

**Requirement & Design Specification**

**TopCV Career Connect System (TOPCV)**

**Version: 1.0**

– Hanoi, Octorber 2025 –

# Record of Changes

| **Version** | **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- | --- |
| V1.0 | 01/10 | A, M | DuongNT, MinhNT |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\*A - Added M - Modified D - Deleted

Contents

[Record of Changes 2](#_em7utzx7jrsa)

[I. Overview 5](#_308fn0qtn7y4)

[1. User Requirements 5](#_gv5m4mnx8w6v)

[1.1 Actors 5](#_wnrykcg817py)

[1.2 Use Cases 5](#_87p90h4fknop)

[a. Diagram(s) 5](#)

[b. Descriptions 9](#)

[2. Overall Functionalities 11](#_bmdv9kp8bcld)

[2.1 Screens Flow 11](#_4sz92mje9wkk)

[2.2 Screen Descriptions 11](#_qrkc4hwiqucx)

[2.3 Screen Authorization 12](#_f8u27kv40l6e)

[2.4 Non-UI Functions 13](#_hepa8sprdnl)

[3. System High Level Design 14](#_uqg95140oh9t)

[3.1 Database Design 14](#_vpe4uon4hst)

[a. Database Schema 14](#)

[b. Table Descriptions 15](#_xa1dogyilhju)

[3.2 Code Packages 16](#_y0k8pqjq8uud)

[II. Requirement Specifications 18](#_cqf8e5d3ojly)

[1. 18](#_ubygt71g115k)

[1.1 UC01\_Manage company profile 18](#_9aa3e9jbfkr3)

[a. Functionalities 18](#)

[b. Business Rules 22](#)

[1.2 UC03\_Post jobposting 22](#_35sgs312ysc4)

[a. Functionalities 22](#)

[b. Business Rules 26](#_czwcgqq8uz8p)

[1.3 UC04\_Manage application 26](#_nenguhwmnqm8)

[a. Functionalities 26](#_3o125sv41kfc)

[b. Business Rules 28](#_idtb3znoh0a8)

[1.4 UC05\_Send Interview Invitation 29](#_4mnwkezgkbje)

[a. Functionalities 29](#_kgvmwnsdgju4)

[b. Business Rules 31](#_5w5iumbt9xa6)

[1.6 UC06\_Manage jobseeker profile 32](#_hohl56sjgrd1)

[a. Functionalities 32](#)

[b. Business Rules 41](#_amau5z1yz6kl)

[1.7 UC07\_Apply for job 41](#_3wkavfocfh34)

[a. Functionalities 41](#_sbbxqn721qr7)

[b. Business Rules 50](#_sscla98dixwj)

[1.8 UC08\_Manage job 50](#_i0lmxry0zh5j)

[a. Functionalities 50](#_m3z197cwnok7)

[b. Business Rules 55](#_2inp5s9a2zal)

[1.9 UC09\_View company 55](#_xt4c56xmnopj)

[a. Functionalities 55](#_n1wnkqwnyn7t)

[b. Business Rules 60](#_6pk68dfoaujh)

[1.10 UC10\_Notify 60](#_ffps3nqvrpb0)

[a. Functionalities 60](#_cb9s0616k2fp)

[b. Business Rules 65](#_3bt7kytatrq)

[2. Common Functions 65](#_4ckuh4b86wtd)

[2.1 UC-19\_Register Account 65](#_k5pjgsoq4wtu)

[a. Functional Description 65](#)

[b. Business Rules 67](#_19dkvz38iltv)

[2.1 UC-20\_Login System 67](#_h2xksjvozg4w)

[a. Functional Description 67](#)

[b. Business Rules 68](#)

[2.2 UC-21\_Reset Password 68](#_ggz9sgauqy8d)

[a. Functional Description 68](#)

[b. Business Rules 71](#_ywnfxcn039fi)

[3. Patron Feature 72](#_nny1ltpcn55u)

[3.1 UC-2\_Pay for job package 72](#_fbx3fkob7y1z)

[a. Functional Description 72](#)

[b. Business Rules 74](#)

[3.2 UC-15\_Manage payment 75](#_m6a3wx5z4eqy)

[a. Functional Description 75](#)

[b. Business Rules 76](#)

[III. Design Specifications 77](#_x6g30d1yzub8)

[1. Company Management 77](#_g6yrtzimctjr)

[1.1 Update company information 77](#_fe4401yu44fs)

[a. Company Information Screen 77](#)

[2. Job Package Management 79](#_yzzwm0f487ui)

[a. Shop-cart 81](#_t7w2e4cig5p5)

[a. Purchased Packages Dashboard 82](#_6a0ly5e4ue3p)

[3. Job Posting Management 84](#_kvw6u6d2e074)

[3.1 Create job 84](#_nuhf72xbke4k)

[a. Create Job Screen 84](#_p0hkabxrnj1r)

[3.2 Manage job list 88](#_82x5g6wd1ttd)

[a. Job List Screen 88](#_v29gxkmzhvdr)

[4. Application Management 89](#_rp76a7f1lgx2)

[4.1 View candidate applications 89](#_ku6i15xk4abm)

[a. Application List Screen 89](#_qhrewdj2w68d)

[b. Candidate Detail 90](#_jc9jc3xv0iqi)

[5. Profile Management 92](#_w3rervy9kn6h)

[5.1 Manage job seeker profile 92](#_uxlczie2y5ce)

[a. Update information 92](#_bwphjxt5zcvm)

[b. Change password 94](#_4f97udgsd6jq)

[c. Manage CV 96](#_u7w0dz4hl0wm)

[6. Job management 99](#_mnzuied0fq73)

[6.1 View job list 99](#_8r1fr97b565x)

[a. View job list 99](#_3opq03jdmmqt)

[b. View saved job 102](#_sgw4xrckix4t)

[7. Job application 105](#_8a3yptj9aeg1)

[7.1 Apply for job 105](#_rrebcry82oms)

[a. View job detail, apply job 105](#_531w3a4q4r7u)

[b. View applied job 111](#_v8chiyrdb0xd)

[8. Company information 114](#_m1yd5j797mof)

[8.1 View all company 114](#_h48vy5cv1d0p)

[a. View all company 114](#_qlbyay9ponxa)

[8.2 View company detail 119](#_crlkt5yxujjp)

[a. View company detail 119](#_cxf8y0m90jhs)

[9. Authentication 124](#_2xit8przmm3k)

[9.1 Login 124](#_sxb8pflt6r5a)

[a. JobSeeker Login 124](#)

[b. Recruiter Login 125](#)

[9.2 Reset password 126](#_or0e1g51gpao)

[a. Request forget Password Screen 126](#)

[b. Reset Password Screen 128](#_ucw2952bu5bg)

[9.3 Register account 129](#_q4g8x2mcdsql)

[a. JobSeeker Register 129](#)

[b. Recruiter Register 130](#_434faoe5m7y)

[**IV. Appendix 132**](#_a4k2cjcjqmzq)

[1. Assumptions & Dependencies 132](#_3wi203yc93lb)

[2. Limitations & Exclusions 135](#_gef1umj2rtlg)

# I. Overview

## 1. User Requirements

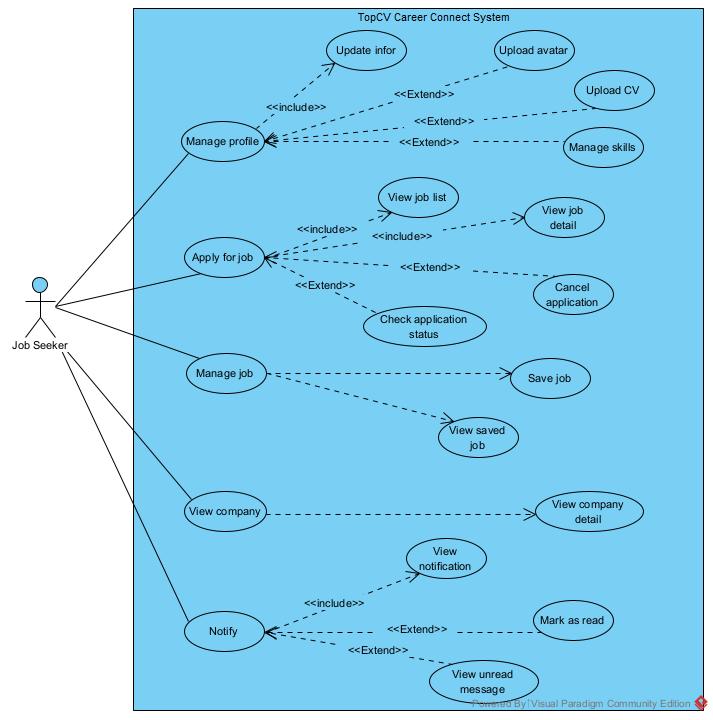
### 1.1 Actors

| **#** | **Actor** | **Description** |
| --- | --- | --- |
| 1 | Admin | General System Management |
| 2 | Job Seeker | User with the purpose of seeking a job |
| 3 | Recruiter | User with the purpose of recruiting |
| 4 | Guest | Unlogged user |

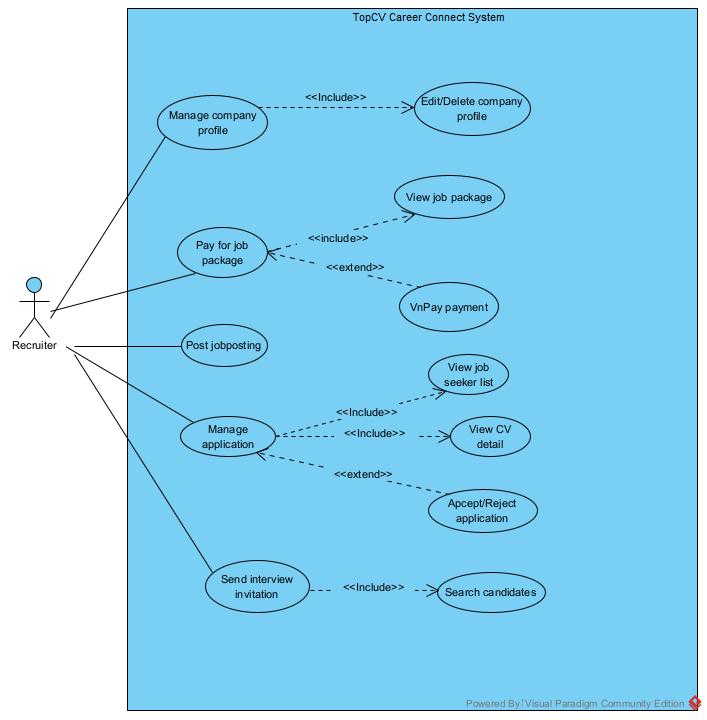
### 1.2 Use Cases

#### *a. Diagram(s)*

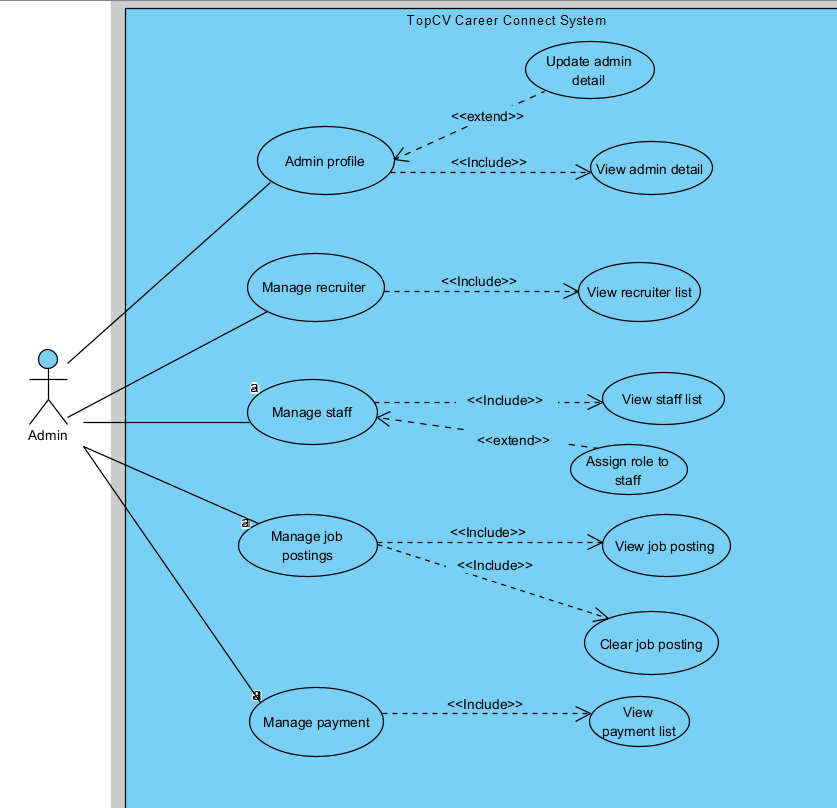
- Job Seeker:



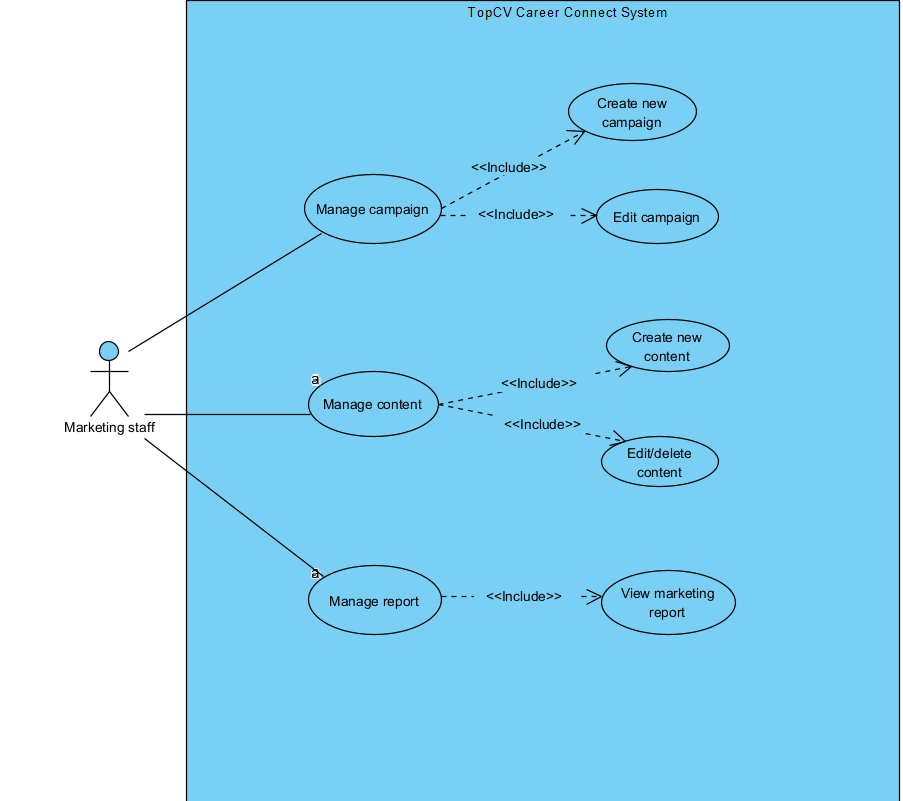
- Recruiter:

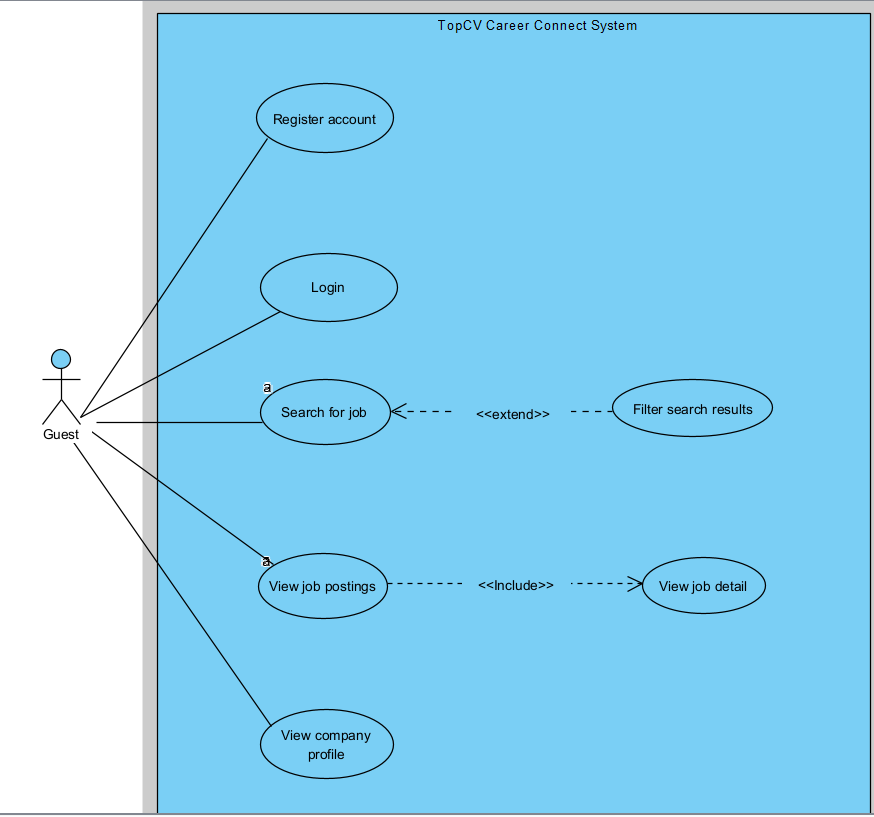


- Admin:



-Marketing Staff:



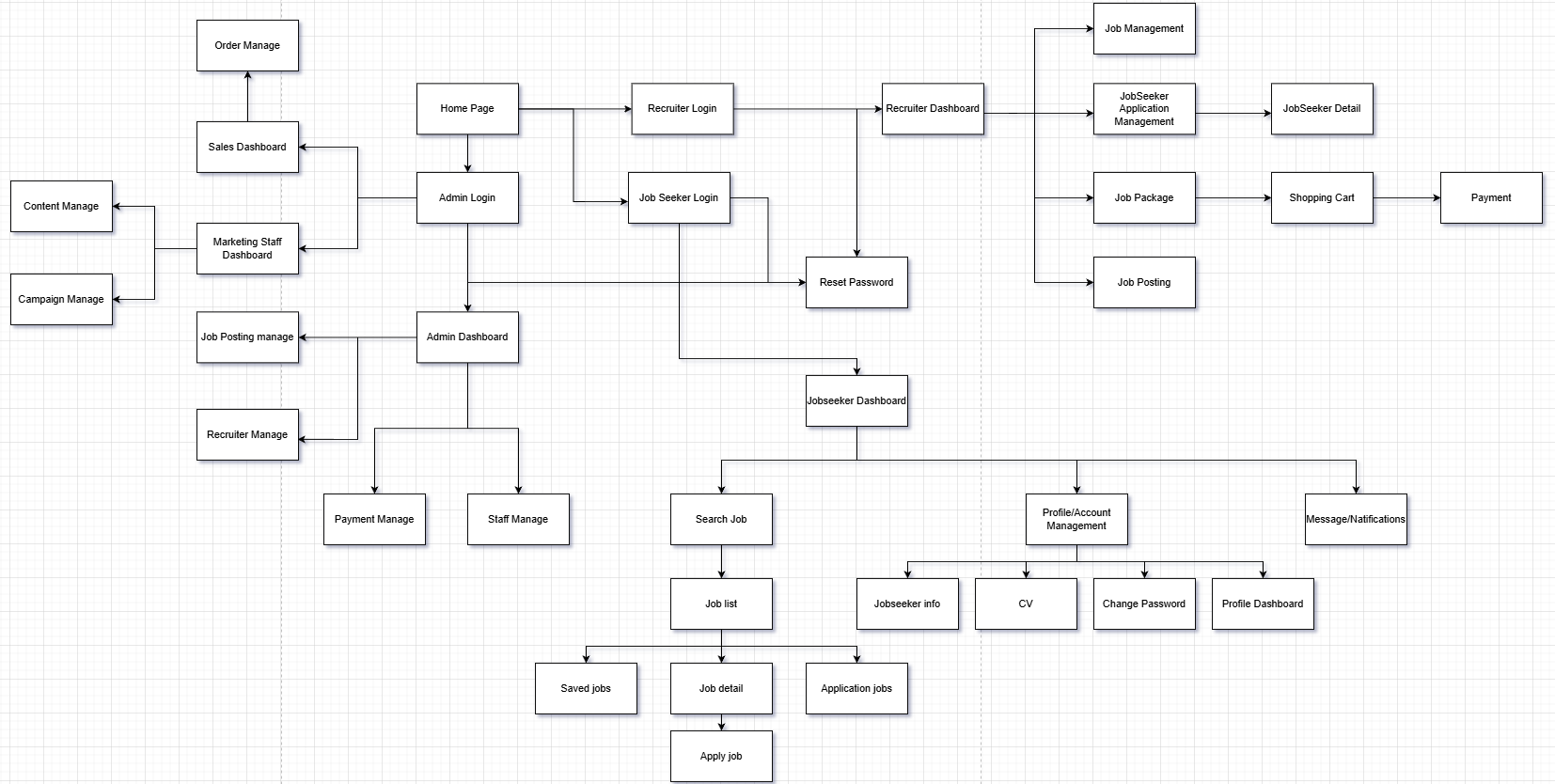
- Guest:

#### *b. Descriptions*

| **ID** | **Feature** | **Use Case** | **Use Case Description** |
| --- | --- | --- | --- |
| UC01 | Company Management | Manage company profile | Recruiters manage company information including updating company information, uploading company photos, logos, etc. |
| UC02 | Job Package Management | Pay for job package | Recruiters purchase posting packages or service packages for themselves and pay through the vnpay service. From there, Recruiters can manage the packages they have purchased. |
| UC03 | Job Posting Management | Post jobposting | Recruiters create jobs and choose the appropriate posting package. From there, Recruiters can manage the created jobs. |
| UC04 | Application Management | Manage application | The recruiter manages users who apply for jobs and can accept or reject them. |
| UC05 | Interview Management | Send interview invitation | The recruiter will send a notification to the applicant's email after accepting or rejecting the application. |
| UC06 | Profile Management | Manage jobseeker profile | Job seekers manage their personal profile information, including updating personal details, uploading a profile picture, uploading a CV, and managing their skills list. |
| UC07 | Job Application | Apply for job | Job seekers view the list of available job postings, view job details, and apply for desired positions. After applying, users can check the status of their applications or cancel them if necessary |
| UC08 | Job Management | Manage job | Job seekers manage interested jobs by saving favorite job postings to a list for later review. The system allows viewing and managing this list of saved jobs. |
| UC09 | Company Information | View company | Job seekers view detailed information about recruiting companies, including company introduction, corporate culture, job locations, and other relevant information to evaluate before applying. |
| UC10 | Notification System | Notify | Job seekers receive and manage notifications from the system regarding application status, interview invitations, and new suitable job postings. Users can view notifications, mark them as read, and view unread messages. |
| UC11 | Admin Management | Admin profile | Administrators can view and update their personal profile information, including name, contact details, and login credentials. They can also change their password and manage security settings to maintain account safety. |
| UC12 | User Management | Manage recruiter | Administrators can view, add, edit, and deactivate recruiter accounts. This includes managing recruiter details, verifying company information, and ensuring compliance with platform policies. |
| UC13 | User Management | Manage staff | Administrators can manage staff accounts by adding new staff, assigning roles and permissions, editing existing profiles, and deactivating inactive users. This ensures proper access control within the system. |
| UC14 | Job Posting Management | Manage job postings | Recruiters can view, update, and ask to delete job postings. They can specify job titles, descriptions, requirements, locations, and salary ranges. The system allows reviewing and approving job listings before publishing. |
| UC15 | Payment Management | Manage payment | Administrators and recruiters can view and manage payment transactions related to job posting services or premium features. Users can track payment history, verify successful payments, and download invoices. |
| UC16 | Marketing Management | Manage Campaign | Marketing staff can create and manage marketing campaigns. This includes defining campaign goals, target audiences, budgets, schedules, and tracking performance metrics to optimize marketing effectiveness. |
| UC17 | Marketing Management | Manage Contents | Marketing staff can create, edit, and delete marketing content such as posts, images, or videos used in campaigns. They can schedule content for publishing on different platforms and monitor engagement. |
| UC18 | Marketing Management | Marketing Report | Marketing staff and administrators can generate and view marketing performance reports. Reports include campaign reach, engagement rate, conversion data, and other key performance indicators to support decision-making. |
| UC19 | Authentication | Register account | After users register for accounts at the registrations corresponding to the roles, the system will identify and redirect users to pages that match their wishes. |
| UC20 | Authentication | Login | After users log in to their accounts at the login pages corresponding to their roles, the system will identify and redirect users to pages that match their wishes. |
| UC21 | Authentication | Reset Password | The users will reset the password if the user forgets their login information. |

