) prevents duplicate applications.

Notifications

The notifications entity manages system alerts sent to users.

- **id** is the primary key.
- **user_id** is a foreign key referencing users.id.
- **role** specifies the recipient's role for filtering.
- **message** contains the notification text.
- **is_read** indicates whether the notification has been read.
- **created_at** records the notification creation timestamp.

CV_templates

The cv_templates entity stores admin-managed CV templates available to users.

- **id** is the primary key.
- **name** holds the template name.
- **description** provides details about the template.
- **file_path** stores the location of the template file.
- **created_at** and uploaded_at record creation and upload timestamps.
- **uploaded_by** identifies the admin who uploaded the template.

Relationships

The entities are interconnected through the following relationships:

- A **one-to-many** relationship exists between users and user_cvs: one user can have many CVs, with cascading deletion on user removal.
- A **one-to-many** relationship exists between users (as employer) and jobs: one employer can post many jobs.
- A **one-to-many** relationship exists between users (as job seeker) and applications: one job seeker can submit many applications.
- A **one-to-many** relationship exists between jobs and applications: one job can receive many applications.

- A **one-to-many** relationship exists between users and notifications: one user can receive many notifications.

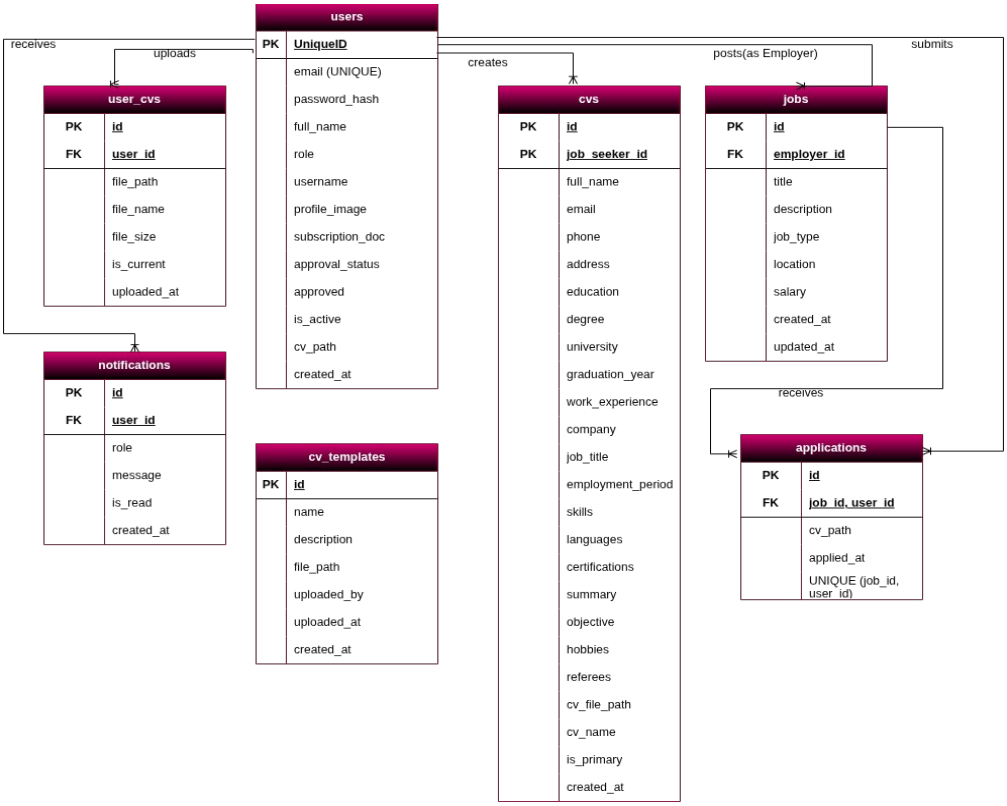

Figure 3: Entity Relationship Diagram

These relationships ensure referential integrity and support the core functionalities of the job portal system.

10. Conclusion

The JobSeekerApp provides an effective solution to the shortcomings that come with the conventional recruitment process by providing a secured and automated online job portal platform. By implementing effective employer verification through administrative approval, job postings, CV management, and online notification systems, the software program improves the recruitment process.

Some key achievements include:

- Correct implementation of role-based access control such that Job Seekers, Employers, and Admins are only able to access functionalities related to their roles.
- System has been designed with modular MVC architecture that helps achieve scalability and easy enhancement.
- Systems-level, seamlessly integrated use of JavaFX for providing a modern, responsive, and aesthetically-rich User Interface with high-class design.
- Support for Effective data management with PostgreSQL, as well as proper entity, relationship, and constraint concepts to ensure the integrity of the retrieved data.

This is a real-life example of applying the concepts of software engineering, right from authentication, designing databases, and layered architectures to designing UI/UX. This application is ideal for implementation in a real-life setting in SMEs, or even in an institution of learning, and can be used as it is or extended for features like supporting mobile or AI-assisted recommendations and other third-party services.

Indeed, the success associated with the implementation of JobSeekerApp makes it an exemplary model of the revolutionary power of well-designed desktop apps in changing the way conventional processes have been followed.