## 2. Overall Functionalities

### 2.1 Screens Flow



### 2.2 Screen Descriptions

| **#** | **Feature** | **Screen** | **Description** |
| --- | --- | --- | --- |
| 1 | Job Search & Discovery | Home Page | Users can view all homepage content, search for jobs, and companies |
| 2 | Onboarding & Auth | JobSeeker Login | Users can log in, register, reset passwords, and log in with a Google account |
| 3 | JobSeeker Dashboard | JobSeeker Dashboard | This is a homepage-like screen with full access to JobSeeker features. |
| 4 | Job Search & Discovery | Search Job/Job List | Users can search, filter, and view available job listings. |
| 5 | Job Search & Discovery | Saved Job | Users can view saved jobs. |
| 6 | Job Search & Discovery | Job Detail | Users can view job details, save jobs, upload CV, apply for jobs. |
| 7 | Job Search & Discovery | Apply Job | A sub-screen of Job Detail where users can apply for jobs. |
| 8 | Job Search & Discovery | Application Job | Display applied jobs, application status, filter jobs by application status. |
| 9 | Profile/Account Management | JobSeeker Profile | Manage personal information |
| 10 | Profile/Account Management | CV | Manage CV |
| 11 | Profile/Account Management | Profile Dashboard | Track job search activities. |
| 12 | Profile/Account Management | Change Password | Change Password |
| 13 | Message/Natifications | Natifications | Receive useful notifications. |
| 14 | Onboarding & Auth | Admin Login | Admins can log in, register, reset passwords, and log in with a Google account |
| 15 | Manage Job Posts | Jobposting Management | Allows recruiters or admins to create, update, and delete job postings, as well as view job details. |
| 16 | Manage Recruiters | Recruiter Management | Enables admins to manage recruiter accounts, verify information, and monitor activity. |
| 17 | Manage Payments | Payment Management | Handles transactions, processes payment histories, and manages billing for premium services. |
| 18 | Manage Staff Accounts | Staff Management | Allows administrators to add, edit, or remove staff accounts and assign roles or permissions. |
| 19 | Company Info | Company Info | Recruiters can view and edit their information |
| 20 | JobSeeker Application Management | Recruiter | Recruiters can manage candidates for each job and can reject or accept them. |
| 21 | Job Package | Job Package | Recruiters can purchase and manage purchased packages. |
| 22 | Search JobSeeker | Recruiter | Recruiters can proactively search for candidates for their jobs. |
| 23 | Job Posting | Job Posting | Employers can create job postings. |
| 24 | Job Management | Job Management | Employers manage posted jobs and can edit them. |
|  |  |  |  |

### 2.3 Screen Authorization

| **Screen** | **Guest** | **Admin** | **JobSeeker** | **Recruiter** |
| --- | --- | --- | --- | --- |
| Home page | X | X | X | X |
| JobSeeker Login | X |  | X |  |
| JobSeeker Dashboard |  |  | X |  |
| Search Job/Job List |  |  | X |  |
| Saved Job |  |  | X |  |
| Job Detail |  | X | X |  |
| Apply Job |  |  | X |  |
| Application Job |  |  | X |  |
| JobSeeker Profile |  | X | X | X |
| CV |  |  | X |  |
| Profile Dashboard |  |  | X |  |
| Change Password |  | X | X | X |
| Notifications |  | X | X | X |
| Jobposting Management |  | X |  |  |
| Recruiter Management |  | X |  |  |
| Payment Management |  | X |  |  |
| Staff Management |  | X |  |  |
| Recruiter Login |  |  |  | X |
| Recruiter Dashboard |  |  |  | X |
| Job Package |  |  |  | X |
| Payment |  |  |  | X |
| Post Job |  |  |  | X |
| Company Info |  |  |  | X |
| Management JobSeeker Apply |  |  |  | X |
| JobSeeker Apply Detail |  |  |  | X |
|  |  |  |  |  |
|  |  |  |  |  |

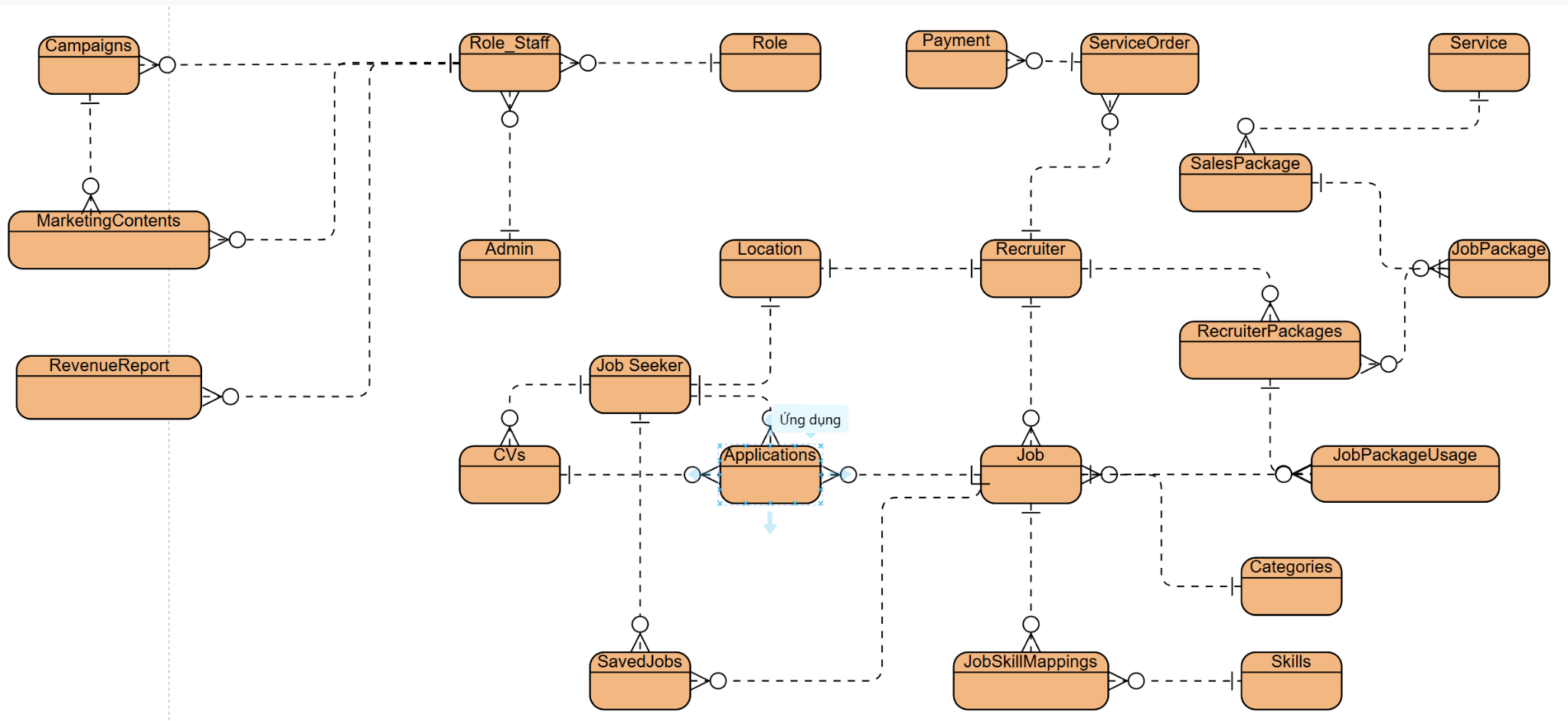
### 2.4 Non-UI Functions

| **#** | **Feature** | **System Function** | **Description** |
| --- | --- | --- | --- |
| 1 | Profile Jobseeker Management | Automatically retrieve FullName from Google when signing in with a Google account. | * "Ensure FullName is never NULL in the database." * "Good user experience: name input is not mandatory during registration." |
| 2 | Profile Jobseeker Management | Automatically generate FullName from Email during regular registration. | * "Ensure FullName is never NULL in the database." * "Good user experience: name input is not mandatory during registration." |
| 2 | Authentication | Automatically hash MD5 for password. | Password is always hashed with MD5 before being stored in the database. |
| 3 | File Management | Automatically generate a unique file name upon upload | Avoid duplicate file names, path traversal attack |
| 4 | Notification System | Auto-notification | Automatically generate notifications |
| 5 | Security | Ownership verification | Automatic security check to ensure users can only access their own data |
| 6 | Authentication | Prevent Google account from changing password. | Users who sign in with Google cannot change password through the system (password managed by Google) |
| 7 | Authentication | Prevent Google login if the email is already registered with a password. | Avoid account conflicts - if email already exists with password-based registration, Google login is blocked |
| 8 | Application Management | Automatically prevent duplicate applications/job saves. | Ensure users cannot apply to the same job multiple times or save the same job multiple times |
| 9 | Application Management | Only allow cancellation of applications with Status = "Pending". | Prevent users from cancelling applications that are already "Approved", "Rejected"to maintain data integrity |
| 10 | Application Management | Prioritize newer CVs over older CVs when applying. | If a user uploads a new CV and applies for a job, the system automatically uses the latest CV instead of older versions |
| 11 | File Upload Validation | Validate real images, not just extension. | Check actual file content (magic bytes/MIME type) to ensure uploaded files are genuine images, preventing malicious files with fake extensions |
| 12 | Session Management | Auto-refresh session after update | Automatically refresh user session data after profile updates to reflect changes immediately without requiring re-login |
| 13 | Email(Google) | Auto send email | If the user resets the password, the employer accepts or rejects the candidate, the system will automatically send the appropriate email. |
| 13 | Password hash | Auto hashing password |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3. System High Level Design

### 3.1 Database Design

#### *a. Database Schema*



#### *b. Table Descriptions*

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | Campaigns | Stores marketing campaign information including target audience (JobSeeker/Recruiter), budget, timeline, and campaign status |
| 02 | MarketingContents | Stores marketing content for each campaign, including title, content text, media URL, and publishing platform |
| 03 | Role\_Staff | Junction table for staff permission management, linking Roles with Admins |
| 04 | Role | Defines system roles (Admin, Marketing Staff, HR, Sales) |
| 05 | Admin | Stores administrator and staff information including personal details and account status |
| 06 | Location | List of locations/cities (Hanoi, Ho Chi Minh City, Da Nang...) |
| 07 | Job Seeker | Job seeker information: email, password, personal profile, skills, and account status |
| 08 | Revenue  Report | Revenue reports by time period, tracking total revenue, transaction count, and revenue sources |
| 09 | CVs | Stores JobSeeker's CVs/resumes including title, content, file URL, and active status |
| 10 | Applications | Manages job applications, linking Jobs with CVs and tracking application status |
| 11 | SavedJobs | Stores JobSeeker's favorite/bookmarked jobs |
| 12 | Recruiter | Recruiter/company information: email, company details, tax code, and business registration |
| 13 | Job | Job postings with detailed information: title, requirements, salary range, timeline, status, and priority features |
| 14 | JobSkillMappings | Junction table linking required skills to each job posting |
| 15 | Skills | List of skills (Java, Python, JavaScript, Design...) |
| 16 | Categories | Industry categories (IT, Business, Healthcare...) with hierarchical structure |
| 17 | JobPackageUsage | Tracks Recruiter's package usage including posted jobs count and remaining time |
| 18 | RecruiterPackages | Links Recruiters with purchased JobPackages, storing start/end dates and job quantity |
| 19 | JobPackage | Service packages for job posting (Bronze, Silver, Gold, Diamond) with pricing and features |
| 20 | Payment | Recruiter payment information: amount, payment method, status, and transaction code |
| 21 | ServiceOrder | Service orders from Recruiters, linked with Payments |
| 22 | Service | List of additional services (job refresh, priority listing, recruitment support...) |
| 23 | SalesPackage | Special packages managed and sold by Sales team to customers |

### 3.2 Code Packages



***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | controller.base | Contains base servlet classes or shared controller logic used across multiple modules |
| 02 | controller.admin | Handles admin-related requests such as managing users, staff, and job postings. |
| 03 | controller.staff | Manages operations performed by staff users |
| 04 | controller.jobseeker | Handles requests from job seekers such as viewing jobs, applying for positions, managing CV/profile, and tracking application status |
| 05 | controller.recruiter | Handles requests from recruiters such as posting job openings, managing job listings, reviewing candidate applications, and managing company profiles |
| 06 | controller | Contains all servlet controllers that handle HTTP requests and responses for the entire application |
| 07 | service | Implements business logic and service layer operations that process data between controllers and DAL |
| 08 | filter | Contains servlet filters for request/response processing such as authentication, authorization, and logging |
| 09 | dal | Data Access Layer - Handles all database operations and queries for interacting with the database |
| 10 | model | Defines entity classes and data models representing database tables and business objects |
| 11 | util | Provides utility classes and helper functions for common tasks like validation, formatting, and email sending |

# II. Requirement Specifications

## 1.

### 1.1 UC01\_Manage company profile

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC01\_Manage company profile | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 15/10/2025 |
| Primary Actor: | Recruiter | Secondary Actors: | None |
| Trigger: | Recruiter accesses the company profile management section or clicks on "Edit Company Profile" button | | |
| Description: | This use case allows recruiters to manage their company's profile information, including updating company details (name, description, location, industry, size), uploading company photos, logos, and other relevant media to maintain an accurate and attractive company presence on the platform. | | |
| Preconditions: | PRE-1: Recruiter must be logged in and authenticated. PRE-2: Recruiter must have appropriate permissions to manage company profile. PRE-3: Company profile must exist in the system. | | |
| Postconditions: | POST-1: Company profile information is updated in the database. POST-2: Changes are reflected on the company's public profile. POST-3: System logs the update with timestamp and user information. POST-4: Uploaded images (logo, photos) are stored and associated with the company profile. | | |
| Normal Flow: | 1.1 Manage company profile   1. Employer navigates to company profile management page 2. System displays company profile template with editable fields (company name, description, industry, website, phone number, etc.) 3. System displays current company logo and image 4. Employer enters/updates company information (name, description, address, contact information) 5. Employer enters company description and mission statement 6. Employer selects industry from drop-down list 7. Employer enters company website URL and social media links 8. Employer enters phone number and contact email 9. Employer clicks "Upload logo" button 10. System opens file dialog to select logo 11. Employer selects logo image file from device 12. System validates file format (PNG, JPG, GIF) and size (max 5MB) 13. System displays preview of selected logo 14. Employer clicks "Upload company photo" button 15. System opens file dialog to select multiple images 16. Employer selects company photo/image (allows multiple files to be selected) 17. System validates format and size of each image file 18. System displays preview thumbnails of selected images 19. Employer reviews all information and Preview 20. Employer clicks "Save Company Profile" button 21. System confirms all required fields are filled in 22. System uploads logo file to cache (CreateFile/SaveFile operation) 23. System uploads company photo to cache 24. System saves logo file path to database (RecruiterDAO.updateCompanyLogo()) 25. System saves company photo file path to database (RecruiterDAO.updateCompanyPhotos()) 26. System updates company information to database (RecruiterDAO.updateCompanyProfile()) 27. System generates thumbnails to optimize display 28. System records successful update to audit log with timestamp and ID Employer 29. The system displays a success message: "Company profile has been updated successfully" 30. The system redirects to the updated company profile view page 31. Employers can view their company profile with all updated information and images | | |
| Alternative Flows: | 1.1 - Update Company Profile Without Images:  Branches at step 9 in normal flow  Recruiter skips logo and photo uploads  Recruiter only updates text information (company name, description, contact, etc.)  Proceed to step 19 of normal flow  Images remain unchanged from previous upload  1.2 - Replace Existing Company Logo:   1. Branches at step 9 in normal flow 2. System displays current logo with "Replace" option 3. Recruiter clicks "Replace Logo" button 4. System opens file dialog 5. Recruiter selects new logo file 6. System deletes old logo file from storage 7. System saves new logo file path in database 8. Rejoin at step 19 of normal flow   1.3 - Delete Company Images:   1. Branches at step 14 in normal flow 2. Recruiter views existing company photos 3. Recruiter selects images to delete from gallery 4. System displays confirmation dialog: "Delete selected images?" 5. Recruiter confirms deletion 6. System removes selected image files from storage 7. System removes image file paths from database 8. Rejoin at step 19 of normal flow   1.4 - Update Company Logo and Photos Only:   1. Branches at step 4 in normal flow 2. Recruiter skips text information update 3. Recruiter only uploads/updates logo and photos 4. Proceed to step 9 of normal flow 5. Text information remains unchanged   1.5 - Reorder Company Photos:   1. Branches at step 18 in normal flow 2. System displays uploaded photos with drag-and-drop interface 3. Recruiter drags photos to reorder them 4. System updates photo display order in database 5. Rejoin at step 19 of normal flow | | |
| Exceptions: | 1.0.E1 - Required Field Missing:   1. Occurs at step 21 in normal flow 2. One or more required fields (company name, description, contact info) are empty 3. System displays error: "Please fill in all required fields" 4. System highlights empty fields in red 5. User returns focus to missing field 6. User can enter missing information and retry   1.0.E2 - Invalid File Format:   1. Occurs at step 12 or 17 in normal flow 2. User selects file with unsupported format (not PNG, JPG, GIF, etc.) 3. System displays error: "Invalid file format. Only PNG, JPG, and GIF are allowed" 4. System returns to file dialog 5. User can select different file   1.0.E3 - File Size Exceeds Limit:   1. Occurs at step 12 or 17 in normal flow 2. Selected file exceeds maximum size limit (5MB for logo, 10MB for photos) 3. System displays error: "File size exceeds limit (Max 5MB for logo, 10MB for photos)" 4. System returns to file dialog 5. User can compress image and select again   1.0.E4 - File Upload Failure:   1. Occurs at step 22 or 23 in normal flow 2. File storage service is temporarily unavailable 3. System displays error: "Unable to upload file. Please try again" 4. System rolls back database changes 5. User can retry upload after waiting   1.0.E5 - Database Update Failure:   1. Occurs at step 26 in normal flow 2. Database connection lost during save operation 3. System displays error: "Failed to save company profile. Please try again" 4. System rolls back all changes (files and database) 5. Uploaded files are deleted from storage 6. User can retry after database recovers   1.0.E6 - Invalid URL Format:   1. Occurs at step 7 in normal flow 2. User enters invalid website or social media URL 3. System displays error: "Invalid URL format. Please check website URL" 4. User corrects URL and continues   1.0.E7 - Invalid Phone Number Format:   1. Occurs at step 8 in normal flow 2. User enters phone number in invalid format 3. System displays error: "Please enter valid phone number (10-15 digits)" 4. User corrects phone number   1.0.E8 - Session Timeout During Upload:   1. Occurs at step 22-23 in normal flow 2. User session expires while file is uploading 3. System displays error: "Your session has expired. Please log in again" 4. Uploaded files are cleaned up from storage 5. User must re-authenticate and retry   1.0.E9 - Duplicate Company Name:   1. Occurs at step 26 in normal flow 2. Company name already exists in database (if unique constraint) 3. System displays error: "Company name already exists. Please use different name" 4. User can modify company name and retry   1.0.E10 - Corrupted Image File:   1. Occurs at step 12 or 17 in normal flow 2. Selected image file is corrupted or unreadable 3. System displays error: "Image file is corrupted. Please select different file" 4. User can select different image file | | |
| Priority: | Must Have | | |
| Frequency of Use: | Estimated 5-10 times per week per company (varies by company activity and profile updates needed) | | |
| Business Rules: | FR1, FR2, FR3, FR4 | | |
| Other Information: | **Quality Attributes:** - Performance: Profile updates should complete within 3 seconds. - Usability: Interface should be intuitive with clear labeling and validation feedback. - Security: All uploads should be scanned for malware. **Related Requirements:** - FR-101: System must support image compression for uploaded photos. - FR-102: System must maintain version history of company profile changes. **Failure Handling:** - If network connection is lost during upload, system should allow resume or retry of upload. | | |
| Assumptions: | ASM-1: Recruiter has stable internet connection for uploading media files. ASM-2: Company profile already exists and is associated with the recruiter's account. ASM-3: Browser supports HTML5 file upload functionality. ASM-4: Uploaded content complies with platform's terms of service and content policy. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| FR1 | Required Company Fields | Company name, email, and contact information,.. are mandatory. Profile cannot be saved without these fields. |
| FR2 | File Size Limit | Logo maximum 1MB, company photos maximum 3MB each, maximum 3 photos per company. |
| FR3 | Supported Image Formats | Only PNG, JPG, JPEG, GIF formats are allowed for uploads. Other formats must be rejected. |
| FR4 | Profile Access Control | Only the recruiter who owns the company or system admin can edit company profile |

### 1.2 UC03\_Post jobposting

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC03\_Pay for Job Package | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 15/10/2025 |
| Primary Actor: | Recruiter | Secondary Actors: | None |
| Trigger: | Recruiter initiates the process by selecting "Create New Job Posting" from the recruitment dashboard.. | | |
| Description: | This use case allows recruiters to create new job postings with detailed job information, select an appropriate posting package (based on visibility, duration, and features), and subsequently manage (edit, update status, close, or delete) the created job postings. | | |
| Preconditions: | - PRE-1: Recruiter has successfully authenticated and logged into the system  - PRE-2: Recruiter has an active account with proper permissions to create job postings  - PRE-3: At least one posting package option is available in the system  - PRE-4: Recruiter's organization has sufficient credits or active subscription (if applicable) | | |
| Postconditions: | - POST-1: Job posting is successfully created and stored in the database  - POST-2: Selected posting package is applied to the job posting  - POST-3: Job posting is published and visible to job applicants based on package settings  - POST-4: Recruiter has access to manage the created job posting (edit, update status, close, delete) | | |
| Normal Flow: | 1.2 Pay for Job Package   1. Recruiter accesses the recruitment dashboard 2. Recruiter clicks "Create New Job Posting" button 3. System displays job posting creation form 4. Recruiter enters job details (title, description, requirements, salary, location, job type) 5. System validates input in real-time 6. Recruiter selects job category and experience level 7. Recruiter clicks "Next" to proceed to package selection 8. System displays available posting packages (Basic, Standard, Premium) 9. Recruiter reviews package options with features and pricing 10. Recruiter selects a posting package 11. System displays posting preview 12. Recruiter reviews and clicks "Confirm & Publish" 13. System processes payment/credits and publishes the posting 14. System displays confirmation message with posting ID 15. Recruiter is redirected to job posting management dashboard 16. System displays the new posting in active jobs list | | |
| Alternative Flows: | 1.2.1 - Save as Draft:   1. At step 4 in Normal Flow, Recruiter clicks "Save as Draft" instead of "Next" 2. System saves all entered information with draft status 3. System returns to dashboard displaying the draft posting 4. Rejoin: Recruiter can later select "Edit Draft" to continue from step 7   1.2.2 - Edit Existing Job Posting:   1. Recruiter selects an existing published job posting from the management list 2. System displays job posting details in edit mode 3. Recruiter modifies desired information 4. Recruiter clicks "Update Posting" 5. System validates and updates the posting 6. System displays confirmation message 7. Rejoin: Return to dashboard   1.2.3 - Delete Company Images:   1. Recruiter selects an active job posting from the dashboard 2. Recruiter clicks "Close Posting" button 3. System prompts for confirmation 4. Recruiter confirms the action 5. System updates posting status to "Closed" 6. Posting becomes invisible to applicants   1.4 - Update Company Logo and Photos Only:   1. Branches at step 4 in normal flow 2. Recruiter skips text information update 3. Recruiter only uploads/updates logo and photos 4. Proceed to step 9 of normal flow 5. Text information remains unchanged   1.5 - Reorder Company Photos:   1. Branches at step 18 in normal flow 2. System displays uploaded photos with drag-and-drop interface 3. Recruiter drags photos to reorder them 4. System updates photo display order in database 5. Rejoin at step 19 of normal flow | | |
| Exceptions: | 1.0.E1 - Required Field Missing:   1. Occurs at step 21 in normal flow 2. One or more required fields (company name, description, contact info) are empty 3. System displays error: "Please fill in all required fields" 4. System highlights empty fields in red 5. User returns focus to missing field 6. User can enter missing information and retry   1.0.E2 - Invalid File Format:   1. Occurs at step 12 or 17 in normal flow 2. User selects file with unsupported format (not PNG, JPG, GIF, etc.) 3. System displays error: "Invalid file format. Only PNG, JPG, and GIF are allowed" 4. System returns to file dialog 5. User can select different file   1.0.E3 - File Size Exceeds Limit:   1. Occurs at step 12 or 17 in normal flow 2. Selected file exceeds maximum size limit (5MB for logo, 10MB for photos) 3. System displays error: "File size exceeds limit (Max 5MB for logo, 10MB for photos)" 4. System returns to file dialog 5. User can compress image and select again   1.0.E4 - File Upload Failure:   1. Occurs at step 22 or 23 in normal flow 2. File storage service is temporarily unavailable 3. System displays error: "Unable to upload file. Please try again" 4. System rolls back database changes 5. User can retry upload after waiting   1.0.E5 - Database Update Failure:   1. Occurs at step 26 in normal flow 2. Database connection lost during save operation 3. System displays error: "Failed to save company profile. Please try again" 4. System rolls back all changes (files and database) 5. Uploaded files are deleted from storage 6. User can retry after database recovers   1.0.E6 - Invalid URL Format:   1. Occurs at step 7 in normal flow 2. User enters invalid website or social media URL 3. System displays error: "Invalid URL format. Please check website URL" 4. User corrects URL and continues   1.0.E7 - Invalid Phone Number Format:   1. Occurs at step 8 in normal flow 2. User enters phone number in invalid format 3. System displays error: "Please enter valid phone number (10-15 digits)" 4. User corrects phone number   1.0.E8 - Session Timeout During Upload:   1. Occurs at step 22-23 in normal flow 2. User session expires while file is uploading 3. System displays error: "Your session has expired. Please log in again" 4. Uploaded files are cleaned up from storage 5. User must re-authenticate and retry   1.0.E9 - Duplicate Company Name:   1. Occurs at step 26 in normal flow 2. Company name already exists in database (if unique constraint) 3. System displays error: "Company name already exists. Please use different name" 4. User can modify company name and retry   1.0.E10 - Corrupted Image File:   1. Occurs at step 12 or 17 in normal flow 2. Selected image file is corrupted or unreadable 3. System displays error: "Image file is corrupted. Please select different file" 4. User can select different image file | | |
| Priority: | Must Have | | |
| Frequency of Use: | Estimated 5-10 times per week per company (varies by company activity and profile updates needed) | | |
| Business Rules: | FR1, FR2, FR3, FR4 | | |
| Other Information: | **Quality Attributes:** - Performance: Profile updates should complete within 3 seconds. - Usability: Interface should be intuitive with clear labeling and validation feedback. - Security: All uploads should be scanned for malware. **Related Requirements:** - FR-101: System must support image compression for uploaded photos. - FR-102: System must maintain version history of company profile changes. **Failure Handling:** - If network connection is lost during upload, system should allow resume or retry of upload. | | |
| Assumptions: | ASM-1: Recruiter has stable internet connection for uploading media files. ASM-2: Company profile already exists and is associated with the recruiter's account. ASM-3: Browser supports HTML5 file upload functionality. ASM-4: Uploaded content complies with platform's terms of service and content policy. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| FR1 | Required Company Fields | Company name, email, and contact information,.. are mandatory. Profile cannot be saved without these fields. |
| FR2 | File Size Limit | Logo maximum 1MB, company photos maximum 3MB each, maximum 3 photos per company. |
| FR3 | Supported Image Formats | Only PNG, JPG, JPEG, GIF formats are allowed for uploads. Other formats must be rejected. |
| FR4 | Profile Access Control | Only the recruiter who owns the company or system admin can edit company profile |

### 

### 1.3 UC04\_Manage application

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC04\_Manage application | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 15/10/2025 |
| Primary Actor: | Recruiter | Secondary Actors: | None |
| Trigger: | Job seeker submits an application for a job posting, or Recruiter accesses the application management dashboard | | |
| Description: | Recruiters manage job applications received from job seekers. They can view application details, accept or reject applications, send messages to applicants, and track application status throughout the hiring process. | | |
| Preconditions: | PRE-1: Recruiter account has been created and authenticated in the system.  PRE-2: Recruiter has at least one active job posting.  PRE-3: Job seeker has submitted an application for the recruiter's job posting.  PRE-4: System database is accessible and functional. | | |
| Postconditions: | POST-1: Application status is updated (Accepted, Rejected, Pending, In Review).  POST-2: Job seeker receives notification of application status change.  POST-3: Recruiter can view comprehensive list of all applications for their job postings.  POST-4: Application history is recorded in the database for future reference. | | |
| Normal Flow: | 1.3 Manage application   1. Recruiter accesses the application management dashboard 2. System displays list of all job postings with application counts 3. Recruiter selects a job posting to view applications 4. System displays all applications received for that job posting (sorted by date, status, or relevance) 5. 5. Recruiter clicks on an application to view detailed information 6. System displays applicant profile, CV/resume, cover letter, and application submission date 7. Recruiter reviews the applicant's qualifications and details 8. Recruiter clicks "Accept" or "Reject" button 9. System displays confirmation dialog 10. Recruiter confirms the decision 11. System updates application status in database 12. System sends notification email to job seeker with the decision 13. System displays confirmation message to recruiter 14. Application status is updated in the list view 15. Recruiter can continue reviewing other applications | | |
| Alternative Flows: | ***1.3.1 No Job Postings Available***   1. Steps 1–2 of Normal Flow. 2. System detects no job postings exist. 3. System displays message: *“No job postings found. Please create a job posting first.”* 4. Use case ends.   ***1.3.2 Job Posting Has No Applications***   1. Recruiter selects a job posting (Normal Flow Step 3). 2. System finds zero applications. 3. System displays message: *“There are no applications for this job posting.”* 4. Recruiter returns to dashboard.   ***1.3.3*** ***Recruiter searches based on criteria***   1. In step 4 of Normal Flow, the recruiter conducts the search 2. The system updates the list based on the selection. 3. The use case returns to step 5 of Normal Flow.   ***1.3.4 The Employer Cancels the Accept/Reject Action***   1. At Step 9 of Normal Flow. 2. The Employer Clicks Cancel in the Confirmation Dialog. 3. The System Does Not Change the Profile Status. 4. The Use Case Returns to Step 7 of Normal Flow.   ***1.3.5 Error loading candidate profile (CV/Profile)***   1. At step 5–6 of Normal Flow. 2. The system cannot load the CV or candidate profile due to file error, server error, etc. 3. The system displays an error message: “Cannot load candidate profile. Please try again later.” 4. The employer can try again or return to the profile list. | | |
| Exceptions: | ***1.3.1 Error loading candidate profile (CV/Profile)***   1. At step 10 of Normal Flow. 2. The system encountered an error while updating the status to the database. 3. The system displays an error message: “Failed to update profile status. Please try again.” 4. The employer chooses to try again or return to the list.   ***1.3.2 Error sending notification email***   1. The profile status is updated successfully. 2. The system fails to send the email. 3. The system displays a warning: “The status has been updated but the notification email could not be sent. Please try again later.” 4. The use case continues at step 13 of Normal Flow. | | |
| Priority: | Must Have | | |
| Frequency of Use: | Estimated from 20–100 times per week depending on the number of job postings and the number of applicants submitted by the company. | | |
| Business Rules: | BR1-BR6 | | |
| Other Information: | **Quality Attributes:**   * The system must display the list of profiles within < 2 seconds. * The profile browsing interface must be easy to use, clearly labeled, and have operation confirmation. * All CV/Resume files displayed must be scanned for malware by the system beforehand.   **Related Requirements:**   * FR-131: The system supports filtering and sorting of applications by submission date, status, and suitability. * FR-132: The system saves the application review history (audit log). * FR-133: The system sends email notifications to candidates when the status changes.   **Failure Handling:**   * If the network connection is lost during the status update process → the system does not change the data and asks the user to try again. * If there is a database error → the profile status is not updated and the system displays an error message. * If there is an email sending error → the profile status is still updated, but the system displays a warning and allows the email to be sent again later. | | |
| Assumptions: | * ASM-1: The employer has a stable internet connection when reviewing the application. * ASM-2: The candidate's application exists in the system and is correctly linked to the job posting. * ASM-3: The user's browser supports displaying PDF, DOCX files or has the ability to download CVs. * ASM-4: The candidate has provided complete data (CV, personal information, application letter) when submitting the application. * ASM-5: The system's email server is working properly when sending the notification. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR1 | Valid profile status | Each application profile is only allowed to have the following status values: Pending, Accepted, Rejected. |
| BR2 | Employer Permissions | Employers are only allowed to view and approve profiles of job postings that they manage or are authorized to. |
| BR3 | Save Action History | When the employer performs Accept/Reject, the system must save information to the Audit Log (operator, time, action). |
| BR4 | Do not change the status without confirmation | The system only updates the profile status after the employer clicks Confirm in the confirmation dialog box. |
| BR5 | Send mandatory email notification | Each time the profile status changes, the system must send an email to notify the candidate of the result (unless the email server is faulty). |
| BR6 | Only process 1 action at a time | Employers cannot Accept and Reject the same profile in the same session; only one action is recorded. |

### 1.4 UC05\_Send Interview Invitation

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC04\_Send Interview Invitation | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 15/10/2025 |
| Primary Actor: | Recruiter | Secondary Actors: | None |
| Trigger: | Recruiter accepts an application and chooses to send an interview invitation | | |
| Description: | When the recruiter accepts/rejects the application, the system sends a notification/email to the candidate. | | |
| Preconditions: | PRE-1: Recruiter is logged in and authenticated.  PRE-2: Candidate has applied to a recruiter's job posting.  PRE-3: System can access NotificationDAO / CandidateEmailService. | | |
| Postconditions: | POST-1: Application status is updated (Accepted or Rejected).  POST-2: Notification record is created in DB (NotificationDAO).  POST-3: Interview invitation/notification email is sent to candidate via CandidateEmailService (if sent successfully). | | |
| Normal Flow: | 1.4 Send Interview Invitation   1. Recruiter selects the application and clicks "Accept" (or "Reject") and confirms in the confirmation dialog. 2. System updates the application status in ApplicationDAO. 3. System creates a notification record in NotificationDAO (type: interview\_invitation). 4. System calls CandidateEmailService to send an interview invitation email to the candidate's email address. 5. System displays a confirmation to the recruiter about the status and the notification sending status. | | |
| Alternative Flows: | ***1.4.1 Recruiter cancels Accept/Reject action***   1. In step 3 of Normal Flow, instead of confirming, the recruiter clicks Cancel. 2. The system closes the dialog box, without updating the status. 3. Returns to the profile details screen.   ***1.4.2 Interview invitation email not sent (email optional)***   1. In step 7 of Normal Flow, the system fails to send email due to an error in the mail sending service. 2. The system still keeps the notification record in NotificationDAO. 3. The system displays a warning: “The status has been updated but the email cannot be sent. Please try again later.” 4. The use case ends normally. | | |
| Exceptions: | ***1.4.1 Status update error in ApplicationDAO***   1. At step 5 of Normal Flow, the system fails to update the status (DB error). 2. The system displays an error message: “Unable to update status. Please try again.” 3. Notification record not created. 4. Email not sent. 5. Profile status remains unchanged (rollback). 6. The recruiter returns to the previous screen.   ***1.4.2 NotificationDAO Creation Error***   1. At step 6 of Normal Flow, the system fails to create a notification record in NotificationDAO. 2. The system rolls back the status (no Accept/Reject is recorded). 3. The system reports an error: “Cannot create notification. Status not updated.” 4. No email sent. 5. The employer is returned to the profile details screen.   ***1.4.3 CandidateEmailService call error (email service completely disconnected)***   1. At step 7 of Normal Flow, email service fatal error (timeout, unreachable). 2. The profile status has been updated and no rollback (because the error occurred after step 6). 3. NotificationDAO still has the record. 4. The system reports an error: “Unable to send email. Please try again in Notification.” 5. Use case continues at step 8. | | |
| Priority: | Must Have | | |
| Frequency of Use: | Estimated from 20–100 times per week depending on the number of job postings and the number of applicants. | | |
| Business Rules: | BR1-BR6 | | |
| Other Information: | **Quality Attributes:**   * The system must display the list of profiles within < 2 seconds. * The application review interface must be easy to use, clearly labeled, and require confirmation before critical actions. * All resume files displayed must be scanned for malware beforehand.   **Related Requirements:**   * **FR-131:** The system supports filtering and sorting of applications by submission date, status, and suitability. * **FR-132:** The system saves the application review history (audit log). * **FR-133:** The system sends email notifications to candidates when the application status changes.     **Failure Handling:**   * If the network connection is lost during the status update → the system does not update the data and asks the user to try again. * If there is a database error → the profile status is not updated and the system displays an error message. * If there is an email sending error → the profile status is still updated, but the system displays a warning and allows the email to be resent later. | | |
| Assumptions: | * **ASM-1:** The employer has a stable internet connection during the review process. * **ASM-2:** The candidate’s application exists in the system and is correctly linked to the job posting. * **ASM-3:** The user’s browser supports viewing PDF, DOCX, or equivalent file types. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR1 | Authorized Recruiter Requirement | Only users with the *Recruiter* role (or higher permission) are allowed to update the application status (Accept/Reject). |
| BR2 | Audit Log Recording | Every status update must be recorded in the audit log, including recruiter ID, timestamp, old status, and new status. |
| BR3 | Valid Status Transition | Application status can only transition according to defined rules (e.g., cannot change from *Rejected* to *Accepted*). Completed applications cannot be updated unless unlocked by an admin. |
| BR4 | Notification Creation Requirement | A notification record must be created in NotificationDAO for every Accept/Reject action before the process is considered successful. |
| BR5 | Email Template Compliance | Sent emails must follow the company’s standard templates: interview invitation for Accept and application closure for Reject. Templates must include candidate name, job position, and recruiter contact. |
| BR6 | Atomic Transaction Rule | Status update, notification creation, and email-sending operation must behave as a single logical transaction. If any step fails (before completion), the system rolls back all steps and keeps the original status. |

### 1.6 UC06\_Manage jobseeker profile

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC06\_Manage jobseeker profile | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 15/10/2025 |
| Primary Actor: | Jobseeker | Secondary Actors: | Recruiter |
| Trigger: | Job seeker clicks on the profile icon or navigates to the profile page from the navigation menu to view or update their personal information. | | |
| Description: | This use case allows authenticated job seekers to manage their personal profile information comprehensively. Job seekers can update personal details (full name, phone, gender, address, headline, contact information), upload and manage their profile picture (avatar), upload and manage their CV documents, and add or remove professional skills from their profile. The system provides real-time validation, progress tracking, and feedback to ensure data quality and completeness. | | |
| Preconditions: | - PRE-1: User must be authenticated and logged into the system  - PRE-2: User must have a valid job seeker role/account  - PRE-3: User session must be active and valid  - PRE-4: Database connection must be available  - PRE-5: File upload directory must be accessible and have sufficient storage space | | |
| Postconditions: | - POST-1: Job seeker profile information is successfully updated in the database  - POST-2: Uploaded avatar image is stored in the server file system with the correct filename format  - POST-3: Uploaded CV files are stored securely with unique identifiers  - POST-4: Skills added or removed are properly reflected in the JobSeeker\_Skill junction table  - POST-5: Profile completion percentage is recalculated and displayed  - POST-6: User session is updated with the latest profile information  - POST-7: Success confirmation message is displayed to the user  . | | |
| Normal Flow: | 6.0.1 View Profile  1. Job seeker navigates to the profile page by clicking on their profile icon or menu  2. System validates the user session and authentication status  3. System retrieves complete profile information from database including:  - Personal details (name, email, phone, gender, address, headline)  - Current avatar image  - List of uploaded CVs  - List of associated skills  - Profile completion statistics  - Recent activity (applied jobs, saved jobs)  4. System calculates profile completion percentage based on:  - Avatar presence (1/7)  - CV upload (1/7)  - Phone number (1/7)  - Address (1/7)  - Headline (1/7)  - Experience level (1/7)  - Gender (1/7)  5. System displays profile dashboard with all information organized in sections  6. Job seeker views their current profile information  6.0.2 Update Personal Information  1. Job seeker clicks the "Edit Profile" button (pencil icon)  2. System opens a modal dialog with editable fields pre-filled with current data  3. Job seeker modifies one or more of the following fields:  - Full Name (required, max 100 characters)  - Email (read-only display)  - Phone (optional, numeric validation)  - Gender (dropdown: Male/Female/Other)  - Headline (optional, max 255 characters - professional title/summary)  - Address (optional, max 255 characters)  - Contact Information (optional, max 255 characters)  - Location (dropdown selection)  - Current Level/Experience (dropdown selection)  - Status (active/inactive)  4. Job seeker clicks "Save Changes" button  5. System validates all input fields:  - Full Name: not empty, length ≤ 100 characters  - Phone: numeric format (if provided)  - Headline: length ≤ 255 characters  - Contact Info: length ≤ 255 characters  - Address: length ≤ 255 characters  6. System sends POST request to `/jobseekerprofile` servlet  7. System updates JobSeeker table in database with new values  8. System updates session with new profile data  9. System displays success message "Profile updated successfully"  10. System refreshes the profile page with updated information  11. System updates profile completion percentage if applicable  6.0.3 Upload Profile Picture (Avatar)  1. Job seeker hovers over their avatar area  2. System displays camera icon overlay  3. Job seeker clicks on the avatar area  4. System displays avatar menu with options:  - "Upload New Photo"  - "Delete Photo" (if avatar exists)  5. Job seeker selects "Upload New Photo"  6. System opens file selection dialog  7. Job seeker selects an image file from their device  8. System validates the selected file:  - File type: JPEG, PNG, or JPG only  - File size: maximum 5MB  - Image dimensions: recommended minimum 200x200 pixels  9. System displays image preview in a confirmation modal  10. Job seeker confirms the upload  11. System generates unique filename using format: `{jobSeekerId}\_{timestamp}.{extension}`  12. System uploads file to `/avatar/` directory on server  13. System updates `Img` field in JobSeeker table with the new filename  14. System displays success message "Avatar updated successfully"  15. System refreshes profile page showing the new avatar  16. System updates profile completion percentage  6.0.4 Upload CV Document  1. Job seeker scrolls to "Profile Documents" section  2. Job seeker clicks on "Choose or drag and drop CV from your device" area or drags a file  3. System highlights the upload area  4. Job seeker selects a CV file (or drops file in drag zone)  5. System validates the file:  - File type: PDF, DOC, DOCX only  - File size: maximum 10MB  - Filename: no special characters  6. System displays selected filename in the upload area  7. Job seeker clicks "Upload" button  8. System generates unique CV identifier (UUID format)  9. System uploads file to `/uploads/cvs/` directory with UUID filename  10. System creates new record in CV table:  - JobSeekerID (foreign key)  - CVFileURL (UUID filename)  - CVTitle (optional, extracted from filename)  - Status (default: 'active')  - CreatedAt (current timestamp)  11. System displays success message "CV uploaded successfully"  12. System adds the CV to the uploaded CVs list on profile page  13. System updates profile completion percentage  14. System shows file management options (view, download, delete) for the new CV  6.0.5 Manage Skills  1. Job seeker clicks "Add Skill" button in the Skills section  2. System opens Skills Management modal  3. System displays:  - Search bar for filtering skills  - List of all available skills from database  - Currently selected skills (checked)  4. Job seeker searches for skills using keywords (optional)  5. System filters and displays matching skills in real-time  6. Job seeker selects/deselects skills by clicking checkboxes  7. System maintains selected skills in temporary collection  8. Job seeker clicks "Save Skills" button  9. System sends POST request to `/manage-skills` servlet with action="add"  10. For each newly selected skill:  - System inserts record into JobSeeker\_Skill junction table  - System sets SkillID and JobSeekerID  11. For each deselected skill:  - System deletes record from JobSeeker\_Skill junction table  12. System displays success message "Skills updated successfully"  13. System closes the modal  14. System refreshes Skills section showing updated skill badges  15. Each skill is displayed as a colored badge with remove (×) option  6.0.6 Remove Skill  1. Job seeker clicks the "×" icon on a skill badge  2. System displays confirmation prompt "Remove this skill?"  3. Job seeker confirms removal  4. System sends POST request to `/manage-skills` servlet with action="remove"  5. System deletes the corresponding record from JobSeeker\_Skill table  6. System displays success message "Skill removed successfully"  7. System removes the skill badge from the display  8. Use case continues at step 6.0.5 if job seeker wants to add more skills | | |
| Alternative Flows: | 6.1 Update Personal Information with Partial Data  Branches from: Step 3 of flow 6.0.2  Condition: Job seeker leaves optional fields empty  1. Job seeker leaves one or more optional fields blank (phone, address, headline, contact info)  2. Job seeker clicks "Save Changes"  3. System validates required fields only (Full Name)  4. System updates database with provided values and NULL for empty optional fields  5. System displays success message with note "Profile updated. Complete remaining fields to increase profile visibility"  6. System shows reduced profile completion percentage  7. Rejoins normal flow at: Step 10 of flow 6.0.2  6.2 Upload Avatar via Drag and Drop  Branches from: Step 3 of flow 6.0.3  Condition: Job seeker prefers drag and drop method  1. Job seeker drags image file over avatar area  2. System detects drag event and highlights the drop zone  3. Job seeker drops the file  4. System captures the dropped file  5. \*\*Rejoins normal flow at:\*\* Step 8 of flow 6.0.3 (file validation)  6.3 Delete Existing Avatar  Branches from: Step 5 of flow 6.0.3  Condition: Job seeker wants to remove current avatar  1. Job seeker selects "Delete Photo" from avatar menu  2. System displays confirmation dialog "Are you sure you want to delete your profile picture?"  3. Job seeker confirms deletion  4. System sends DELETE request to `/delete-avatar` endpoint  5. System deletes the avatar file from `/avatar/` directory  6. System updates `Img` field in JobSeeker table to NULL  7. System displays success message "Avatar deleted successfully"  8. System shows default avatar icon (user silhouette)  9. System decreases profile completion percentage  10. \*\*Use case ends successfully\*\*  6.4 Replace Existing CV  Branches from: Step 4 of flow 6.0.4  Condition: Job seeker already has uploaded CVs  1. System displays existing CVs in the documents list  2. Job seeker can view, download, or delete existing CVs  3. Job seeker uploads a new CV file  4. Continues with normal flow at: Step 5 of flow 6.0.4  5. System adds new CV to the list without removing existing ones  6. Job seeker can manage multiple CV versions  6.5 Delete Uploaded CV  Branches from: Step 2 of flow 6.0.4  Condition: Job seeker wants to remove an uploaded CV  1. Job seeker clicks "Delete" button (trash icon) on a CV entry  2. System displays confirmation dialog "Are you sure you want to delete this CV?"  3. Job seeker confirms deletion  4. System sends DELETE request with CV ID  5. System deletes the CV file from `/uploads/cvs/` directory  6. System updates CV record status to 'deleted' or removes the record  7. System displays success message "CV deleted successfully"  8. System removes CV from the display list  9. System updates profile completion percentage if no CVs remain  10. \*\*Use case ends successfully\*\*  6.6 Quick Skill Search  Branches from: Step 4 of flow 6.0.5  Condition: Job seeker uses search functionality  1. Job seeker types keywords in the skill search box  2. System filters available skills in real-time using JavaScript  3. System displays only matching skills (case-insensitive partial match)  4. Job seeker sees reduced list of relevant skills  5. \*\*Rejoins normal flow at:\*\* Step 6 of flow 6.0.5  6.7 View Profile Statistics  Branches from:Step 6 of flow 6.0.1  Condition: Job seeker wants to see detailed activity  1. Job seeker scrolls to Activity Overview section  2. System displays statistics:  - Number of job applications submitted  - Number of saved jobs  - Profile views count (if implemented)  - Last profile update timestamp  3. Job seeker reviews their activity metrics  4. Job seeker can click on "View Applied Jobs" or "View Saved Jobs"  5. System navigates to respective detailed pages  6. Use case ends | | |
| Exceptions: | 6.0.E1 Session Expired or Unauthorized Access  Occurs at: Any step in any flow  Condition: User session is invalid or user is not authenticated  1. System detects invalid session or missing authentication  2. System displays error message "Your session has expired. Please log in again."  3. System redirects user to login page with return URL parameter  4. System stores intended action in session for post-login redirect  5. \*\*Use case ends with failure\*\*  6. \*\*Recovery:\*\* User logs in again and is redirected back to profile page  6.0.E2 Database Connection Failure  \*\*Occurs at:\*\* Steps involving database operations (read/write)  \*\*Condition:\*\* Database is unavailable or connection timeout  1. System attempts database operation  2. System catches SQLException or connection timeout  3. System logs error details to server log  4. System displays user-friendly error message "Service temporarily unavailable. Please try again later."  5. System rolls back any partial transactions  6. \*\*Use case ends with failure\*\*  7. \*\*Recovery:\*\* System administrator investigates and restores database connection  6.2.E1 Invalid Personal Information Format  \*\*Occurs at:\*\* Step 5 of flow 6.0.2  \*\*Condition:\*\* Validation fails for one or more fields  1. System detects validation error:  - Full Name empty or exceeds 100 characters  - Phone contains non-numeric characters  - Headline exceeds 255 characters  - Contact Info exceeds 255 characters  - Address exceeds 255 characters  2. System prevents form submission  3. System highlights invalid field(s) with red border  4. System displays specific error message below each invalid field:  - "Full Name is required and must be less than 100 characters"  - "Phone number must contain only digits"  - "Headline must be less than 255 characters"  - etc.  5. System keeps modal open with user's input preserved  6. \*\*Continues at:\*\* Step 3 of flow 6.0.2 (user corrects the errors)  6.3.E1 Invalid Avatar File Type  \*\*Occurs at:\*\* Step 8 of flow 6.0.3  \*\*Condition:\*\* Selected file is not an accepted image format  1. System validates file extension and MIME type  2. System detects invalid format (not JPEG, PNG, or JPG)  3. System displays error message "Please upload a valid image file (JPEG, PNG, or JPG only)"  4. System clears file selection  5. System keeps file dialog available  6. \*\*Continues at:\*\* Step 7 of flow 6.0.3 (user selects different file)  6.3.E2 Avatar File Size Exceeds Limit  \*\*Occurs at:\*\* Step 8 of flow 6.0.3  \*\*Condition:\*\* Selected image file is larger than 5MB  1. System checks file size  2. System detects file size > 5MB  3. System displays error message "Image file size must not exceed 5MB. Please compress or select a smaller image."  4. System rejects the upload  5. System suggests image optimization tools or alternative  6. \*\*Continues at:\*\* Step 7 of flow 6.0.3  6.3.E3 Avatar Upload Server Error  \*\*Occurs at:\*\* Step 12 of flow 6.0.3  \*\*Condition:\*\* File system error during avatar upload  1. System attempts to write file to `/avatar/` directory  2. System encounters IOException (disk full, permission denied, etc.)  3. System logs detailed error to server log  4. System displays error message "Failed to upload avatar. Please try again or contact support."  5. System rolls back database update (keeps old avatar filename)  6. \*\*Use case ends with failure\*\*  7. \*\*Recovery:\*\* User retries upload; admin checks server storage and permissions  6.4.E1 Invalid CV File Type  \*\*Occurs at:\*\* Step 5 of flow 6.0.4  \*\*Condition:\*\* Selected file is not a valid document format  1. System validates file extension and MIME type  2. System detects invalid format (not PDF, DOC, or DOCX)  3. System displays error message "Please upload a valid CV file (PDF, DOC, or DOCX only)"  4. System clears file selection  5. System highlights upload area with error state  6. \*\*Continues at:\*\* Step 4 of flow 6.0.4  6.4.E2 CV File Size Exceeds Limit  \*\*Occurs at:\*\* Step 5 of flow 6.0.4  \*\*Condition:\*\* Selected CV file is larger than 10MB  1. System checks file size  2. System detects file size > 10MB  3. System displays error message "CV file size must not exceed 10MB. Please compress your document or remove large images."  4. System rejects the upload  5. \*\*Continues at:\*\* Step 4 of flow 6.0.4  6.4.E3 CV Storage Server Error  \*\*Occurs at:\*\* Step 9 of flow 6.0.4  \*\*Condition:\*\* File system error during CV upload  1. System attempts to write file to `/uploads/cvs/` directory  2. System encounters IOException or storage error  3. System logs error details  4. System displays error message "Failed to upload CV. Storage error occurred. Please try again."  5. System does not create CV record in database  6. \*\*Use case ends with failure\*\*  7. \*\*Recovery:\*\* User retries upload; admin checks server storage capacity  6.5.E1 Skill Not Found in Database  \*\*Occurs at:\*\* Step 10 of flow 6.0.5  \*\*Condition:\*\* Selected skill ID doesn't exist in Skill table  1. System attempts to insert record in JobSeeker\_Skill table  2. System encounters foreign key constraint violation  3. System logs error with skill ID details  4. System displays error message "Selected skill is no longer available. Please refresh and try again."  5. System rolls back the transaction  6. \*\*Use case ends with failure\*\*  7. \*\*Recovery:\*\* System refreshes skill list; user selects valid skill  6.5.E2 Duplicate Skill Entry  \*\*Occurs at:\*\* Step 10 of flow 6.0.5  \*\*Condition:\*\* Job seeker already has the skill in their profile  1. System attempts to insert duplicate record  2. System detects unique constraint violation  3. System ignores the duplicate insertion (treats as success)  4. System continues processing remaining skills  5. System displays success message without mentioning the duplicate  6. \*\*Continues at:\*\* Step 12 of flow 6.0.5  6.6.E1 Skill Removal Fails  \*\*Occurs at:\*\* Step 5 of flow 6.0.6  \*\*Condition:\*\* Database error during skill removal  1. System attempts to delete record from JobSeeker\_Skill table  2. System encounters SQLException  3. System logs error details  4. System displays error message "Failed to remove skill. Please try again."  5. System keeps skill badge visible  6. \*\*Use case ends with failure\*\*  7. \*\*Recovery:\*\* User retries removal operation  6.0.E3 Network Connection Lost  \*\*Occurs at:\*\* Any step involving AJAX request  \*\*Condition:\*\* Client loses internet connection during operation  1. System attempts to send AJAX request  2. Request times out or returns network error  3. JavaScript catches the error  4. System displays error message "Network connection lost. Please check your internet connection and try again."  5. System preserves user's input data in browser memory  6. \*\*Use case is suspended\*\*  7. \*\*Recovery:\*\* User restores internet connection and retries the operation  6.0.E4 Concurrent Update Conflict  \*\*Occurs at:\*\* Step 7 of flow 6.0.2  \*\*Condition:\*\* Profile was modified by another session/device simultaneously  1. System attempts to update database with new values  2. System detects record version mismatch (if using optimistic locking)  3. System displays warning message "Your profile was updated from another device. Please refresh and try again."  4. System shows "Refresh" button  5. Job seeker clicks Refresh  6. System reloads latest profile data  7. \*\*Continues at:\*\* Step 1 of flow 6.0.2 | | |
| Priority: | High | | |
| Frequency of Use: | - Profile View: 15-30 times per user per month (average)  - Profile Update: 3-5 times per user per month  - Avatar Upload: 1-2 times per user lifecycle  - CV Upload: 2-4 times per user per month (updates, versions)  - Skills Management: 2-3 times per user per month  - Estimated Daily Active Users: 1,000-5,000 job seekers  - Peak Concurrent Users: 200-500 during business hours (9 AM - 6 PM) | | |
| Business Rules: | BR-001, BR-002, BR-003, BR-004, BR-005, BR-006, BR-007, BR-008, BR-009,BR-010 | | |
| Other Information: | - Profile page must load within 2 seconds under normal load  - Avatar upload must complete within 5 seconds for 5MB files  - CV upload must complete within 10 seconds for 10MB files  - Skills update must respond within 1 second  - Database queries must be optimized with proper indexing  - File operations must be asynchronous to prevent UI blocking | | |
| Assumptions: | 1. Job seekers have basic computer literacy and understand how to upload files  2. Job seekers have access to their CV in electronic format (PDF, DOC, DOCX)  3. Job seekers have a stable internet connection for file uploads  4. Avatar images are appropriate and professional (manual moderation may be required)  5. Server has sufficient storage capacity for uploaded files (monitored by admin)  6. Email addresses are verified during registration (prerequisite)  7. Job seekers understand the importance of profile completeness for job matching  8. Browser supports JavaScript and modern web APIs (FileReader, FormData, fetch)  9. File system permissions are correctly configured for upload directories  10. Database backup and recovery procedures are in place  11. Skills taxonomy is pre-populated and regularly updated by administrators  12. Users consent to data processing as per privacy policy during registration | | |

#### b. Business Rules

| ID | Business Rule | Business Rule Description |
| --- | --- | --- |
| BR-001 | Update profile | Job seekers must complete at least 50% of their profile before applying for jobs |
| BR-002 | Avatar | Avatar images must be in professional format (JPEG, PNG) and appropriate content |
| BR-003 | CV | CV files must not contain malicious code or scripts |
| BR-004 | Job | Each job seeker can upload maximum 5 CV versions |
| BR-005 | Skill | Skill list must be maintained from a predefined taxonomy (admin-managed) |
| BR-006 | Profile completion | Profile completion percentage affects search ranking and recruiter visibility |
| BR-007 | Email | Email address cannot be changed after registration (security measure) |
| BR-008 | Inactive profiles | Inactive profiles (no login for 365 days) may be archived |
| BR-009 | Data | Personal data must comply with GDPR/privacy regulations |
| BR-010 | File | File storage must be secure and access-controlled |

### 1.7 UC07\_Apply for job

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC07\_Apply for job | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 15/10/2025 |
| Primary Actor: | Jobseeker | Secondary Actors: | System, Recruiter |
| Trigger: | Job seeker browses available job postings and clicks "Apply Now" button on a job listing or job detail page. | | |
| Description: | This use case enables authenticated job seekers to view available job postings, browse job details including requirements and company information, submit applications with selected CV, track application status, and cancel pending applications when needed. The system validates eligibility, prevents duplicate applications, stores application records, sends notifications to both job seekers and recruiters, and maintains application history with status tracking. | | |
| Preconditions: | PRE-1: User must be authenticated and logged in as job seeker  PRE-2: User session must be active and valid  PRE-3: Job seeker must have at least one uploaded CV in their profile  PRE-4: Selected job must be in "Published" status  PRE-5: Job posting must not be expired or closed  PRE-6: Database connection must be available | | |
| Postconditions: | POST-1: New application record created in Applications table with status "Pending"  POST-2: Application date timestamp recorded  POST-3: Selected CV linked to the application  POST-4: Notification sent to recruiter about new application  POST-5: Notification sent to job seeker confirming application submission  POST-6: Application appears in job seeker's "Applied Jobs" list  POST-7: Application counter for the job is incremented  POST-8: Job seeker cannot apply to the same job again until current application is resolved | | |
| Normal Flow: | 7.0.1 Browse Job Listings  1. Job seeker navigates to job listing page or homepage  2. System retrieves all published jobs from database  3. System displays job cards with key information:  - Job title  - Company name and logo  - Location  - Salary range  - Job type (Full-time, Part-time, Contract, Internship)  - Posted date  - Number of applicants  4. Job seeker browses through available jobs  5. System provides filtering options:  - Location (dropdown)  - Job type (checkbox)  - Salary range (slider)  - Category/Industry  - Experience level  6. System provides search functionality by keywords  7. Job seeker applies filters or searches for specific jobs  8. System updates job listing based on criteria  9. Job seeker views filtered results  7.0.2 View Job Details  1. Job seeker clicks on a job card or "View Details" button  2. System retrieves complete job information from Jobs table joined with Recruiter, Location, Type, Category tables  3. System checks if job seeker has already applied for this job  4. System displays comprehensive job detail page:  - Job title and company name  - Company logo and industry  - Location and job type  - Salary range  - Posted date and expiration date  - Number of applicants  - Job description (full text)  - Job requirements (skills, experience, education)  - Benefits and perks  - Application deadline  - Contact information  - Company profile section  - Similar jobs recommendations  5. System displays "Apply Now" button if not already applied  6. System displays "Already Applied" badge if application exists  7. Job seeker reads job details thoroughly  8. Job seeker clicks "Apply Now" button  7.0.3 Submit Job Application  1. System validates job seeker authentication  2. System checks if job seeker has already applied for this job  3. System opens application modal dialog  4. System displays application form with:  - Job title (read-only)  - Company name (read-only)  - CV selection dropdown (list of uploaded CVs)  - Cover letter text area (optional)  - Terms and conditions checkbox  5. System pre-selects most recently uploaded CV  6. Job seeker reviews job information  7. Job seeker selects CV from dropdown (or keeps pre-selected)  8. Job seeker optionally enters cover letter or additional message  9. Job seeker checks terms and conditions checkbox  10. Job seeker clicks "Submit Application" button  11. System validates form data:  - CV selection is not empty  - Terms checkbox is checked  - Cover letter length within limit (if provided)  12. System sends POST request to /job-application servlet  13. System performs eligibility checks:  - Job seeker is authenticated  - Job ID is valid and job exists  - CV ID is valid and belongs to job seeker  - Job seeker has not already applied for this job  - Job is still published and accepting applications  14. System creates new Application record:  - JobID (foreign key)  - CVID (foreign key)  - ApplicationDate (current timestamp)  - Status (default: "Pending")  15. System inserts record into Applications table  16. System sends notification to recruiter:  - Type: "application"  - Message: "New application received for [Job Title]"  - Link to application detail page  17. System sends confirmation notification to job seeker:  - Type: "application"  - Message: "Application submitted successfully for [Job Title]"  - Link to applied jobs page  18. System displays success message "Application submitted successfully"  19. System closes application modal  20. System refreshes page showing "Already Applied" status  21. System redirects to "Applied Jobs" page or stays on job detail page  7.0.4 View Applied Jobs List  1. Job seeker clicks "Applied Jobs" in navigation menu  2. System sends request to /applied-jobs endpoint  3. System validates job seeker session  4. System retrieves all applications for current job seeker from database using query joining:  - Applications table  - CVs table  - Jobs table  - Recruiter table  - Locations table  - Types table  - Categories table  5. System orders applications by ApplicationDate descending (newest first)  6. System displays applied jobs list with each entry showing:  - Job title (clickable link to job detail)  - Company name  - Location  - Application date (formatted)  - Application status badge (Pending/Accepted/Rejected)  - CV used for application  - Action buttons based on status  7. System provides filtering options:  - By status (All, Pending, Accepted, Rejected)  - By date range (Last 7 days, Last 30 days, Last 90 days, All time)  8. System provides search functionality to find specific applications  9. Job seeker views application history and status  10. System displays application statistics:  - Total applications submitted  - Pending applications count  - Accepted applications count  - Rejected applications count  7.0.5 Check Application Status  1. Job seeker opens "Applied Jobs" page  2. System displays current status for each application  3. Job seeker views status badge:  - "Pending" - yellow badge, clock icon  - "Accepted" - green badge, check icon  - "Rejected" - red badge, X icon  4. System shows status change timestamp (if status was updated)  5. Job seeker clicks "View Details" button on an application  6. System navigates to job detail page  7. System displays application status history (if available)  8. Job seeker reviews application details and status  7.0.6 Cancel Pending Application  1. Job seeker is on "Applied Jobs" page  2. System displays "Cancel Application" button only for applications with "Pending" status  3. Job seeker clicks "Cancel Application" button for a pending application  4. System opens confirmation modal  5. System displays warning message: "Are you sure you want to cancel application for [Job Title]? This action cannot be undone!"  6. Job seeker confirms cancellation  7. System sends POST request to /cancel-application servlet with applicationId  8. System validates:  - Job seeker is authenticated  - Application ID is valid  - Application belongs to current job seeker (security check)  - Application status is "Pending" (only pending applications can be cancelled)  9. System deletes application record from Applications table  10. System sends notification to job seeker:  - Message: "Application cancelled successfully. You can reapply anytime."  11. System displays success message "Application cancelled successfully"  12. System removes cancelled application from the list  13. System refreshes applied jobs page  14. Use case ends successfully | | |
| Alternative Flows: | 7.1 Quick Apply from Job Card  Branches from: Step 4 of flow 7.0.1  Condition: Job seeker clicks "Quick Apply" directly from job listing card  1. Job seeker clicks "Quick Apply" button on job card  2. System opens simplified application modal  3. System pre-fills with most recent CV  4. Job seeker confirms application without viewing full details  5. Rejoins normal flow at: Step 11 of flow 7.0.3  7.2 Save Job for Later  Branches from: Step 7 of flow 7.0.2  Condition: Job seeker wants to save job without applying immediately  1. Job seeker clicks "Save Job" button (heart icon)  2. System validates authentication  3. System adds job to SavedJobs table  4. System displays confirmation "Job saved successfully"  5. System changes button to "Saved" with filled heart icon  6. Job seeker can access saved jobs from "Saved Jobs" page  7. Use case ends successfully  7.3 Apply with New CV Upload  Branches from: Step 7 of flow 7.0.3  Condition: Job seeker wants to upload new CV during application  1. Job seeker clicks "Upload New CV" link in application modal  2. System opens file selection dialog  3. Job seeker selects CV file (PDF, DOC, DOCX)  4. System validates file (type, size under 10MB)  5. System uploads CV to server  6. System creates CV record in database  7. System adds new CV to dropdown selection  8. System automatically selects newly uploaded CV  9. Rejoins normal flow at: Step 9 of flow 7.0.3  7.4 View Similar Jobs  Branches from: Step 5 of flow 7.0.2  Condition: Job seeker wants to see related job opportunities  1. Job seeker scrolls to "Similar Jobs" section  2. System displays jobs matching:  - Same category/industry  - Same location  - Same experience level  - Similar salary range  3. Job seeker clicks on a similar job  4. System navigates to that job's detail page  5. Rejoins normal flow at: Step 2 of flow 7.0.2  7.5 Filter Applied Jobs by Status  Branches from: Step 7 of flow 7.0.4  Condition: Job seeker wants to filter applications by specific status  1. Job seeker selects status filter (Pending/Accepted/Rejected)  2. System filters applications list matching selected status  3. System updates displayed count  4. Job seeker views filtered results  5. Job seeker can reset filter to view all applications  6. Rejoins normal flow at: Step 9 of flow 7.0.4  7.6 View Application from Notification  Branches from: Job seeker receives notification about application status change  Condition: Recruiter updates application status  1. Job seeker receives notification "Application status updated"  2. Job seeker clicks notification  3. System navigates to specific application detail  4. System highlights status change  5. Job seeker views updated status  6. Rejoins normal flow at: Step 8 of flow 7.0.5  7.7 Reapply After Cancellation  Branches from: Step 13 of flow 7.0.6  Condition: Job seeker decides to reapply after cancelling  1. Job seeker navigates back to job detail page  2. System shows "Apply Now" button again (cancellation removed restriction)  3. Job seeker clicks "Apply Now"  4. Rejoins normal flow at: Step 1 of flow 7.0.3 | | |
| Exceptions: | .0.E1 Session Expired During Application  Occurs at: Any step requiring authentication  Condition: User session expires or becomes invalid  1. System detects invalid or expired session  2. System displays error "Session expired. Please log in again."  3. System stores application form data in browser localStorage  4. System redirects to login page with return URL  5. Job seeker logs in  6. System restores application form data  7. System redirects back to application modal  8. Continues at: Step 9 of flow 7.0.3  7.0.E2 Database Connection Failure  Occurs at: Any database operation step  Condition: Database is unavailable  1. System attempts database query  2. System catches SQLException or timeout  3. System logs error details  4. System displays error "Service temporarily unavailable. Please try again."  5. Use case ends with failure  6. Recovery: Admin restores database connection; user retries operation  7.2.E1 No CV Available  Occurs at: Step 4 of flow 7.0.3  Condition: Job seeker has not uploaded any CV  1. System checks job seeker's CV list  2. System finds no CV records  3. System displays message "Please upload your CV before applying"  4. System provides "Upload CV" button  5. Job seeker clicks "Upload CV"  6. System navigates to profile page CV section  7. Job seeker uploads CV  8. Continues at: Step 1 of flow 7.0.3  7.3.E1 Duplicate Application Attempt  Occurs at: Step 13 of flow 7.0.3  Condition: Job seeker already applied for this job  1. System checks Applications table for existing record  2. System finds matching JobID and JobSeekerID  3. System prevents duplicate insertion  4. System displays error "You have already applied for this position"  5. System shows link to "View Your Application"  6. Job seeker clicks link  7. System navigates to Applied Jobs page with application highlighted  8. Use case ends  7.3.E2 Job No Longer Available  Occurs at: Step 13 of flow 7.0.3  Condition: Job was closed or deleted during application process  1. System validates Job status  2. System finds job is not "Published" or does not exist  3. System displays error "This job is no longer accepting applications"  4. System closes application modal  5. System refreshes job detail page showing "No longer available"  6. Use case ends with failure  7.3.E3 Application Submission Server Error  Occurs at: Step 15 of flow 7.0.3  Condition: Database error during application record insertion  1. System attempts to insert Application record  2. System encounters SQLException  3. System rolls back transaction  4. System logs error details  5. System displays error "Failed to submit application. Please try again."  6. System keeps application modal open with data preserved  7. Job seeker retries submission  8. Continues at: Step 10 of flow 7.0.3  7.3.E4 Invalid CV Selection  Occurs at: Step 11 of flow 7.0.3  Condition: Selected CV does not belong to job seeker or was deleted  1. System validates CV ownership  2. System finds CV does not belong to current job seeker  3. System displays error "Invalid CV selection"  4. System refreshes CV dropdown  5. System prompts job seeker to select valid CV  6. Continues at: Step 7 of flow 7.0.3  7.6.E1 Cannot Cancel Non-Pending Application  Occurs at: Step 8 of flow 7.0.6  Condition: Application status is not "Pending"  1. System validates application status  2. System finds status is "Accepted" or "Rejected"  3. System displays error "Only pending applications can be cancelled"  4. System closes confirmation modal  5. System shows appropriate message:  - If Accepted: "Application already accepted by employer"  - If Rejected: "Application already reviewed and closed"  6. Use case ends  7.6.E2 Application Cancellation Fails  Occurs at: Step 9 of flow 7.0.6  Condition: Database error during deletion  1. System attempts to delete Application record  2. System encounters SQLException  3. System logs error details  4. System displays error "Failed to cancel application. Please try again."  5. System keeps application visible in list  6. Use case ends with failure  7. Recovery: Job seeker retries cancellation  7.0.E3 Network Connection Lost  Occurs at: Any AJAX/fetch request  Condition: Client loses internet connectivity  1. System attempts to send request  2. Request times out or returns network error  3. JavaScript catches error  4. System displays error "Network connection lost. Please check your connection."  5. System preserves user input data  6. Use case is suspended  7. Recovery: User restores connection and retries  7.4.E1 No Jobs Found with Filters  Occurs at: Step 8 of flow 7.0.1  Condition: No jobs match selected filter criteria  1. System executes filtered query  2. System returns empty result set  3. System displays "No jobs found matching your criteria"  4. System suggests:  - Clear some filters  - Try different keywords  - View all jobs  5. Job seeker adjusts filters or clears them  6. Continues at: Step 7 of flow 7.0.1 | | |
| Priority: | High | | |
| Frequency of Use: | Profile View: 10-20 times per user per week (browsing sessions)  Job Detail View: 5-15 times per user per week  Application Submission: 2-5 times per user per week  Applied Jobs Check: 3-7 times per user per week  Application Cancellation: 0.1-0.5 times per user per week  Estimated Daily Applications: 500-2000 across all users  Peak Hours: 9-11 AM, 1-3 PM, 7-9 PM (lunch breaks and after work) | | |
| Business Rules: | BR-011 - BR-021 | | |
| Other Information: | Job listing page must load within 2 seconds with 50+ jobs  Job detail page must load within 1.5 seconds  Application submission must complete within 3 seconds  Applied jobs list must load within 2 seconds  Search and filter results must update within 1 second  Database queries must use proper indexes on JobID, JobSeekerID, Status, ApplicationDate  Application modal must open instantly (under 300ms) | | |
| Assumptions: | ASSUM-13: Job seekers understand they can only apply once per job  ASSUM-14: Job seekers have prepared CV ready for upload  ASSUM-15: Recruiters regularly check and update application statuses  ASSUM-16: Job postings have accurate and complete information  ASSUM-17: Network connectivity is generally stable for most users  ASSUM-18: Users understand status terminology (Pending, Accepted, Rejected)  ASSUM-19: Job seekers check their applications regularly for updates  ASSUM-20: Email notifications reach users' inbox (not spam)  ASSUM-21: Job expiration dates are properly maintained  ASSUM-22: System has sufficient capacity during peak application hours  ASSUM-23: Users have valid email addresses for notifications  ASSUM-24: Job seekers consent to data sharing with recruiters | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR-011 | Job Application Status | Job seekers can only apply to published jobs with active status. |
| BR-012 | Duplicate Applications | One job seeker can apply to the same job only once (no duplicate applications). |
| BR-013 | CV Requirement for Application | Job seeker must have at least one CV uploaded to apply. |
| BR-014 | Application Status Flow | Applications start with "Pending" status, can be updated to "Accepted" or "Rejected" by recruiter. |
| BR-015 | Application Cancellation | Only "Pending" applications can be cancelled by job seeker. |
| BR-016 | Application Date Logging | Application date is recorded with a timestamp for tracking. |
| BR-017 | Event Notification | Notifications are sent to both recruiter and job seeker on application events. |
| BR-018 | Cancelled Application Handling | Cancelled applications are permanently deleted (not soft-deleted). |
| BR-019 | Reapplication | Job seekers can reapply after cancelling a previous application. |
| BR-020 | Reporting Data Retention | Application history is retained for reporting even if the job is deleted |
| BR-021 | Expired Jobs | Expired jobs do not accept new applications. |
| BR-021 | Application Count Display | Application count is displayed to job seekers for popularity indication. |

### 1.8 UC08\_Manage job

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC08\_Manage job | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 16/10/2025 |
| Primary Actor: | Jobseeker | Secondary Actors: | None |
| Trigger: | Job seeker clicks the heart/save icon on a job listing or job detail page to save the job for later review. | | |
| Description: | This use case allows authenticated job seekers to save interesting job postings to a favorites list for later review. Job seekers can save jobs from listing pages or job detail pages, view their complete list of saved jobs, check saved job status, and remove jobs from the saved list when no longer interested. The system maintains the saved jobs list persistently and provides quick access for job seekers to track opportunities they want to apply for later. | | |
| Preconditions: | PRE-1: User must be authenticated and logged in as job seeker  PRE-2: User session must be active and valid  PRE-3: Job must exist in the database with valid JobID  PRE-4: Database connection must be available | | |
| Postconditions: | POST-1: Job is added to or removed from SavedJobs table  POST-2: Save status is updated in real-time on UI  POST-3: Saved jobs count is updated  POST-4: Job seeker can access saved job from saved jobs list  POST-5: Save action is recorded with timestamp | | |
| Normal Flow: | 8.0.1 Save Job from Listing  1. Job seeker browses job listings or searches for jobs  2. System displays job cards with save button (heart icon)  3. System checks if each job is already saved for current user  4. System displays appropriate icon state (empty heart = not saved, filled heart = saved)  5. Job seeker clicks save button (heart icon) on a job card  6. System validates user authentication  7. System sends POST request to /saved-jobs servlet with action=save and jobId  8. System checks if job is already saved in SavedJobs table  9. System inserts new record into SavedJobs table with JobSeekerID and JobID  10. System records SavedDate with current timestamp  11. System returns success response with message "Job saved successfully"  12. System updates button UI to filled heart icon  13. System displays brief success notification  14. Job remains saved for future access  8.0.2 Save Job from Detail Page  1. Job seeker views job detail page  2. System displays save button in job action section  3. System checks if job is already saved  4. System shows current save status  5. Job seeker clicks save button  6. System executes save operation (same as flow 8.0.1 steps 6-13)  7. Job is added to saved jobs list  8.0.3 View Saved Jobs List  1. Job seeker clicks "Saved Jobs" in navigation menu  2. System sends GET request to /saved-jobs servlet  3. System validates job seeker session  4. System retrieves all saved jobs from database with query joining:  - SavedJobs table  - Jobs table  - Recruiter table  - Locations table  - Types table  5. System orders saved jobs by SavedDate descending (newest first)  6. System displays saved jobs page with job cards showing:  - Job title  - Company name  - Location  - Salary range  - Job type  - Posted date  - Saved date  - "View Details" button  - "Unsave" button  7. System shows total count of saved jobs  8. Job seeker views saved jobs list  8.0.4 Remove Job from Saved List  1. Job seeker is on saved jobs page or viewing any job  2. Job seeker clicks unsave button (heart icon or "Remove" button)  3. System displays confirmation "Remove this job from saved list?"  4. Job seeker confirms removal  5. System sends POST request to /saved-jobs servlet with action=unsave and jobId  6. System validates user authentication and job ownership  7. System deletes record from SavedJobs table where JobSeekerID matches and JobID matches  8. System returns success response "Job removed from saved list"  9. System updates UI:  - If on saved jobs page: removes job card from list  - If on job detail page: changes icon to empty heart  10. System decrements saved jobs count  11. Job is removed from saved list successfully  8.0.5 Check Save Status  1. Job seeker views any job listing or detail page  2. System sends POST request to /saved-jobs servlet with action=check and jobId  3. System queries SavedJobs table for matching JobSeekerID and JobID  4. System returns response with isSaved boolean value  5. System updates UI icon based on saved status  6. Job seeker sees current save state visually | | |
| Alternative Flows: | 8.1 Toggle Save/Unsave from Same Button  Branches from: Step 5 of flow 8.0.1 or 8.0.2  Condition: User clicks already-saved job  1. Job seeker clicks save button on already saved job  2. System detects job is currently saved  3. System executes unsave operation instead  4. System removes job from saved list  5. System updates icon to empty heart  6. System shows "Job removed from saved list" message  7. Rejoins normal flow at: Step 14 of flow 8.0.1  8.2 Empty Saved Jobs List  Branches from: Step 6 of flow 8.0.3  Condition: User has no saved jobs  1. System retrieves empty list from database  2. System displays empty state message  3. System shows icon and text "You haven't saved any jobs yet"  4. System provides "Find Jobs" button  5. Job seeker clicks "Find Jobs"  6. System navigates to job listing page  7. Use case ends  8.3 Save Job While Not Logged In  Branches from: Step 5 of flow 8.0.1  Condition: User is not authenticated  1. Job seeker clicks save button without being logged in  2. System detects no active session  3. System displays login prompt "Please login to save jobs"  4. System redirects to login page  5. Job seeker logs in  6. System redirects back to previous page  7. Rejoins normal flow at: Step 5 of flow 8.0.1  8.4 Quick Apply from Saved Jobs  Branches from: Step 8 of flow 8.0.3  Condition: User wants to apply directly from saved list  1. Job seeker clicks "View Details" on saved job  2. System navigates to job detail page  3. Job seeker clicks "Apply Now"  4. System opens application modal  5. Job seeker submits application  6. Use case transitions to UC07 Job Application | | |
| Exceptions: | 8.0.E1 Session Expired  Occurs at: Any step requiring authentication  Condition: User session is invalid  1. System detects expired or invalid session  2. System returns error "Session expired. Please login again"  3. System redirects to login page  4. Use case ends with failure  5. Recovery: User logs in and retries operation  8.0.E2 Database Connection Failure  Occurs at: Any database operation  Condition: Database is unavailable  1. System attempts database query  2. System catches SQLException  3. System logs error details  4. System displays error "Service temporarily unavailable"  5. Use case ends with failure  6. Recovery: Admin restores connection, user retries  8.1.E1 Duplicate Save Attempt  Occurs at: Step 9 of flow 8.0.1  Condition: Job already exists in saved list  1. System checks SavedJobs table  2. System finds existing record  3. System prevents duplicate insertion  4. System returns response "Job already saved"  5. System keeps UI in saved state  6. Use case ends successfully (no change needed)  8.1.E2 Invalid Job ID  Occurs at: Step 8 of flow 8.0.1  Condition: JobID does not exist or is invalid  1. System validates JobID parameter  2. System finds job does not exist  3. System returns error "Job not found"  4. System displays error message to user  5. Use case ends with failure  8.4.E1 Job No Longer Available  Occurs at: Step 6 of flow 8.0.3  Condition: Saved job was deleted or closed  1. System retrieves saved jobs list  2. System finds some jobs no longer published  3. System still displays job in saved list with "No longer available" badge  4. Job seeker can view details but cannot apply  5. Job seeker can remove from saved list  6. Use case continues normally  8.4.E2 Remove Operation Fails  Occurs at: Step 7 of flow 8.0.4  Condition: Database error during deletion  1. System attempts to delete record  2. System encounters SQLException  3. System logs error  4. System displays error "Failed to remove job. Please try again"  5. System keeps job in saved list  6. Use case ends with failure  7. Recovery: User retries removal  8.0.E3 Network Connection Lost  Occurs at: Any AJAX request  Condition: Client loses connection  1. System attempts to send request  2. Request times out  3. JavaScript catches error  4. System displays error "Connection lost. Please check your network"  5. System preserves current UI state  6. Use case is suspended  7. Recovery: User restores connection and retries | | |
| Priority: | Medium | | |
| Frequency of Use: | Save Job Action: 3-8 times per user per session  View Saved Jobs: 2-5 times per user per week  Unsave Job: 1-3 times per user per week  Check Save Status: Automatic on every job view  Estimated Daily Save Actions: 500-1500 across all users  Average Saved Jobs per User: 5-15 jobs | | |
| Business Rules: | BR-023 - BR-030 | | |
| Other Information: | Heart icon must clearly indicate saved vs not saved state  One-click save/unsave toggle for ease of use  Visual feedback on save actions (animation, notification)  Empty state must encourage job exploration  Mobile-friendly touch targets for save buttons | | |
| Assumptions: | ASSUM-25: Job seekers understand save function is for personal tracking  ASSUM-26: Users have stable connection for real-time updates  ASSUM-27: Job seekers regularly review saved jobs  ASSUM-28: Heart icon universally understood as save/favorite  ASSUM-29: Users know saved jobs do not auto-apply | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| **BR-023** | Save Uniqueness Rule | Each job can be saved only once per job seeker (no duplicate saves). |
| **BR-024** | Saved Job Persistence | Saved jobs persist indefinitely until manually removed by the job seeker. |
| **BR-025** | Save Limit | Job seekers can save an unlimited number of jobs. |
| **BR-026** | Save Date Recording | The saved date is recorded for sorting and tracking purposes. |
| **BR-027** | Deleted or Expired Job Handling | Deleted or expired jobs remain in the saved list but are clearly marked as unavailable. |
| **BR-028** | Privacy of Saved List | Saved jobs are private and accessible only to the respective job seeker. |
| **BR-029** | Recruiter Notification | The save action does not send notifications to recruiters. |
| **BR-030** | Save and Unsave Restrictions | Job seekers can save and unsave jobs without any restrictions (e.g., no cool-down period between actions). |

### 1.9 UC09\_View company

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC09\_View company | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 16/10/2025 |
| Primary Actor: | Jobseeker | Secondary Actors: | None |
| Trigger: | Job seeker clicks on company name or "View Company Profile" link from job listing, job detail page, or company directory. | | |
| Description: | This use case allows job seekers to view comprehensive information about recruiting companies before applying for positions. Job seekers can access detailed company profiles including company introduction, corporate culture, company size, industry, benefits offered, work locations, current job openings, and contact information. This information helps job seekers evaluate whether the company aligns with their career goals and values before submitting applications. | | |
| Preconditions: | PRE-1: Company profile must exist in database with valid RecruiterID  PRE-2: Company information must be published and approved  PRE-3: Database connection must be available  PRE-4: User can be authenticated or guest (no login required) | | |
| Postconditions: | OST-1: Company information is displayed to job seeker  POST-2: Company profile view count is incremented  POST-3: Related job openings are shown  POST-4: Job seeker can save jobs or apply directly from company page  POST-5: Company contact information is accessible | | |
| Normal Flow: | .0.1 Access Company Profile  1. Job seeker encounters company name in job listing or job detail page  2. Job seeker clicks company name link or "View Company Profile" button  3. System sends GET request to /company-profile servlet with company recruiterId  4. System validates recruiterId parameter  5. System retrieves company information from Recruiter table  6. System redirects to company profile page with company data  9.0.2 View Company Overview  1. System displays company profile page header with:  - Company name  - Company logo  - Industry/Category  - Company size (number of employees)  - Location/Address  - Website link  - Contact phone number  2. System shows company introduction section with:  - Company description and overview  - Mission and vision statement  - Core values  3. System displays company benefits section listing:  - Salary and compensation  - Health insurance  - Annual leave and holidays  - Training and development opportunities  - Work environment perks  - Other benefits  4. Job seeker reads company information  5. Job seeker scrolls through sections to learn about company  9.0.3 View Company Culture  1. System displays company culture section  2. System shows company images (office photos, team photos, events)  3. System displays company video if available (embedded YouTube/video player)  4. System shows company culture highlights:  - Work environment  - Team activities  - Company events  - Employee testimonials (if available)  5. Job seeker views visual content about company culture  6. Job seeker plays company introduction video (optional)  9.0.4 View Job Openings  1. System queries Jobs table for all active jobs by company (recruiterId)  2. System filters jobs with status "Published"  3. System displays list of current job openings with:  - Job title  - Job type (Full-time, Part-time, etc.)  - Location  - Salary range  - Posted date  - Number of applicants  4. System shows total count of available positions  5. Job seeker browses company job openings  6. Job seeker can click job to view details or apply  9.0.5 View Contact Information  1. System displays company contact section with:  - Company address  - Office phone number  - Contact person name  - Email address (if available)  - Website URL  - Social media links (if available)  2. System shows office location on map (if map integration available)  3. Job seeker views contact details  4. Job seeker can click website link to open in new tab | | |
| Alternative Flows: | .1 Access Company Profile from Job Detail  Branches from: Job detail page  Condition: Job seeker viewing specific job details  1. Job seeker views job detail page  2. Job seeker sees company information sidebar  3. Job seeker clicks "View Company Profile" link  4. Rejoins normal flow at: Step 3 of flow 9.0.1  9.2 Access Company from Company Directory  Branches from: Company listing page  Condition: Job seeker browsing all companies  1. Job seeker navigates to company directory/culture page  2. System displays list of all active companies  3. Job seeker filters by industry or location  4. Job seeker clicks on company card  5. Rejoins normal flow at: Step 3 of flow 9.0.1  9.3 Apply Directly from Company Page  Branches from: Step 6 of flow 9.0.4  Condition: Job seeker wants to apply for listed job  1. Job seeker clicks "Apply Now" on job listing  2. System validates authentication (redirects to login if needed)  3. System opens application modal  4. Use case transitions to UC07 Job Application  9.4 Save Job from Company Page  Branches from: Step 6 of flow 9.0.4  Condition: Job seeker wants to save job for later  1. Job seeker clicks save button (heart icon) on job card  2. System validates authentication  3. System saves job to saved jobs list  4. Use case transitions to UC08 Job Management  9.5 Visit Company Website  Branches from: Step 5 of flow 9.0.5  Condition: Job seeker clicks external website link  1. Job seeker clicks website URL link  2. System opens company website in new browser tab  3. Job seeker views external company website  4. Job seeker returns to profile page  5. Use case continues | | |
| Exceptions: | 9.0.E1 Invalid Company ID  Occurs at: Step 4 of flow 9.0.1  Condition: RecruiterID does not exist or is invalid  1. System validates recruiterId parameter  2. System finds no matching company record  3. System displays error page "Company not found"  4. System provides link to return to job listings  5. Use case ends with failure  9.0.E2 Company Profile Incomplete  Occurs at: Step 5 of flow 9.0.1  Condition: Company has minimal information filled  1. System retrieves company record  2. System finds many fields are null or empty  3. System displays available information only  4. System shows message "Company profile is being updated"  5. Job seeker sees limited information  6. Use case continues with partial data  9.0.E3 No Active Jobs Available  Occurs at: Step 2 of flow 9.0.4  Condition: Company has no published job openings  1. System queries for active jobs  2. System returns empty result set  3. System displays message "No job openings available at this time"  4. System suggests checking back later  5. Use case continues normally  9.0.E4 Database Connection Failure  Occurs at: Step 5 of flow 9.0.1  Condition: Database is unavailable  1. System attempts to retrieve company data  2. System catches SQLException  3. System logs error details  4. System displays error "Service temporarily unavailable"  5. Use case ends with failure  6. Recovery: Admin restores connection, user retries  9.3.E1 Broken Company Website Link  Occurs at: Step 2 of flow 9.5  Condition: Website URL is invalid or site is down  1. Job seeker clicks website link  2. Browser attempts to open URL  3. Browser shows "Page not found" or timeout  4. Job seeker returns to company profile  5. Job seeker can report broken link (optional)  6. Use case ends  9.0.E5 Media Content Load Failure  Occurs at: Step 2-3 of flow 9.0.3  Condition: Images or video fail to load  1. System attempts to load company images or video  2. Media fails to load (file not found, network error)  3. System shows placeholder image or error icon  4. System continues displaying other available content  5. Job seeker can still view text information  6. Use case continues with degraded experience | | |
| Priority: | Medium Priority - Enhances job seeker decision-making and builds employer brand trust but not critical for core application flow. | | |
| Frequency of Use: | Company Profile Views: 2-5 times per user per session  Company Profile from Job Detail: 50-70% of job detail page views  Direct Company Search: 1-2 times per user per week  Average Time on Company Page: 2-4 minutes  Estimated Daily Views: 1000-3000 across all companies  Peak Hours: Same as job browsing (9-11 AM, 1-3 PM, 7-9 PM) | | |
| Business Rules: | BR-031 - BR-039 | | |
| Other Information: | Company information must be well-organized and scannable  Visual hierarchy should guide users through content sections  Company benefits should be clearly listed and easy to read  Job openings should be prominently displayed  Mobile-responsive design for all screen sizes  Clear call-to-action buttons for applications | | |
| Assumptions: | ASSUM-30: Companies maintain up-to-date profile information  ASSUM-31: Job seekers understand company research improves application success  ASSUM-32: Company logos and images are appropriate and professional  ASSUM-33: Website URLs are valid and accessible  ASSUM-34: Users have browsers that support HTML5 video  ASSUM-35: Company descriptions are in Vietnamese or English  ASSUM-36: Job seekers compare multiple companies before applying  ASSUM-37: Company culture content is authentic and representative | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR-031 | Public Accessibility of Profiles | Company profiles are publicly accessible (no login required to view). |
| BR-032 | Admin Approval for Display | Company information must be approved by the Admin before it can be publicly displayed. |
| BR-033 | Job Visibility on Profile | Only published jobs are shown on the company profile page. |
| BR-034 | View Count Tracking | Company profile view count is tracked for analytics. |
| BR-035 | Incomplete Profile Viewability | Incomplete company profiles can still be viewed by the public. |
| BR-036 | Contact Information Visibility | Company contact information is visible to all users. |
| BR-037 | Categorization of Size and Industry | Company size and industry must be selected from predefined lists/categories. |
| BR-038 | Expired/Closed Jobs Exclusion | Expired or closed jobs are not displayed on the company page. |
| BR-039 | Media Approval for Appropriateness | Company logo and images must be approved by the Admin for appropriateness. |

### 1.10 UC10\_Notify

#### *a. Functionalities*

**Functional Description Template**

| UC ID and Name: | UC10\_Notify | | |
| --- | --- | --- | --- |
| Created By: | DuongNT | Date Created: | 16/10/2025 |
| Primary Actor: | Job Seeker | Secondary Actors: | None |
| Trigger: | System automatically creates notification when significant events occur (application status change, interview invitation, new job matches), or user clicks notification icon to view notifications. | | |
| Description: | This use case enables job seekers to receive, view, and manage system notifications regarding important events such as application status updates, interview invitations, new suitable job postings, and system announcements. Users can view notification list with read/unread status, mark notifications as read, filter notifications by status, and navigate to related content through notification links. The system maintains notification history and provides real-time badge count of unread notifications. | | |
| Preconditions: | PRE-1: User must be authenticated and logged in  PRE-2: User session must be active and valid  PRE-3: Notifications table must exist in database  PRE-4: Database connection must be available  PRE-5: Notification servlet must be properly configured | | |
| Postconditions: | POST-1: Notifications are displayed to user with correct read/unread status  POST-2: Unread count badge is updated in real-time  POST-3: Selected notifications are marked as read in database  POST-4: ReadAt timestamp is recorded when notification is marked read  POST-5: User can navigate to related content from notifications  POST-6: Notification history is maintained for user reference | | |
| Normal Flow: | 10.0.1 View Notification Badge  1. Job seeker logs into system  2. System loads user dashboard or any page with navigation header  3. System sends GET request to /notifications?action=getUnreadCount  4. System queries Notifications table for count where userID matches, userType is "jobseeker", and isRead is false  5. System returns unread count to client  6. System displays notification icon with badge showing unread count  7. Badge shows number if count greater than 0, hidden if count is 0  8. Job seeker sees notification indicator  10.0.2 Open Notification Dropdown  1. Job seeker clicks notification bell icon in header  2. System opens notification dropdown panel  3. System sends GET request to /notifications?action=getRecent&limit=20  4. System retrieves recent 20 notifications from database ordered by createdAt descending  5. System displays notification list in dropdown with:  - Notification icon based on type (application, profile, system)  - Notification title  - Notification message  - Time ago (e.g., "2 hours ago", "1 day ago")  - Blue dot indicator for unread notifications  6. System displays filter tabs: "All", "Unread", "Read"  7. Job seeker views notification list  8. Default tab is "All" showing all notifications  10.0.3 Filter Notifications  1. Job seeker clicks "Unread" or "Read" tab  2. System filters notification list based on selected tab:  - All: Display all notifications  - Unread: Display only notifications where isRead is false  - Read: Display only notifications where isRead is true  3. System updates notification list display  4. System shows empty state message if no notifications match filter  5. Job seeker views filtered notifications  10.0.4 Mark Notification as Read  1. Job seeker clicks on a notification item in dropdown  2. System checks if notification is unread  3. If unread, system sends POST request to /notifications?action=markAsRead with notificationID  4. System updates Notifications table:  - Set isRead to true  - Set readAt to current timestamp  5. System returns success response  6. System updates UI immediately:  - Remove blue dot indicator from notification  - Remove unread styling  - Decrement badge count by 1  7. If on "Unread" filter, remove notification from list  8. Notification is marked as read successfully  10.0.5 Navigate from Notification  1. Job seeker clicks on notification with actionURL  2. System retrieves actionURL from notification data  3. System navigates to specified URL (e.g., /applied-jobs, /job-detail?jobId=123)  4. System closes notification dropdown  5. Job seeker views related content page  10.0.6 Receive Automatic Notification  1. System event triggers notification creation (e.g., application status change)  2. System calls NotificationDAO.sendNotification() with parameters:  - userID (recipient job seeker ID)  - userType ("jobseeker")  - notificationType (application, profile, job, system)  - title (notification headline)  - message (detailed message)  - relatedID (e.g., applicationID, jobID)  - relatedType (application, job, user)  - actionURL (link to related page)  - priority (0=low, 1=medium, 2=high)  3. System inserts new record into Notifications table with:  - isRead set to false  - createdAt set to current timestamp  4. If user is currently logged in, system updates badge count in real-time  5. Notification appears in dropdown on next refresh or page load  6. Job seeker receives notification successfully | | |
| Alternative Flows: | 10.1 Empty Notification List  Branches from: Step 7 of flow 10.0.2  Condition: User has no notifications  1. System retrieves empty notification list  2. System displays empty state in dropdown:  - Icon (inbox icon)  - Message "No notifications yet"  3. Job seeker sees empty state  4. Use case ends  10.2 Mark All as Read  Branches from: Step 8 of flow 10.0.2  Condition: User wants to mark all notifications as read at once  1. Job seeker clicks "Mark All as Read" button (if implemented)  2. System sends POST request to /notifications?action=markAllAsRead  3. System updates all unread notifications for user to isRead true  4. System updates all readAt timestamps  5. System returns success response  6. System updates UI:  - Remove all blue dots  - Update badge count to 0  - Refresh notification list  7. All notifications marked as read  8. Use case ends  10.3 Notification with Priority  Branches from: Step 6 of flow 10.0.6  Condition: High priority notification created  1. System creates notification with priority 2 (high)  2. System displays notification with special styling:  - Highlighted background  - Bold title  - Priority icon or indicator  3. High priority notifications appear at top of list  4. Job seeker notices important notification immediately  5. Rejoins normal flow at: Step 8 of flow 10.0.2  10.4 Load More Notifications  Branches from: Step 7 of flow 10.0.2  Condition: User wants to see more than initial 20 notifications  1. Job seeker clicks "View All Notifications" link in dropdown  2. System navigates to full notifications page (if implemented)  3. System loads all notifications with pagination  4. Job seeker browses complete notification history  5. Use case ends | | |
| Exceptions: | 10.0.E1 Session Expired  Occurs at: Any step requiring authentication  Condition: User session is invalid or expired  1. System detects no active session  2. System returns error response  3. System redirects to login page  4. Use case ends with failure  5. Recovery: User logs in and notifications load normally  10.0.E2 Database Connection Failure  Occurs at: Any database query operation  Condition: Database is unavailable  1. System attempts to query Notifications table  2. System catches SQLException  3. System logs error details  4. System displays error in dropdown "Unable to load notifications"  5. System shows retry button  6. Use case ends with failure  7. Recovery: Admin restores connection, user clicks retry  10.4.E1 Mark as Read Fails  Occurs at: Step 4 of flow 10.0.4  Condition: Database update fails  1. System attempts to update notification  2. System encounters SQLException  3. System logs error  4. System returns failure response  5. System displays error message "Failed to mark as read"  6. Notification remains in unread state  7. Badge count not decremented  8. Use case ends with failure  9. Recovery: User retries clicking notification  10.6.E1 Invalid Notification ID  Occurs at: Step 3 of flow 10.0.4  Condition: NotificationID does not exist  1. System attempts to mark notification as read  2. System finds no matching notification record  3. System returns error "Notification not found"  4. System removes notification from UI (cleanup)  5. System adjusts badge count  6. Use case ends  10.0.E3 Notification Creation Fails  Occurs at: Step 3 of flow 10.0.6  Condition: Error during notification insertion  1. System attempts to insert notification  2. System encounters database constraint violation or error  3. System logs error details  4. System continues operation without notification  5. User does not receive notification (silent failure)  6. Admin can review logs for failed notifications  7. Use case continues (main operation not affected)  10.0.E4 JSON Parse Error  Occurs at: Step 5 of flow 10.0.2  Condition: Response from server is malformed  1. Client JavaScript receives response  2. JSON.parse() throws error  3. JavaScript catches error  4. System displays error in dropdown "Failed to load notifications"  5. System logs error to console  6. Use case ends with failure  7. Recovery: User refreshes page | | |
| Priority: | High | | |
| Frequency of Use: | iew Badge Count: Every page load when logged in (continuous)  Open Notification Dropdown: 5-10 times per user per session  Mark as Read: 3-8 notifications per session  New Notifications Created: 2-5 per user per day  Check Notifications: Daily for active users  Peak Notification Times: After application submissions, during business hours | | |
| Business Rules: | None | | |
| Other Information: | adge clearly visible and updates in real-time  Notification dropdown accessible from all pages  Clear visual distinction between read and unread notifications  Time ago format (e.g., "2 hours ago") more user-friendly than timestamps  Empty state provides guidance  Icon types help users quickly identify notification category  Mobile-responsive dropdown for small screens | | |
| Assumptions: | ASSUM-38: Job seekers check notifications regularly  ASSUM-39: Users understand notification icons and color coding  ASSUM-40: Notifications are delivered within seconds of event occurrence  ASSUM-41: Users have JavaScript enabled in browser  ASSUM-42: Database has sufficient capacity for notification volume  ASSUM-43: Notification messages are clear and actionable  ASSUM-44: Users understand time ago format  ASSUM-45: Badge count is sufficient indicator without real-time push | | |

#### *b. Business Rules*

*None*

## 2. Common Functions

### 2.1 UC-19\_Register Account

#### *a. Functional Description*

| UC ID and Name: | **UC-20\_Register System** | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 28/10/2025 |
| Primary Actor: | Common | Secondary Actors: | None |
| Trigger: | The login user must have an internet connection.  User clicks the "Register" or "Sign Up" button on the authentication/login page and decides to create a new account. | | |
| Description: | This use case allows new users to create an account by providing necessary information and selecting their role/category. After successful registration, the system authenticates the user's credentials and redirects them to a role-specific dashboard or home page that matches their selected role. | | |
| Preconditions: | PRE-1: User is not already logged into the system  PRE-2: User has access to the registration page  PRE-3: System is operational and database is accessible | | |
| Postconditions: | * New user account has been created in the system database * User credentials have been authenticated * User is logged in and authenticated in the current session * User has been redirected to role-appropriate dashboard/page | | |
| Normal Flow | **2.0 Register System**   1. User navigates to the registration page 2. System displays registration form with fields (name, email, password...) 3. User enters required information 4. User clicks "Register" button 5. System validates input data format and completeness 6. System checks if Field already exists in database 7. System creates new user account 8. System authenticates user credentials 9. System creates active session for user 10. System redirects user to role-specific dashboard/home page 11. System displays welcome message with user's role information | | |
| Alternative Flows: | ***2.0.1 Information Validate***   1. User enters the required information. 2. System will notify specific errors of the information. 3. User proceeds to edit information 4. Return to step 4 of Normal Flow | | |
| Exceptions: | ***2.0.E1 Invalid Field Format:***   1. Occurs at step 5 in normal flow 2. System displays error message 3. System returns focus to invalid field 4. User re-enters field and continues   ***2.0.E2 Field Already Exists:***   1. Occurs at step 6 in normal flow 2. System displays error message 3. System returns focus to Field Already Exists 4. User re-enters field and continues   ***2.0.E3 Database Connection Error:***   1. Occurs at step 9 in normal flow 2. System displays error message 3. User is redirected to login page 4. User must log in manually | | |
| Priority: | Must Have | | |
| Frequency of Use: | Estimated 10-50 times per day (depending on system maturity and user base growth) | | |
| Business Rules: | FR1, FR2 | | |
| Other Information: | 1. Quality Attributes: Security, Usability, Performance 2. Related Functional Requirements: FR-001 (User Authentication), FR-002 (Role Management) 3. Failure Handling: All database changes are rolled back on exception; user returns to registration form with error messages preserved 4. System should log all registration attempts for audit purposes | | |
| Assumptions: | 1. User has valid email address 2. User will provide accurate information during registration 3. System roles are pre-configured and available 4. Email service is reliable for verification (if applicable) 5. User has stable internet connection during registration process | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| FR1 | Password Encoding | User’s password must be encoded with MD5 hashing |
| FR2 | Field | Some fields must be unique within the same role. |

### 2.1 UC-20\_Login System

#### *a. Functional Description*

| UC ID and Name: | **UC-20\_Login System** | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 28/10/2025 |
| Primary Actor: | Common | Secondary Actors: | None |
| Trigger: | The login user must have an internet connection.  User clicks Login button from the page header, or  User accesses an authenticated feature (from a link or type the page URL directly into the address bar) | | |
| Description: | As a user, I want to be able to log into the system so that I can use the system’s authenticated features and access my personalized account. | | |
| Preconditions: | User account has been created. | | |
| Postconditions: | * User logs in the system successfully | | |
| Normal Flow | **2.0 Login System**  1. User accesses the User Login screen  2. User types in the login details or choo other login options .  3. User clicks the Login button  4. System validates the login details  5. System allows user to access  6. System accesses the Home Page (or the previous calling page if any) | | |
| Alternative Flows: | ***2.1 Google Login***  1. User chooses to login system using Google account  2. System redirects the user to the Google’s Login screen  3. User types in the Google account details and chooses to login  4. Google validates user’s login information successfully and redirect him/her back to the system  5. Return to step 5 of normal flow. | | |
| Exceptions: | ***2.1.E1 System can’t authenticate the user***  1. The Error Message screen is shown to the user  2. User cancels the logging in *=> UC stops, change to View Home Page*  3. User clicks “Forgot Password?” link *=> change to View Reset Password*  4. User clicks “Register” link *=> change to Register User Account* | | |
| Priority: | Must Have | | |
| Frequency of Use: | Very High (e.g., every active user logs in at least once a day on all weekdays; peak times can have hundreds of requests per hour). | | |
| Business Rules: | FR1, FR2 | | |
| Other Information: | 1. The login feature supports all modern web browsers and devices. 2. Each user must have a valid account with a unique username and password. 3. The system may log unsuccessful login attempts for security monitoring. | | |
| Assumptions: | 1. Users are already registered in the system before attempting to log in. 2. Users have a stable internet connection to access the system. 3. User credentials are kept confidential and not shared with others. | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| FR1 | Password Encoding | User’s password must be encoded with MD5 hashing |
| FR2 | Invalid Logging In | User can’t be authenticated to login the system if below cases   * His/her logging-in details are incorrect * His/her account has not been verified * His/her account has been locked or blocked |

### 2.2 UC-21\_Reset Password

#### *a. Functional Description*

| UC ID and Name: | **UC-21\_Reset Password** | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 28/10/2025 |
| Primary Actor: | Common | Secondary Actors: | None |
| Trigger: | The login user must have an internet connection.  User clicks "Forgot Password" link on the login page or clicks "Reset Password" in account settings. | | |
| Description: | This use case allows users who have forgotten their login password to reset it. The system verifies the user's identity through email verification, generates a reset link, and allows the user to create a new password. After successful password reset, the user can log in with the new credentials. | | |
| Preconditions: | User account has been created. | | |
| Postconditions: | Reset Password Sucessful | | |
| Normal Flow | **2.2 Reset Password**   1. User navigates to login page and clicks "Forgot Password" link 2. System displays password reset request form with email input field 3. User enters their registered email address 4. User clicks "Submit" or "Send Reset Link" button 5. System validates if email exists in the database (via LoginService) 6. System generates a unique reset token (via ResetService) 7. System sends password reset email containing reset link with token to user's email 8. System displays confirmation message: "Reset link has been sent to your email" 9. User receives email and clicks on the reset password link 10. System extracts token from URL and validates token 11. System verifies token is not expired and belongs to valid user 12. System displays password reset form with new password and confirm password fields 13. User enters new password and confirms it 14. User clicks "Reset Password" button 15. System validates new password strength and format (via ResetService) 16. System encrypts new password using MD5 encryption (MD5Util) 17. System updates user password in database (via LoginService or UserDAO) 18. System marks reset token as used/expired in database 19. System displays success message: "Password has been reset successfully" 20. System redirects user to login page 21. User logs in with new credentials | | |
| Alternative Flows: | ***2.2.9 Resend Reset Email:***   1. Branches at step 9 in normal flow 2. User doesn't receive email and requests to resend reset link 3. System regenerates new token 4. System sends new reset email 5. Continue from step 9   ***2.2.5 Check Email:***   1. Branch in step 5 in normal process 2. System reports "Request sent failed" error 3. User re-enters email 4. Continues from step 4 | | |
| Exceptions: | ***2.2.E1 Email Not Found in System:***   1. Occurs at step 5 in normal flow 2. System displays error: "Email address not found in our system" 3. System returns to step 3 4. User can enter different email or exit   ***2.2.E2 Email Service Failure:***   1. Occurs at step 8 in normal flow 2. System cannot send email due to email service unavailability 3. System displays error: "Unable to send reset email. Please try again later" 4. Transaction is rolled back; token is deleted from database 5. User can retry the password reset process   ***2.2.E3 Invalid or Expired Reset Token:***   1. Occurs at step 12 in normal flow 2. System detects token is expired (beyond expiration time limit) 3. System displays error: "Reset link has expired. Please request a new one" 4. User is redirected to password reset request page 5. User can request a new reset link   ***2.2.E4 Token Tampering Detected:***   1. Occurs at step 12 in normal flow 2. System detects invalid/corrupted token format 3. System displays error: "Invalid reset link" 4. User is redirected to password reset request page   ***2.2.E5 Weak Password Format:***   1. Occurs at step 16 in normal flow 2. New password doesn't meet security requirements (length, special characters, etc.) 3. System displays error: "Password must contain at least 8 characters including letters and numbers" 4. System returns focus to password field 5. User re-enters stronger password   ***2.2.E6 Passwords Do Not Match:***   1. Occurs at step 16 in normal flow 2. Password and confirm password fields don't match 3. System displays error: "Passwords do not match" 4. User re-enters passwords   ***2.2.E7 Database Update Failure:***   1. Occurs at step 18 in normal flow 2. Database cannot update user password due to connection error or constraint violation 3. System displays error: "Password reset failed. Please try again" 4. Transaction is rolled back; old password remains in database 5. User can retry the reset process   ***2.2.E8 Session Timeout During Process:***   1. Occurs at step 14 in normal flow 2. User session expires while filling reset form 3. System displays error: "Your session has expired. Please restart the password reset process" 4. User is redirected to password reset request page | | |
| Priority: | High | | |
| Frequency of Use: | Estimated 5-20 times per day (depending on user base size and password management practices) | | |
| Business Rules: | FR1, FR2, FR3, FR4, FR5, FR6 | | |
| Other Information: | 1. Quality Attributes: Security, Reliability, Usability, Performance 2. Related Functional Requirements:    * FR-001: User Authentication    * FR-002: Email Notification Service    * FR-003: Password Encryption (MD5/Hashing)    * FR-004: Token Management 3. Failure Handling:    * Email service failure: Transaction rolled back, token deleted    * Database failure: Transaction rolled back, original password remains intact    * Token expiration: User must request new reset link    * Network timeout: User can retry process 4. Security Considerations:    * Use HTTPS for all password reset communications    * Log all password reset attempts for audit trails    * Implement rate limiting to prevent password reset token abuse    * Clear reset tokens after successful use    * Don't expose whether email exists in error messages (prevent user enumeration) | | |
| Assumptions: | 1. User has access to their registered email account 2. User remembers their registered email address 3. Email system is reliable and secure 4. Password reset tokens are generated with cryptographic strength 5. Users will reset password within the token expiration window 6. System maintains audit logs of password changes 7. MD5 encryption is used for password storage (or stronger hashing algorithm) 8. User has stable internet connection during reset process | | |

#### *b. Business Rules*

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| FR1 | Token Expiration Requirement | Reset password token must automatically expire after 10 minute from creation. System must invalidate any token that exceeds this time window. |
| FR2 | Password Strength Requirement | New password must meet minimum security standards: minimum 8 characters, contain at least one uppercase letter, one lowercase letter, one number, and one special character. |
| FR3 | Single Active Token Policy | User can only have one active reset token at any given time. Generating a new token must invalidate all previously issued tokens for that user. |
| FR4 | Email Verification Rule | Password reset email can only be sent to the email address registered in the user's account. System must not reveal whether an email exists in the system. |
| FR5 | Token One-Time Use | Each reset token can only be used once. After successful password reset, the token must be marked as used and cannot be reused. |
| FR6 | Session Invalidation on Reset | All active sessions for the user must be invalidated after successful password reset. User must log in again with new credentials. |

## 

## 3. Patron Feature

### 3.1 UC-2\_Pay for job package

#### *a. Functional Description*

| ID and Name: | **UC-2 Pay for job package** | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 05/10/25 |
| Primary Actor: | Recruiter | Secondary Actors: | Payment System(VNPAY) |
| Description: | Recruiters purchase posting packages or service packages and pay through VNPay payment gateway. After payment, recruiters can manage the packages they have purchased. | | |
| Trigger: | Recruiter clicks on "Purchase Package" or "View Packages" button | | |
| Preconditions: | PRE-1. Recruiter account has been created and authenticated in the system.  PRE-2. Recruiter has stable internet connection  PRE-3. Job packages are configured in the system (JobPackagesDAO). | | |
| Postconditions: | POST-1: Payment transaction is recorded in PaymentsDAO / PaymentDetailsDAO.  POST-2: Package purchase record is created in RecruiterPackagesDAO.  POST-3: Recruiter receives payment confirmation via email (CandidateEmailService).  POST-4: Package status is updated and available for recruiter to use. | | |
| Normal Flow: | **3.0 Order a Single Meal**   1. The employer visits the job package selection page. 2. The system displays the packages available from JobPackagesDAO with prices and features | 3. Recruiter selects package (Silver, Bronze, Gold..) 4. System displays package details and total price 5. Recruiter clicks on "Pay" button 6. System creates payment order and generates payment gateway URL 7. System redirects recruiter to VNPay payment gateway (VNPayConfig) 8. VNPay displays payment form 9. Recruiter enters payment information and completes payment 10. VNPay processes payment 11. VNPay redirects recruiter back to system with payment status 12. System validates payment response and updates PaymentsDAO 13. System verifies transaction status 14. System creates purchase record in RecruiterPackagesDAO 15. System records package assignment to recruiter account 16. System displays successful notification to recruiter 17. Recruiter is redirected to package management dashboard 18. Recruiter can view purchased packages and remaining balance 19. System displays active packages and recruiter usage statistics | | |
| Alternative Flows: | **3.1 Recruiter cancels package selection**   1. In step 3, the recruiter leaves the page without selecting a package. 2. The system returns to the dashboard without creating any order.   **3.2 Recruiter cancels payment at VNPay**   1. At the VNPay page (Step 9), the recruiter cancels payment. 2. VNPay redirects back with a “User Cancelled” status. 3. The system records the failed payment in **PaymentsDAO**. 4. The system notifies the recruiter: “Payment was cancelled. No charges applied.” 5. No package is assigned.   ***3.3 Payment completed but email confirmation fails***   1. After step 15, the system attempts to send a confirmation email (optional). 2. Email Service fails → system logs the error. Payment and package assignment remain successful. 3. The recruiter receives a warning message: “Payment successful, but the email could not be sent.” | | |
| Exceptions: | **3.0.E1 Payment signature invalid (security risk)**   1. After step 15, the system attempts to send a confirmation email (optional). 2. Email Service fails → system logs the error. 3. Payment and package assignment remain successful. 4. The recruiter receives a warning message: “Payment successful, but the email could not be sent.”   **3.1.E2 PaymentsDAO update failure**   1. In step 12, database error prevents saving payment record. 2. System displays error message. 3. Transaction is marked as failed. 4. No package is assigned.   **3.2.E3 RecruiterPackagesDAO creation failure**   1. Payment is successful but system cannot assign package due to DB error. 2. System logs the error. 3. System displays: “Payment successful, but package assignment failed.” Admin intervention may be required. | | |
| Priority: | Must Have | | |
| Frequency of Use: | High (e.g., recruiters purchase packages multiple times during recruitment cycle; peak times can have hundreds of purchase transactions per hour) | | |
| Business Rules: | BR1 - BR6 | | |
| Other Information: | **Quality Attributes:**   1. The system must load the list of job packages within **2 seconds**. 2. The payment redirection process must be smooth with clear instructions and labels. 3. All communication with VNPay must use **secure HTTPS protocol** and follow VNPay security signatures. 4. The package assignment process must complete within **1 second** after payment confirmation. 5. The interface must clearly display payment status (Success, Failed, Pending) with user-friendly messages.   **Related Requirements:**   * **FR-201:** The system must allow recruiters to view all available job packages. * **FR-202:** The system must integrate with VNPay for online payment. * **FR-203:** The system must validate payment signatures returned from VNPay. * **FR-204:** The system must store payment transactions in PaymentsDAO. * **FR-205:** The system must assign purchased packages to the recruiter after a successful payment. * **FR-206:** The system must allow recruiters to view purchased packages and remaining balance.   **Failure Handling:**   * If VNPay is unreachable → system shows: “VNPay service is temporarily unavailable.” * If system cannot verify payment → mark payment as failed and show warning. * If database fails at any step → system logs error and stops the process safely. * If browser closes during payment → order stays in “Pending” for later verification. | | |
| Assumptions: | **ASM-1:** VNPay API is available and functioning.  **ASM-2:** Recruiter has a valid payment method.  **ASM-3:** System server time is synchronized with VNPay time (required for signature validation).  **ASM-4:** Recruiter’s internet connection is stable during checkout. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR1 | Valid Recruiter Role | Only authenticated recruiters can purchase job packages. |
| BR2 | Fixed Package Pricing | Package prices must match the pricing stored in **JobPackagesDAO** at the time of purchase. |
| BR3 | Order Creation Requirement | A payment order must exist before redirecting to VNPay. |
| BR4 | Secure Payment Signature | VNPay responses must be validated using VNPayConfig signature rules before processing. |
| BR5 | Successful Payment Rule | A package is assigned only if VNPay reports **Success** and the signature is valid. |
| BR6 | Atomic Package Assignment | Package assignment and payment update must be atomic. If assignment fails, the system flags the order for admin review. |

### 3.2 UC-15\_Manage payment

#### *a. Functional Description*

| ID and Name: | **UC-15 Manage payment** | | |
| --- | --- | --- | --- |
| Created By: | MinhNTQ | Date Created: | 05/10/25 |
| Primary Actor: | Admin, Recruiter | Secondary Actors: | Payment System(VNPAY) |
| Description: | Administrators and recruiters view and manage payment transactions related to job posting services or premium features: track payment history, verify payment status from VNPay, and download invoices/receipts. | | |
| Trigger: | User (Admin or Recruiter) opens the payment management page or system receives VNPay payment callback | | |
| Preconditions: | PRE-1. Actor is authenticated and authorized (Admin or Recruiter).  PRE-2. Payment records exist in PaymentsDAO/PaymentDetailsDAO. | | |
| Postconditions: | POST-1: Payment status updated/verified in PaymentsDAO.  POST-2: Invoice/receipt available for download.  POST-3: Access logs and payment history viewable by actor. | | |
| Normal Flow: | **3.0 Manage payment**   1. Actor opens Payment Management page 2. System displays transaction list (amount, package, VNPay transaction ID, date, payer) 3. Actor applies search 4. System returns searched list | | |
| Alternative Flows: | **3.1. No transactions found**   1. At step 2, if the system finds no payment transactions, 2. The system displays an empty state message: “No transactions found.” 3. The use case continues normally and allows search actions.   **3.2 No results for the search filter**   1. During step 4, if the search criteria match no records, 2. The system returns an empty list. 3. The system displays a message: “No transactions match your search criteria.”   **3.3 Actor clears filter**   1. After applying a search, the actor clicks **Clear Filters**. 2. The system reloads all transactions and displays the full list. | | |
| Exceptions: | **3.0.E1 PaymentDAO retrieval failure**   1. In step 2, the system fails to fetch data due to database error. 2. The system displays an error message: “Unable to load transactions. Please try again later.” 3. The system logs the error for admin review.   **3.1.E2 Search query invalid (malformed input)**   1. Actor enters an invalid search term (e.g., incorrect date format). 2. The system displays a validation error message: “Invalid search value. Please check your input.” 3. No search is performed until corrected.   **3.2.E3 Session timeout**   1. Actor attempts to search after session expiration. 2. The system redirects actor to the login page. 3. No transaction data is displayed until login is restored. | | |
| Priority: | High | | |
| Business Rules: | BR1 - BR5 | | |
| Other Information: | **Quality Attributes:**   1. The system must display the list of transactions within **2 seconds**. 2. All search operations must return results within **1 second** under normal load. 3. The Payment Management interface must clearly show essential payment information (amount, date, transaction ID). 4. The system must follow secure read-access rules to protect financial information.   **Related Requirements:**   * **FR-301:** The system must allow viewing all payment transactions. * **FR-302:** The system must support searching by date, payer name, amount, and transaction ID. * **FR-303:** The system must display VNPay transaction identifiers for reference. * **FR-304:** The system must handle empty or invalid search responses gracefully. | | |

#### *b. Business Rules*

Provide the business rules those are applied only to the use case

| **ID** | **Business Rule** | **Business Rule Description** |
| --- | --- | --- |
| BR1 | Authorized Access | Only authenticated recruiters or admins may access the Payment Management page. |
| BR2 | Valid Search Input | Search filters (date, VNPay transaction ID, payer) must follow valid formats before submitting. |
| BR3 | Transaction Integrity | All transactions displayed must come directly from PaymentsDAO and must not be altered by the UI. |
| BR4 | Sorting Consistency | The transaction list must be sorted by date (newest first) unless the actor applies a different sort. |
| BR5 | Successful Payment Rule | A package is assigned only if VNPay reports **Success** and the signature is valid. |

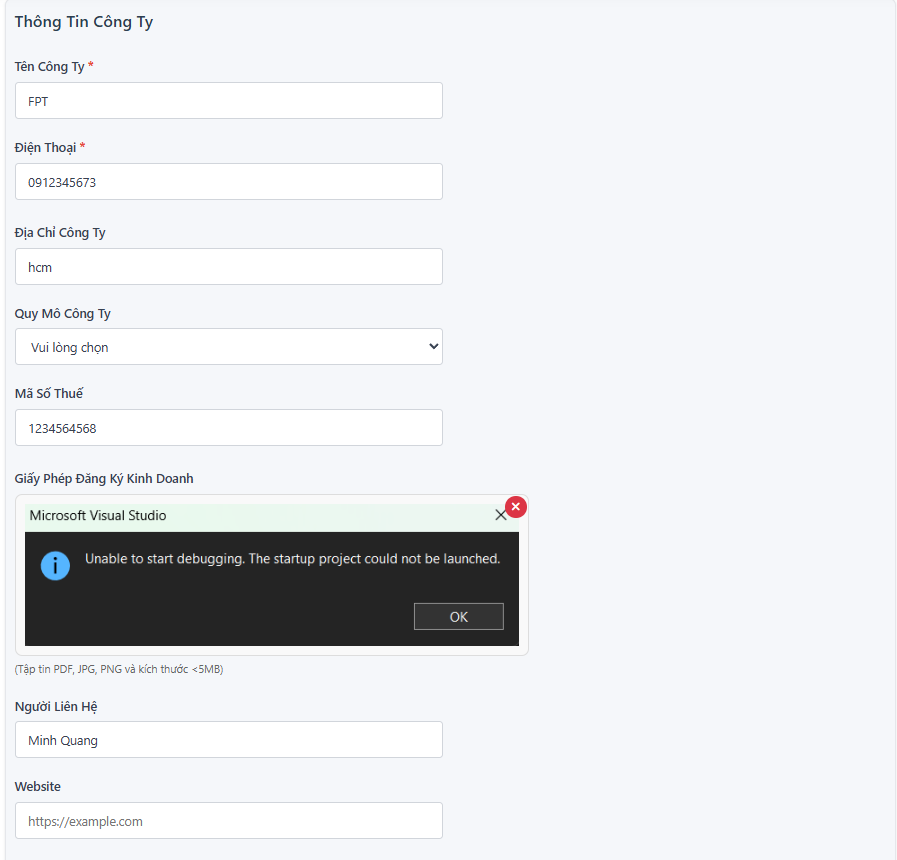
# 

# III. Design Specifications

## 1. Company Management

### 1.1 Update company information

#### *a. Company Information Screen*

**

**

##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Company Name\* | Text Box | This is for the user to input the full name of the company. **Mandatory** field. |
| Phone\* | Text Box | This is for the user to input the company's contact phone number. **Mandatory** field. Input should be numeric. |
| Company Address | Text Box | This is for the user to input the company's address (e.g., location, headquarters). |
| Company Size | Dropdown | This allows the user to select the company's size from a predefined list of options. |
| Tax Code | Text Box | This is for the user to input the company's tax ID number. Input should be numeric/alphanumeric. |
| Business Registration Certificate | File Upload Control | This allows the user to upload the business registration license file (PDF, JPG, PNG, max size <5MB). |
| Contact Person | Text Box | This is for the user to input the name of the main contact person. |
| Website | Text Box | This is for the user to input the company's official website URL. |
| Company Benefits\* | Text Box | This is for the user to input the company's benefits (e.g., one benefit per line). **Mandatory** field. |
| Company Profile | Text Box | This is for the user to input a brief description/introduction of the company. Input limit is **10,000 characters**. |
| Logo | File Upload Control (Drag & Drop) | Allows the user to upload the company's logo. File extensions accepted: .png, .jpg, .jpeg, .gif. Max size: **<1MB**. |
| Company Image | Multi File Upload Control (Drag & Drop) | Allows the user to upload company images. Maximum: **5 files**. File extensions accepted: .png, .jpg, .jpeg, .gif. Max size: **<1MB per file**. |
| Company Video | Text Box | This is for the user to input/paste a **YouTube link** related to the company. |
| Save | Buuton | Users who click on this button will save company information into the system. |

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Company | CRU | Stores all primary text fields and non-media data: Name, Phone, Address, Tax ID, Contact Person, Website, Benefits, Overview, Video URL, and ScaleID. |
| Company\_Scale\_Lookup | R | Provides the predefined options for the **Quy Mô Công Ty** (Company Size) dropdown list. |
| Company\_Assets | CRD | Stores the file paths/URLs and metadata for all uploaded media: **Giấy Phép Kinh Doanh**, **Logo**, and **Hình Ảnh Công Ty**. |

***SQL Commands***

***1.***Stores the main details of the company, including Name, Phone, Address, Tax ID, and Contact Info.

*SELECT \* FROM Recruiter WHERE RecruiterID = ?*

*UPDATE Recruiter SET CompanyName = ?, Phone = ?, CompanyAddress = ?, "*

*+ "CompanySize = ?, ContactPerson = ?, CompanyBenefits = ?, "*

*+ "CompanyDescription = ?, CompanyVideoURL = ?, Website = ?, "*

*+ "CompanyLogoURL = ?, Img = ?, Taxcode = ?, RegistrationCert = ?*

*WHERE RecruiterID = ?*

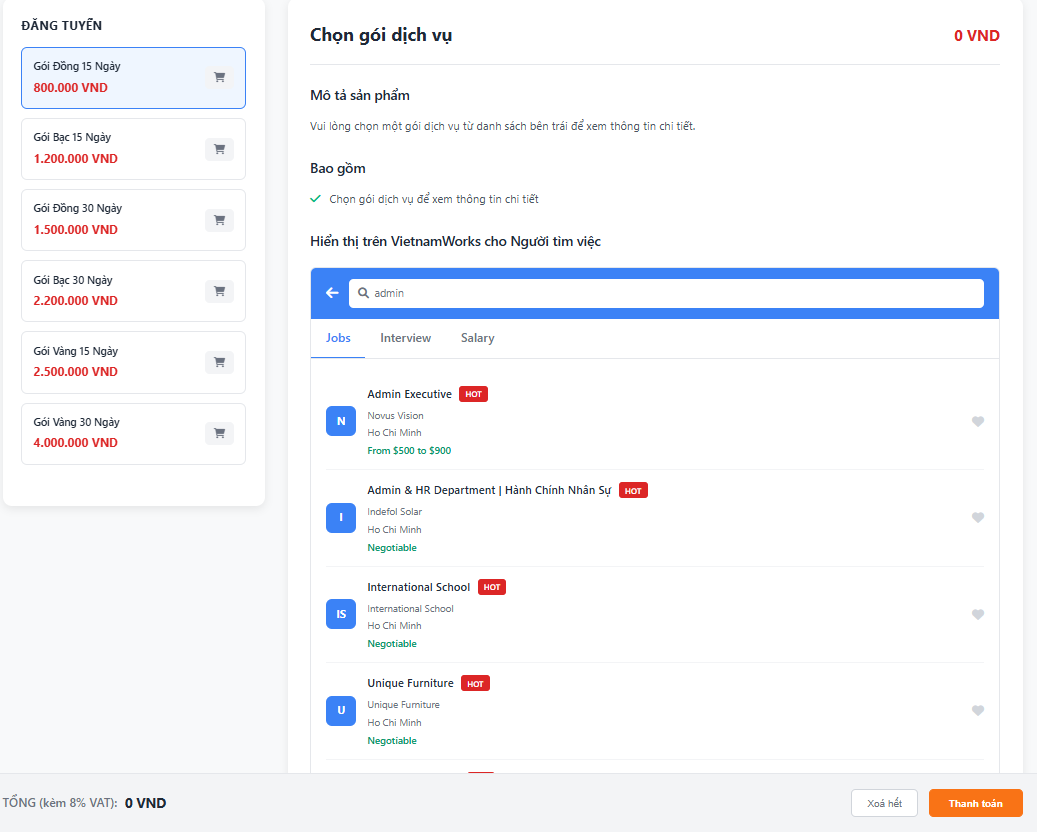
*2.*Stores the file paths/URLs and metadata for all uploaded media: **Giấy Phép Kinh Doanh**, **Logo**, and **Hình Ảnh Công Ty**.

UPDATE Recruiter SET Img = ? WHERE RecruiterID = ?

## 2. Job Package Management

2.1 View packages

a.Package List Screen



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| 15 Day Bronze Package | Button/List Item | Option to select the 15-day Bronze package. Price: 800,000 VND. |
| 15-Day Silver Package | Button/List Item | Option to select the 15-day Silver package. Price: 1,200,000 VND. |
| 30 Day Bronze Package | Button/List Item | Option to select the 30-day Bronze package. Price: 1,500,000 VND. |
| 30 Day Silver Package | Button/List Item | Option to select the 30-day Silver package. Price: 2,200,000 VND. |
| 15 Day Gold Package | Button/List Item | Option to select the 15-day Gold package. Price: 2,500,000 VND. |
| 30 Day Gold Package | Button/List Item | Option to select the 30-day Gold package. Price: 4,000,000 VND. |
| Product Description | Text/Display Area | Displays the detailed features included in the currently selected service package. |
| Delete all | Button | User click on this button will delete all packages added in cart |
| Pay | Button | If the user clicks on this button, the system will directly transfer information to the payment page. |

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Service\_Package | R | Query and display all available recruitment service packages, their prices, and features to populate the list items (Gói Đồng, Gói Bạc, Gói Vàng). |
| Transaction\_Order | C | Create a new order record when the user clicks **Thanh Toán**. Records the total amount and payment status (Pending). |
| Order\_Detail | C | Records the specific package(s) purchased in the current transaction (e.g., PackageID, quantity). |

***SQL Commands***

1.Query and display all available recruitment service packages, their prices, and features to populate the list items (Gói Đồng, Gói Bạc, Gói Vàng).

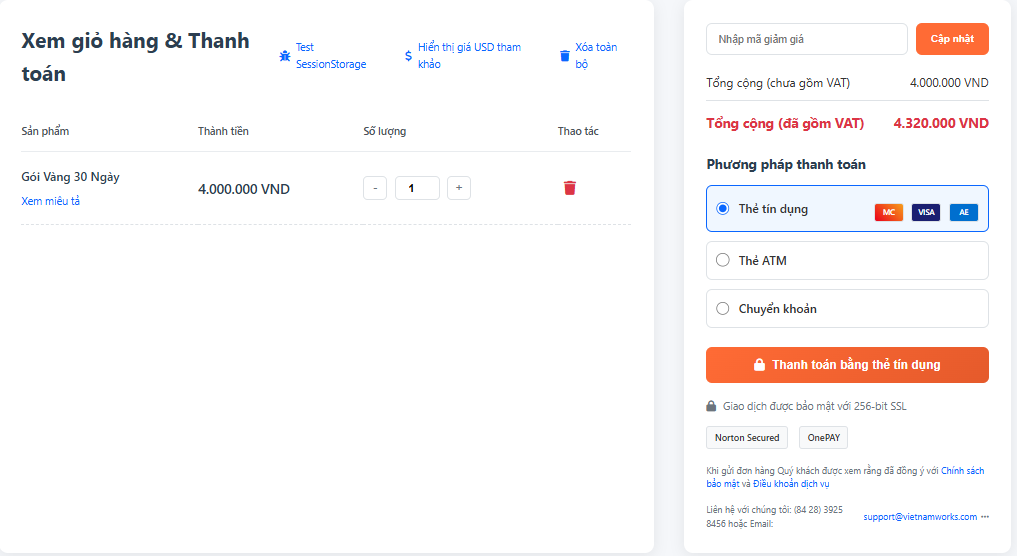
SELECT PackageID, PackageName, PackageType, Description, Price, Duration, Points, Features, IsActive, CreatedAt, UpdatedAt

FROM JobPackages

WHERE IsActive = 1

2.2 View packages

#### *a. Shop-cart*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| 30 Day Gold Package | Text | Displays the name of the service package that the user has added to the cart. |
| Total amount | Text | Displays the amount to be paid for the product (quantity × unit price). |
| Quantity | Number Input | Number field for user to change product quantity. |
| Decrease quantity (-) | Button | When the user presses, the system decreases the number by 1 (minimum = 1). |
| Increase quantity (+) | Button | When pressed, the system increases the product quantity by 1. |
| Delete product | Icon | Trash icon – User clicks to remove product from cart. |
| Total (excluding VAT) | Text | Displays the estimated amount before VAT. |
| Total (including VAT) | Text | Show final amount after VAT – prominent display. |
| Pay | Button | If the user clicks on this button, the system will directly transfer information to the payment page. |
| Credit card | Radio Button | Select credit card payment method. |
| ATM | Radio Button | Choose to pay by domestic ATM card. |
| Transfer | Radio Button | Lựa chọn thanh toán bằng chuyển khoản ngân hàng. |

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Cart | RUD | Get the product list, update quantities, or remove products from the cart. |
| Product | R | Get product details (name, price, description). |
| Order | C | Create an order before proceeding to checkout. |
| PaymentTransaction | C | Record payment transaction information. |

***SQL Commands***

1.Create an order before proceeding to checkout.

INSERT INTO Payments (RecruiterID, Amount, PaymentMethod, PaymentStatus, TransactionCode, PaymentDate, Notes)

VALUES (?, ?, ?, ?, ?, ?, ?)

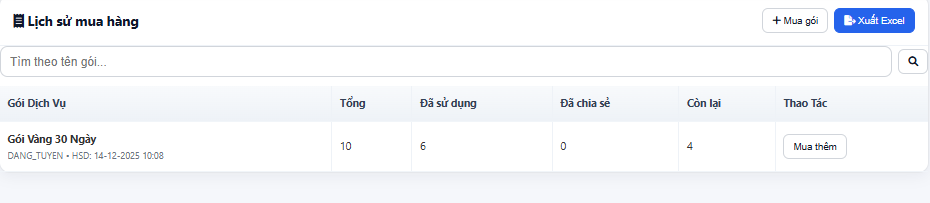
2.Record payment transaction information.

INSERT INTO PaymentDetails (PaymentID, PackageID, Quantity, UnitPrice)

VALUES (?, ?, ?, ?)

2.3 View purchased packages

#### *a. Purchased Packages Dashboard*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Search by package name | Text Box | Users enter the service package name to search in purchase history. |
| Search button (magnifying glass icon) | Button/Icon | Perform search by user entered keyword. |
| Buy a package | Button | Navigate to the new service package purchase page. |
| Pay | Button | If the user clicks on this button, the system will directly transfer information to the payment page. |

##### 

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| PurchaseHistory | R | Get a list of purchase history to display. |

***SQL Commands***

1.Get a list of purchase history to display.

SELECT rp.\*, jp.PackageName, jp.PackageType, jp.Description, jp.Duration, jp.Price

FROM RecruiterPackages rp

INNER JOIN JobPackages jp ON rp.PackageID = jp.PackageID

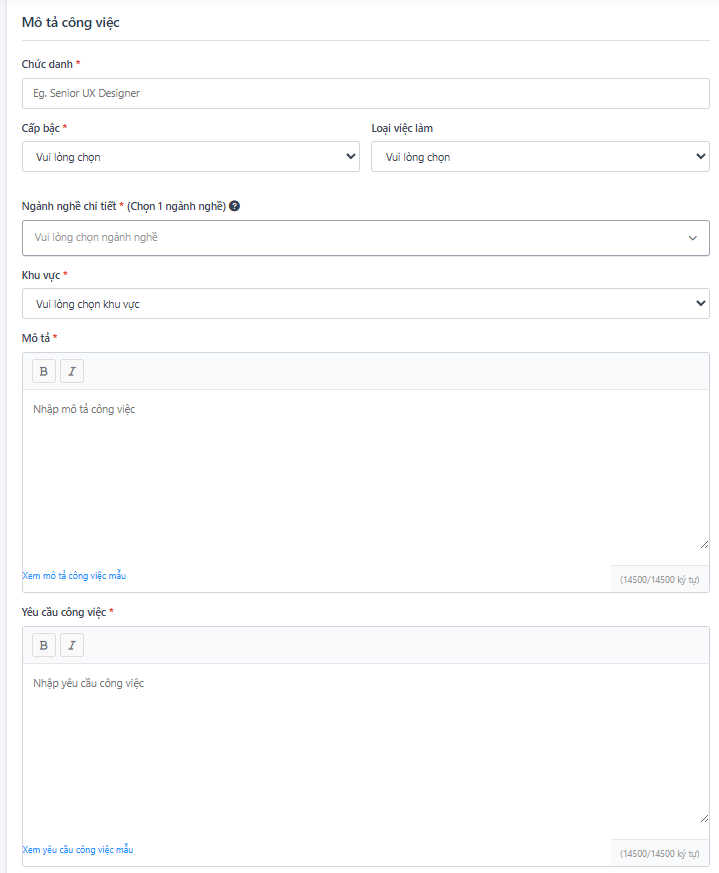
WHERE rp.RecruiterID = ?

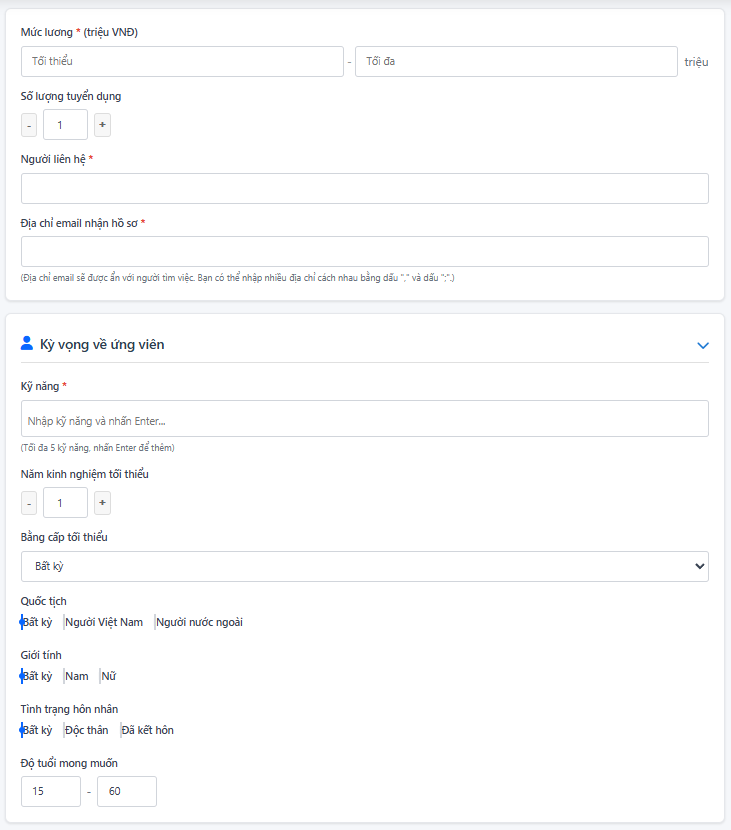
ORDER BY rp.PurchaseDate DESC

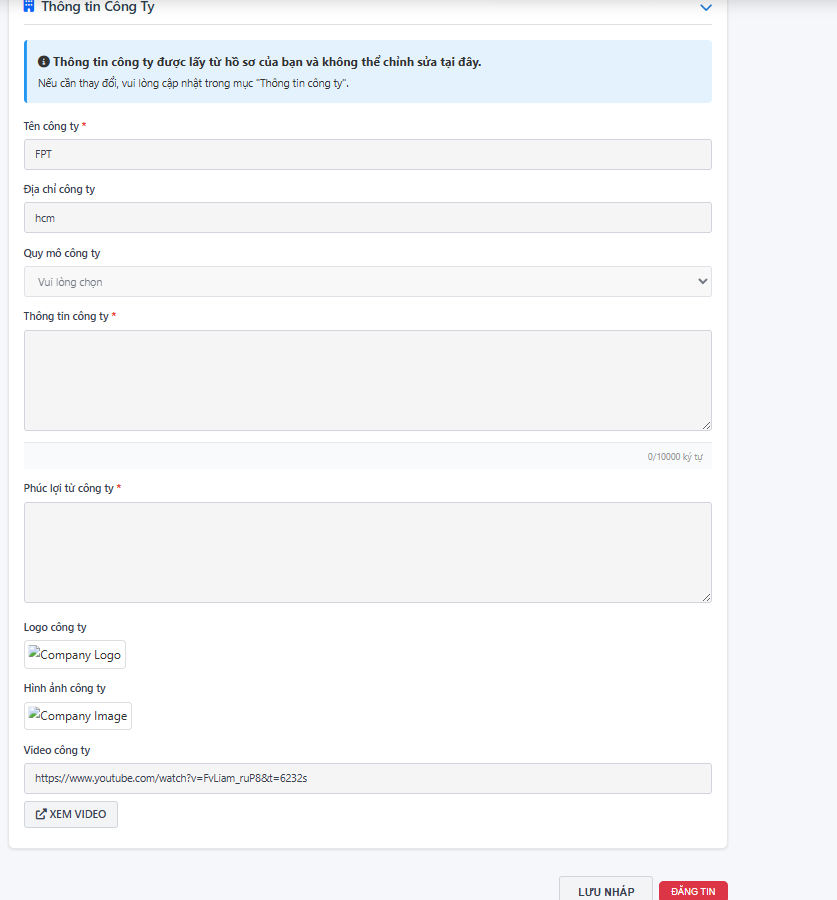
## 3. Job Posting Management

### 3.1 Create job

#### *a. Create Job Screen*







##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Chức danh\* | Text Box | User enters the job title to be recruited |
| Cấp bậc\* | Dropdown | The user selects one of the levels. |
| Loại việc làm\* | Dropdown | Users select one of the current job types |
| Ngành nghề chi tiết \* | Dropdown | Users select jobs to recruit |
| Khu vực \* | Dropdown | User selects the area to recruit |
| Mô tả \* | Text Box | User enters job description |
| Yêu cầu công việc\* | Text Box | User enters details of job requirements |
| Mức lương\* | Number | Minimum and maximum salary a candidate can receive |
| Số lượng ứng tuyển | Number (stepper) | Number of people who can apply |
| Người liên hệ\* | Text Box | Name of the person handling the application |
| Địa chỉ email nhận hồ sơ\* | Text Box | Email to receive application documents |
| Kỹ năng\* | Tag Input | Enter up to 5 skills, press Enter to add. |
| Năm kinh nghiệm tối thiểu | Number | Minimum number of years of experience required by employers |
| Bằng cấp tối thiểu | Dropdown | User selects minimum qualification that candidate must achieve |
| Quốc tịch | Check Box | Candidate's nationality |
| Giới tính | Check Box | Candidate's gender |
| Tình trạng hôn nhân | Check Box | Any / Male / Female. |
| Độ tuổi mong muốn | Number | Minimum and maximum age required by employers |
| Lưu nháp | Button | Save a draft of the job posting. |
| Đăng tin | Button | Post job ads. |

##### 

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| JobPost | CRU | Save and update recruitment information. |
| Company | R | Get company profile data. |
| Skills | CR | Get and create skills |
| JobCategories | R | Get the list of industries. |
| Locations | R | Get the list of areas. |

***SQL Commands***

1.Save and update recruitment information.

INSERT INTO Jobs (RecruiterID, JobTitle, Description, Requirements,

JobLevelID, LocationID, SalaryRange, PostingDate, ExpirationDate, CategoryID,

AgeRequirement, Status, JobTypeID, HiringCount, ViewCount, IsUrgent,

IsPriority, PriorityExpiryDate, ContactPerson, ApplicationEmail,

MinExperience, Nationality, Gender, MaritalStatus,

AgeMin, AgeMax, JobCode, CertificatesID)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

UPDATE Jobs SET JobTitle = ?, Description = ?, Requirements = ?,

JobLevelID = ?, LocationID = ?, SalaryRange = ?, ExpirationDate = ?,

CategoryID = ?, AgeRequirement = ?, Status = ?, JobTypeID = ?,

HiringCount = ?, ContactPerson = ?, ApplicationEmail = ?,

MinExperience = ?, Nationality = ?, Gender = ?, MaritalStatus = ?,

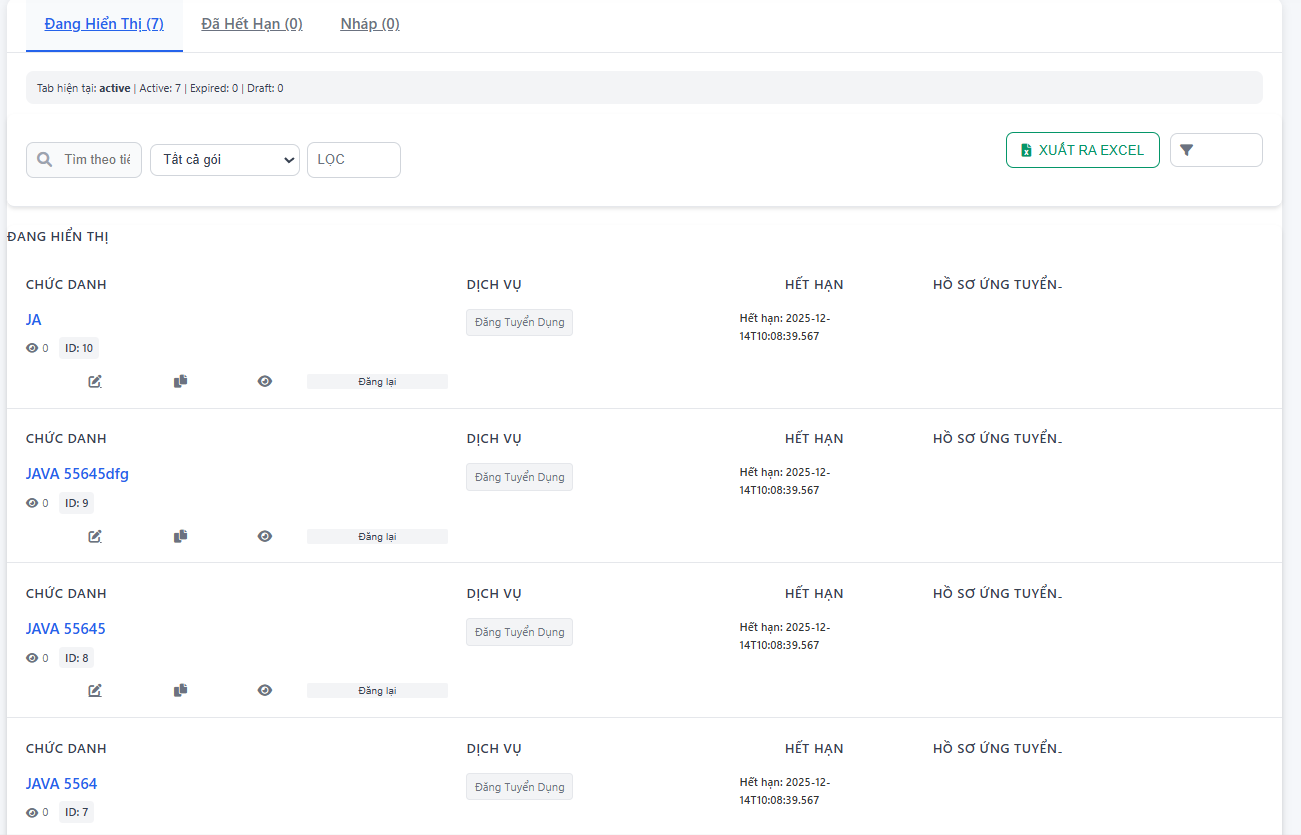
AgeMin = ?, AgeMax = ?, CertificatesID = ?

WHERE JobID = ?

2.Get and create skills  
INSERT INTO JobSkillMappings (JobID, SkillID) VALUES (?, ?)

### 3.2 Manage job list

#### *a. Job List Screen*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Search by Title | Text Box | Search by job title. |
| Package Filter | Dropdown | Filter by service package (All packages / package A / package B…). |
| Filter Button | Button | Filter according to selection criteria. |

##### 

##### **Database Access**

*[Provide the design description for the screen/function to access the database here: what table the screen/function would access, which transactions does it make (C-Create, R-Read, U-Update, or D-Delete), and how/purpose of the access (by providing Description and SQL commands)]*

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| JobPost | R | Get list of posts by tab (Active / Expired / Draft). |
| JobServicePackage | R | Get service package information associated with the message. |
| JobCandidate | R | Count the number of applications by post ID. |

***SQL Commands***

1.Get list of posts by tab (Active / Expired / Draft).

SELECT \* FROM Jobs

WHERE RecruiterID = ?

AND (LOWER(Status) = 'active' OR LOWER(Status) = 'published')

AND (ExpirationDate IS NULL OR ExpirationDate > GETDATE())

ORDER BY PostingDate DESC

SELECT \* FROM Jobs

WHERE RecruiterID = ?

AND (LOWER(Status) = 'active' OR LOWER(Status) = 'published')

AND ExpirationDate IS NOT NULL AND ExpirationDate <= GETDATE()

ORDER BY ExpirationDate DESC

SELECT \* FROM Jobs

WHERE RecruiterID = ?

AND LOWER(Status) IN ('draft','draw')

ORDER BY PostingDate DESC

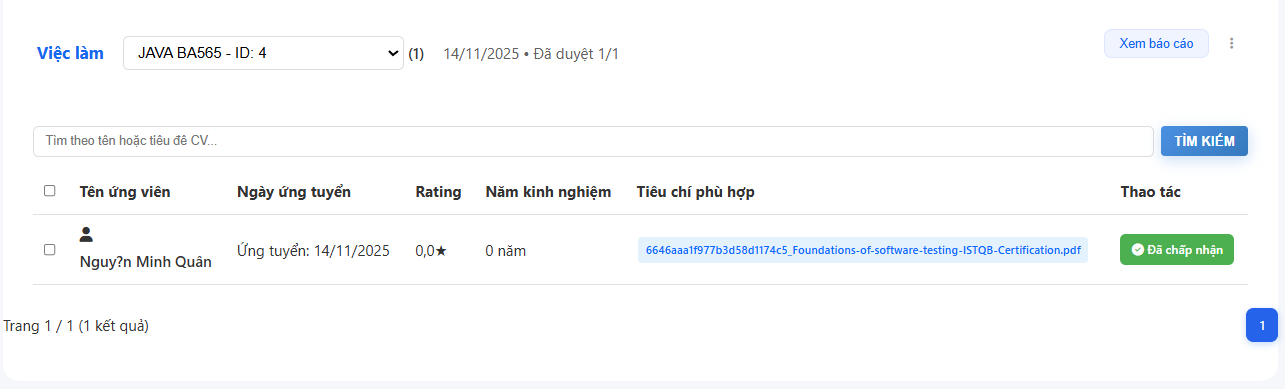
2.Count the number of applications by post ID.

SELECT COUNT(\*) FROM Applications WHERE JobID = ?

## 4. Application Management

### 4.1 View candidate applications

#### *a. Application List Screen*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Job Selector | Dropdown | Displays a list of job postings. Users select a posting to view a list of candidates. |
| Job Summary | Label | Summary information: date posted, number of approved candidates / total candidates. |
| More Options | Button | Accept / Reject Candidate |
| Candidate Name | Hyperlink | Display candidate information |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| JobPost | R | Get the list of job postings in the dropdown. |
| JobCandidate | R | Get list of candidates by JobID. |
| JobCandidate | U | Update candidate status (accepted/rejected). |
| Candidate | R | Get applicant information. |

***SQL Commands:***

1.Get the list of job postings in the dropdown.

SELECT \* FROM Jobs

WHERE RecruiterID = ? AND Status = 'Published'

ORDER BY PostingDate DESC

2.Get list of candidates by JobID.

SELECT

a.ApplicationID, a.JobID, a.CVID, a.ApplicationDate, a.Status,

js.JobSeekerID, js.FullName as CandidateName, js.Email as CandidateEmail,

js.Phone as CandidatePhone, cv.CVTitle, j.JobTitle,

0 as ExperienceYears, 0.0 as Rating

FROM Applications a

INNER JOIN CVs cv ON a.CVID = cv.CVID

INNER JOIN JobSeeker js ON cv.JobSeekerID = js.JobSeekerID

INNER JOIN Jobs j ON a.JobID = j.JobID

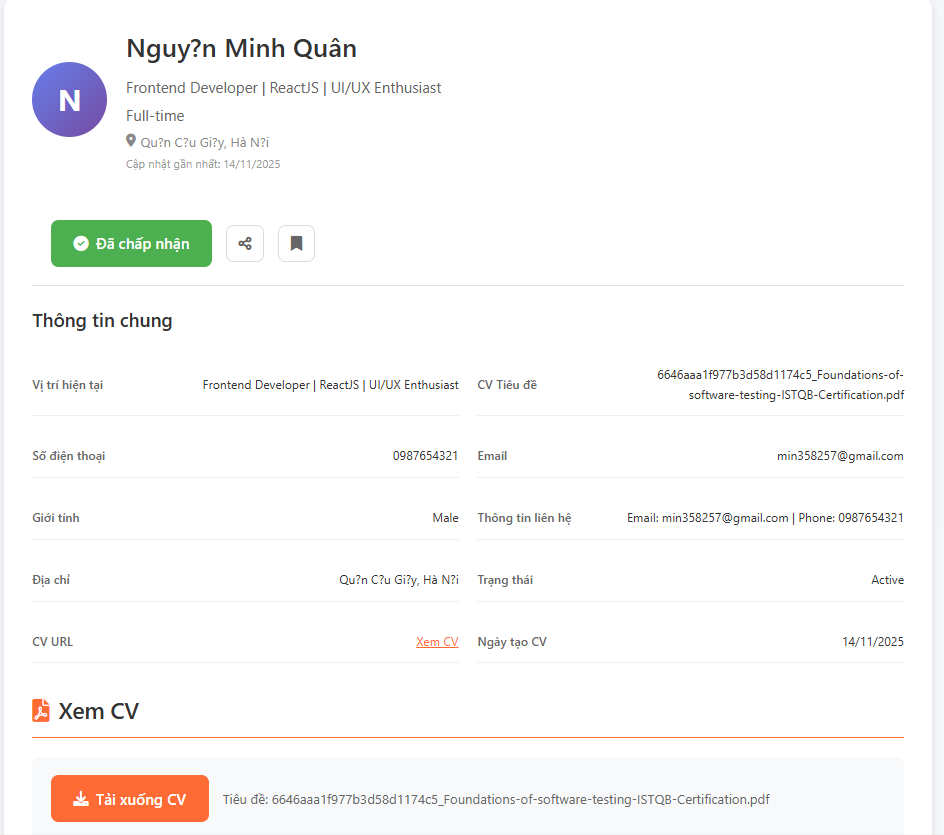
WHERE a.JobID = ? AND j.RecruiterID = ?

ORDER BY a.ApplicationDate DESC

3.Update candidate status (accepted/rejected).

UPDATE Applications SET Status = ? WHERE ApplicationID = ?

#### *b. Candidate Detail*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Avatar | Icon Circle | Displays the candidate's initials or real avatar. |
| FullName | Label | Candidate name. |
| Position Summary | Label | Quick description lines: current position / main skills. |
| Employment Type | Label | Full-time / Part-time / Freelance. |
| Location | Label | Candidate address. |
| Status Button | Button | For example: “Accepted”. Indicates the candidate status. |
| Current Position | Text | Candidate's current job position. |
| CV Title | Text | Original CV file name. |
| Phone | Text | Candidate phone number. |
| Email | Text | Candidate email. |
| Gender | Text | Sex. |
| Contact Info | Text | Contact information summary. |
| Address | Text | Residential address. |
| Status | Text | Active / Inactive / Pending. |
| CV URL | Hyperlink | Link “View CV”. |
| CV Upload Date | Date | Date of uploading CV to the system. |
| CV Preview Header | Label | Title “View CV”. |
| Download CV Button | Button | Download CV file to your computer. |
| CV File Name | Label | CV file name (.pdf). |
| Inline PDF Viewer | PDF Frame | Display CV directly on the page. |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Candidate | R | Get all candidate information. |
| CandidateCV | R | Get information & CV file. |
| JobCandidate | R | Get application status (accepted/rejected/pending). |
| JobCandidate | U | Update candidate status. |

***SQL Commands:***

1.Get all candidate information.

SELECT \* FROM JobSeeker WHERE JobSeekerID = ?

2.Get information & CV file.

SELECT CVID, JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate

FROM CVs

WHERE JobSeekerID = ?

SELECT CVID, JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate

FROM CVs

WHERE CVID = ?

3,Update candidate status.

UPDATE Applications SET Status = ? WHERE ApplicationID = ?

### 

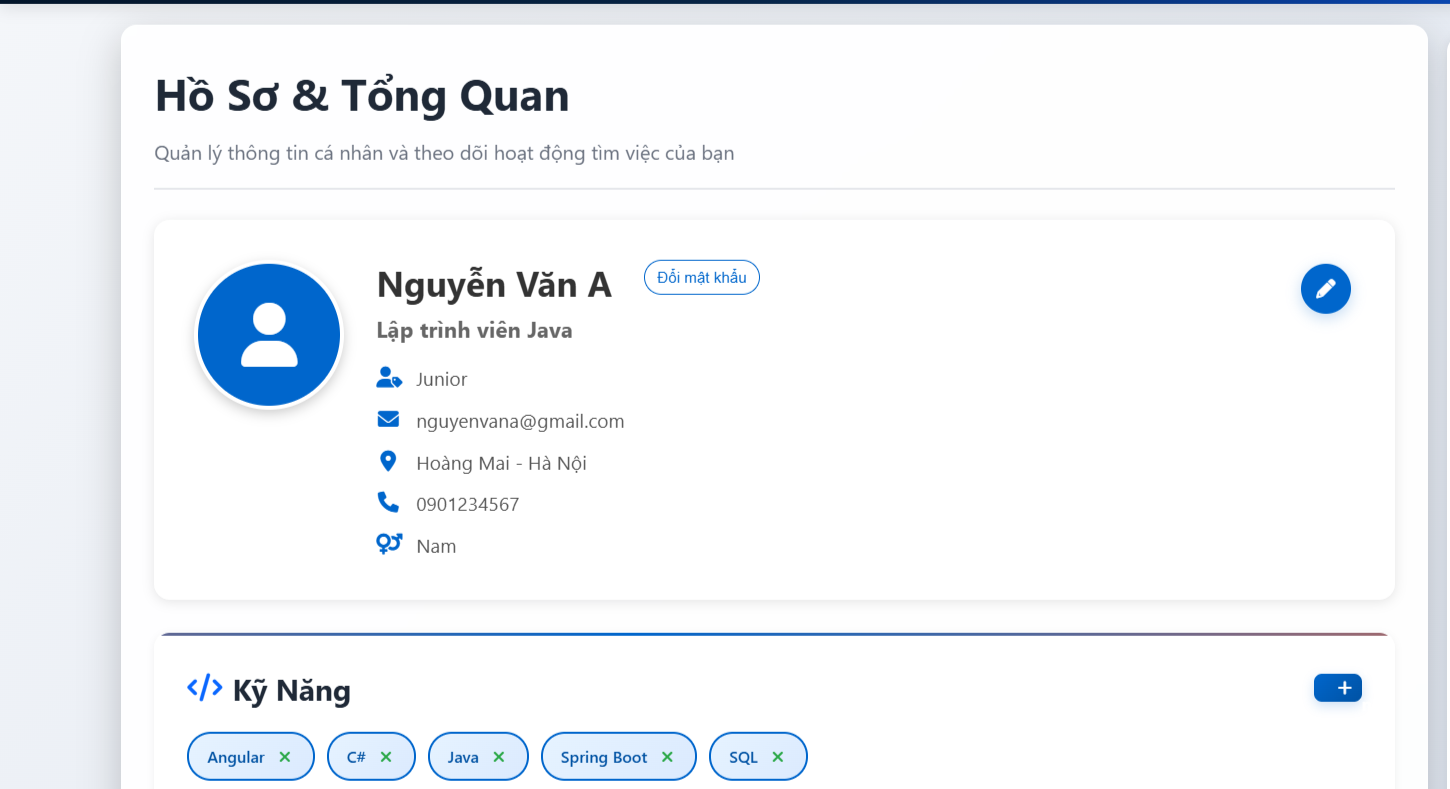
## 5. Profile Management

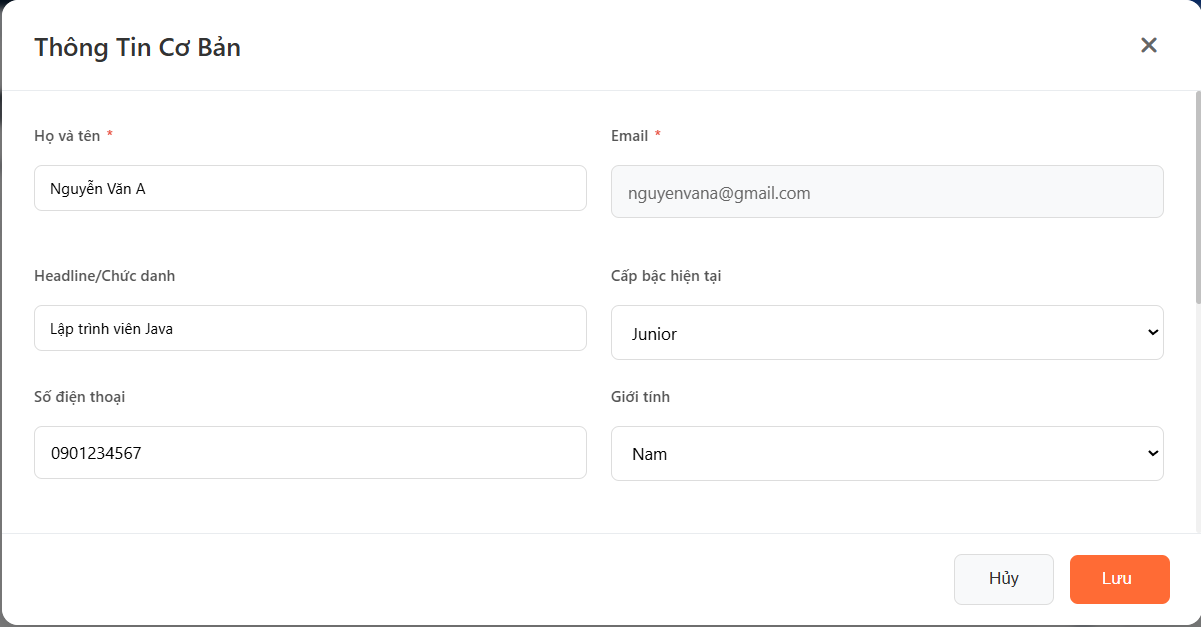
### 5.1 Manage job seeker profile

#### *a. Update information*

##### **UI Design**

* [UC06\_Manage jobseeker profile](#_h2xksjvozg4w)





| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Full Name \* | Text Box | User's full name. Only accepts letters and spaces, maximum 100 characters. Validates using regex: ^[\p{L}\s]+$ |
| Email\* | Text Box (Read-only) | User's email address. This field is read-only and cannot be modified. |
| Phone | Text Box | User's phone number. Must be 10-11 digits. Validates using regex: ^\d{10,11}$ |
| Gender | Dropdown | User's gender. Options: Male, Female, Other |
| Headline | Text Area | Professional headline or job title. Maximum 200 characters. |
| Contact Info | Text Area | Additional contact information. Maximum 255 characters. |
| Address | Text Box | Detailed address. Maximum 255 characters. |
| Location | Dropdown | Select location/city from predefined list. References Location table. |
| Current Level | Dropdown | Select current job level/position. References Type table (JobLevel). |
| Status | Dropdown | Account status. Options: Active, Inactive |
| Avatar Image | Image Upload | Profile picture. Supports image file upload with preview. |
| **Data Table** | | |
| Email | VARCHAR(255) | User's unique email address |
| Password | VARCHAR(255) | User's hashed password (MD5) |
| FullName | VARCHAR(100) | User's full name |
| Phone | VARCHAR(15) | User's phone number |
| Gender | VARCHAR(20) | User's gender |
| Headline | VARCHAR(200) | Professional headline |
| ContactInfo | VARCHAR(255) | Additional contact information |
| Address | VARCHAR(255) | Detailed address |
| LocationID | Integer (FK) | Foreign key reference to Location table |
| Img | VARCHAR(255) | Path to profile image file |
| CurrentLevelID | Integer (FK) | Foreign key reference to Type table (Job Level) |
| Status | VARCHAR(50) | Account status (Active/Inactive) |
| **Data Actions** | | |
| Edit Profile | Edit Icon (Pencil) | Opens modal dialog with profile edit form. Displays as circular button with icon. |
| Save Changes | Button | Validates all input fields and submits form to update profile information in database. |
| Cancel | Button | Closes the edit modal without saving changes. Returns to profile view. |
| Change Password | Pill Button | Opens separate modal for password change functionality. |
| Upload Avatar | Upload Button | Allows user to select and upload new profile picture. |
| View Avatar | Click on Avatar | Opens full-size view of profile picture in modal. |
| Remove Avatar | Delete Icon | Removes current profile picture. |

##### **Database Access**

| Table | CRUD | Description |
| --- | --- | --- |
| JobSeeker | R | Query job seeker information by JobSeekerID to display current profile data |
| JobSeeker | U | Update job seeker profile information (FullName, Phone, Gender, Headline, ContactInfo, Address, LocationID, CurrentLevelID, Status) |
| Location | R | Query all available locations/cities for dropdown selection |
| Type | R | Query all job levels (CurrentLevel) for dropdown selection |

***SQL Commands:***

1/ Get Job Seeker profile by ID:

SELECT \* FROM JobSeeker WHERE JobSeekerID = ?

2/ Update Job Seeker profile:

UPDATE JobSeeker

SET Email=?, FullName=?, Phone=?, Gender=?,

Headline=?, ContactInfo=?, Address=?,

LocationID=?, CurrentLevelID=?, Status=?

WHERE JobSeekerID=?

3/ Get all available locations:

SELECT \* FROM Location ORDER BY LocationName

4/ Get all job levels:

SELECT \* FROM Type WHERE Category = 'JobLevel' ORDER BY TypeName

5/ Get location name by ID:

SELECT LocationName FROM Location WHERE LocationID = ?

6/ Get job level name by ID:

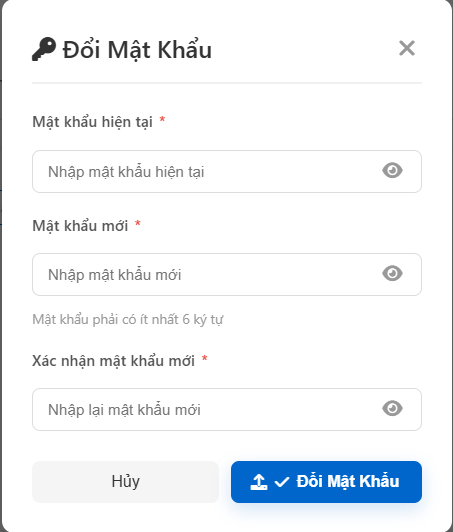
SELECT TypeName FROM Type WHERE TypeID = ?

7/ Update avatar image:

UPDATE JobSeeker SET Img = ? WHERE JobSeekerID = ?

#### *b. Change password*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Current Password\* | Password Input | Current password for verification. Must match user's current password in database. |
| New Password\* | Password Input | New password. Min 6 characters, max 50 characters. Must be different from current password. |
| Confirm Password\* | Password Input | Confirmation of new password. Must match New Password field. |
| ***Data Table*** | | |
| JobSeekerID | Integer | Primary key, user identifier |
| Password | VARCHAR(255) | Hashed password (MD5). Value "GOOGLE\_LOGIN" for Google accounts (cannot change password) |
| ***Data Actions*** | | |
| Toggle Password | Eye Icon | Show/hide password text for each password field |
| Submit | Button | Validates all inputs and updates password in database |
| Cancel | Button | Closes change password modal without saving |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| JobSeeker | R | SELECT \* FROM JobSeeker WHERE JobSeekerID = ? - Verify current user and password |
| JobSeeker | U | UPDATE JobSeeker SET Password = ? WHERE JobSeekerID = ? - Update password with MD5 hash |

***SQL Commands:***

**1/ Query user information and verify current password:**

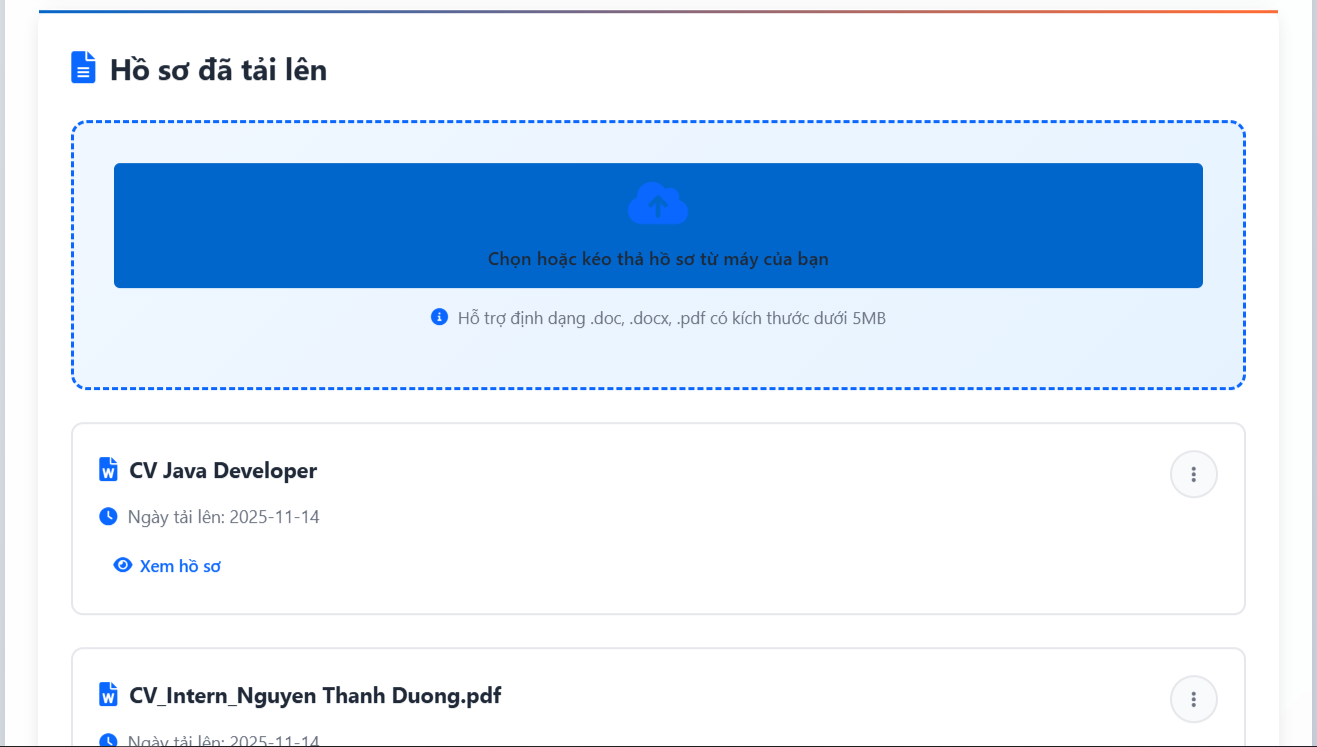
SELECT \* FROM JobSeeker WHERE JobSeekerID = ?

**2/ Update password:**

UPDATE JobSeeker SET Password = ? WHERE JobSeekerID = ?

#### *c. Manage CV*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| CV File\* | File Upload | Upload CV file. Accepts .pdf, .doc, .docx formats. Max file size: 5MB. |
| CV Title | Text (Auto) | Original filename is automatically used as CV title |
| Is Active | Toggle/Checkbox | Toggle CV visibility for recruiters. Active CVs are searchable by employers. |
| CV Title | Text | Original filename of uploaded CV |
| CVURL | Link | Path to CV file for download/view |
| Creation Date | DateTime | Date and time when CV was uploaded |
| Is Active | Status Badge | Shows whether CV is active (visible) or inactive (hidden) |
| ***Data Table*** | | |
| CVID | Integer | Primary key, auto-increment CV identifier |
| JobSeekerID | Integer (FK) | Foreign key reference to JobSeeker table |
| CVTitle | VARCHAR(255) | Original filename of uploaded CV |
| CVContent | TEXT | Optional text content (currently NULL) |
| CVURL | VARCHAR(500) | Relative path to CV file (/uploads/cvs/uniquefilename.ext) |
| IsActive | BIT/BOOLEAN | Status flag (1=Active/Visible, 0=Inactive/Hidden) |
| CreationDate | DATETIME | Timestamp when CV was uploaded |
| ViewCount | Integer | Number of times CV was viewed (optional) |
| ***Data Actions*** | | |
| Upload CV | Upload Button | Opens file selector to choose CV file for upload (PDF, DOC, DOCX only) |
| View/Download CV | Eye/Download Icon | Opens or downloads the CV file |
| Toggle Active Status | Toggle Switch | Enable/disable CV visibility for recruiter searches |
| Delete CV | Delete Icon | Removes CV from database and deletes physical file from server |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| CVs | C | INSERT INTO CVs (JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate) VALUES (?, ?, ?, ?, ?, ?) |
| CVs | R | SELECT \* FROM CVs WHERE JobSeekerID = ? - Get all CVs for a job seeker |
| CVs | R | SELECT \* FROM CVs WHERE CVID = ? - Get specific CV details |
| CVs | U | UPDATE CVs SET IsActive = ? WHERE CVID = ? - Toggle CV active status |
| CVs | D | DELETE FROM CVs WHERE CVID = ? - Remove CV record |
| CVs | R | SELECT COUNT(\*) FROM CVs WHERE CVID = ? AND JobSeekerID = ? - Verify CV ownership |

***SQL Commands:***

**1/ Query user information and verify current password:**

SELECT \* FROM JobSeeker WHERE JobSeekerID = ?

**2/ Update password:**

UPDATE JobSeeker SET Password = ? WHERE JobSeekerID = ?

**1/ Get all CVs for a Job Seeker:**

SELECT CVID, JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate

FROM CVs

WHERE JobSeekerID = ?

**2/ Upload/Insert new CV:**

INSERT INTO CVs (JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate)

VALUES (?, ?, ?, ?, ?, ?)

**3/ Get CV by ID:**

SELECT CVID, JobSeekerID, CVTitle, CVContent, CVURL, IsActive, CreationDate

FROM CVs

WHERE CVID = ?

**4/ Toggle CV active status:**

UPDATE CVs SET IsActive = ? WHERE CVID = ?

**5/ Delete CV:**

DELETE FROM CVs WHERE CVID = ?

**6/ Check CV ownership:**

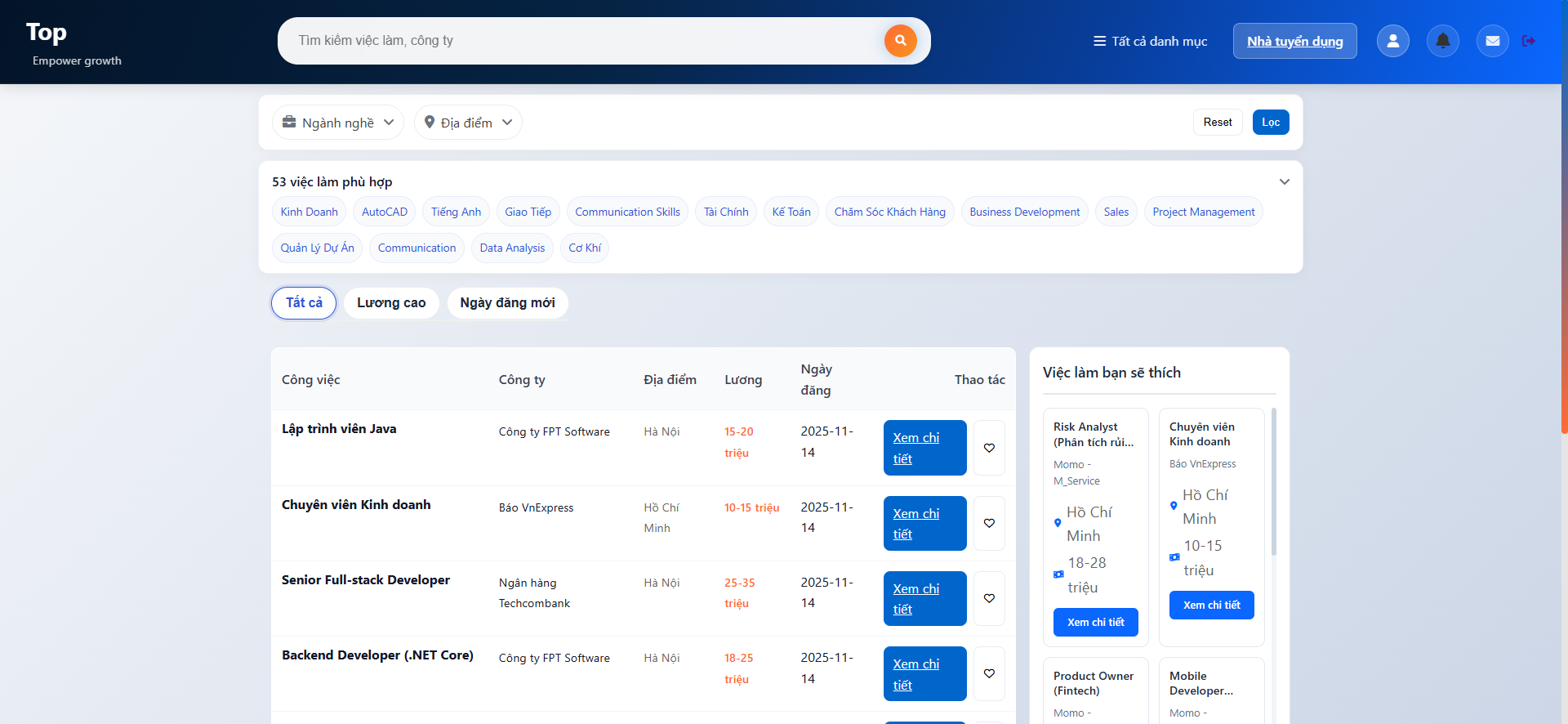
SELECT COUNT(\*) FROM CVs WHERE CVID = ? AND JobSeekerID = ?

## 6. Job management

### 6.1 View job list

#### *a. View job list*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Keyword | Search Box | Search jobs by job title or company name. Trims whitespace and handles empty values. |
| Category Filter | Multi-select Checkbox | Filter jobs by category (multiple selection allowed). Uses CategoryID array. |
| Location Filter | Dropdown | Filter jobs by location/city. Single selection from Location table. |
| Job Title | Text Link | Job title, clickable to view details. Maximum display per job card. |
| Company Name | Text | Recruiter's company name. Retrieved from Recruiter table. |
| Location | Text/Icon | Job location/city name. Retrieved from Location table. |
| Salary Range | Text | Salary range in format "15-20 triệu", "25-35 triệu", etc. |
| Posting Date | Date | Date when job was posted. Format: YYYY-MM-DD |
| Package Badge | Badge/Label | Shows package level: Gold (featured), Silver (highlight), Bronze (standard) |
| ***Data Table*** | | |
| JobID | Integer | Primary key, auto-increment job identifier |
| RecruiterID | Integer (FK) | Foreign key reference to Recruiter table |
| JobTitle | VARCHAR(255) | Job position title |
| Description | TEXT | Detailed job description |
| Requirements | TEXT | Job requirements and qualifications |
| JobLevelID | Integer (FK) | Foreign key to Type table (job level) |
| LocationID | Integer (FK) | Foreign key to Location table |
| SalaryRange | VARCHAR(100) | Salary range text (e.g., "15-20 triệu") |
| PostingDate | DATETIME | Date job was posted |
| ExpirationDate | DATETIME | Date job posting expires |
| CategoryID | Integer (FK) | Job category/industry |
| AgeRequirement | Integer | Age requirement for applicants |
| Status | VARCHAR(50) | Job status (Published, Closed, Draft, Pending) |
| ViewCount | Integer | Number of times job was viewed |
| ***SavedJobs table*** | | |
| SavedJobID | Integer | Primary key |
| JobSeekerID | Integer (FK) | Reference to JobSeeker |
| JobID | Integer (FK) | Reference to Jobs |
| SavedDate | DATETIME | When job was saved |
| ***Data Actions*** | | |
| Search Jobs | Search Button | Submits search form with keyword, categories, and location filters |
| View Job Detail | "Xem chi tiết" Button | Navigates to job detail page with job information |
| Save Job | Heart Icon | Saves job to user's saved jobs list (requires login) |
| Unsave Job | Filled Heart Icon | Removes job from saved jobs list |
| Filter by Category | Checkbox | Apply multiple category filters to job list |
| Filter by Location | Dropdown Selection | Filter jobs by selected location |
| Clear Filters | Clear/Reset Button | Removes all applied filters and shows all published jobs |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Jobs | R | SELECT \* FROM Jobs WHERE Status = 'Published' - Get all published jobs |
| Jobs | R | Search jobs with filters (keyword, category, location) |
| Recruiter | R | JOIN to get CompanyName and CompanyLogoURL |
| Locations | R | JOIN to get LocationName for display |
| SavedJobs | C | INSERT INTO SavedJobs (JobSeekerID, JobID, SavedDate) VALUES (?, ?, GETDATE()) |
| SavedJobs | D | DELETE FROM SavedJobs WHERE JobSeekerID = ? AND JobID = ? |
| SavedJobs | R | SELECT COUNT(\*) FROM SavedJobs WHERE JobSeekerID = ? AND JobID = ? - Check if saved |
| RecruiterPackages | R | JOIN to determine package level (Gold/Silver/Bronze) |

***SQL Commands:***

**1/ Get all published jobs:**

SELECT j.JobID, j.RecruiterID, j.JobTitle, j.Description, j.Requirements,

j.JobLevelID, j.LocationID, j.SalaryRange, j.PostingDate, j.ExpirationDate,

j.CategoryID, j.AgeRequirement, j.Status,

r.CompanyName, l.LocationName

FROM Jobs j

JOIN Recruiter r ON j.RecruiterID = r.RecruiterID

JOIN Locations l ON j.LocationID = l.LocationID

WHERE j.Status = 'Published'

**2/ Search jobs with filters:**

SELECT j.\*, r.CompanyName, l.LocationName

FROM Jobs j

JOIN Recruiter r ON j.RecruiterID = r.RecruiterID

JOIN Locations l ON j.LocationID = l.LocationID

WHERE j.Status = 'Published'

AND (j.JobTitle LIKE ? OR r.CompanyName LIKE ?)

AND j.CategoryID IN (?, ?, ...)

AND j.LocationID = ?

**3/ Save job:**

INSERT INTO SavedJobs (JobSeekerID, JobID, SavedDate)

VALUES (?, ?, GETDATE())

**4/ Unsave job:**

DELETE FROM SavedJobs

WHERE JobSeekerID = ? AND JobID = ?

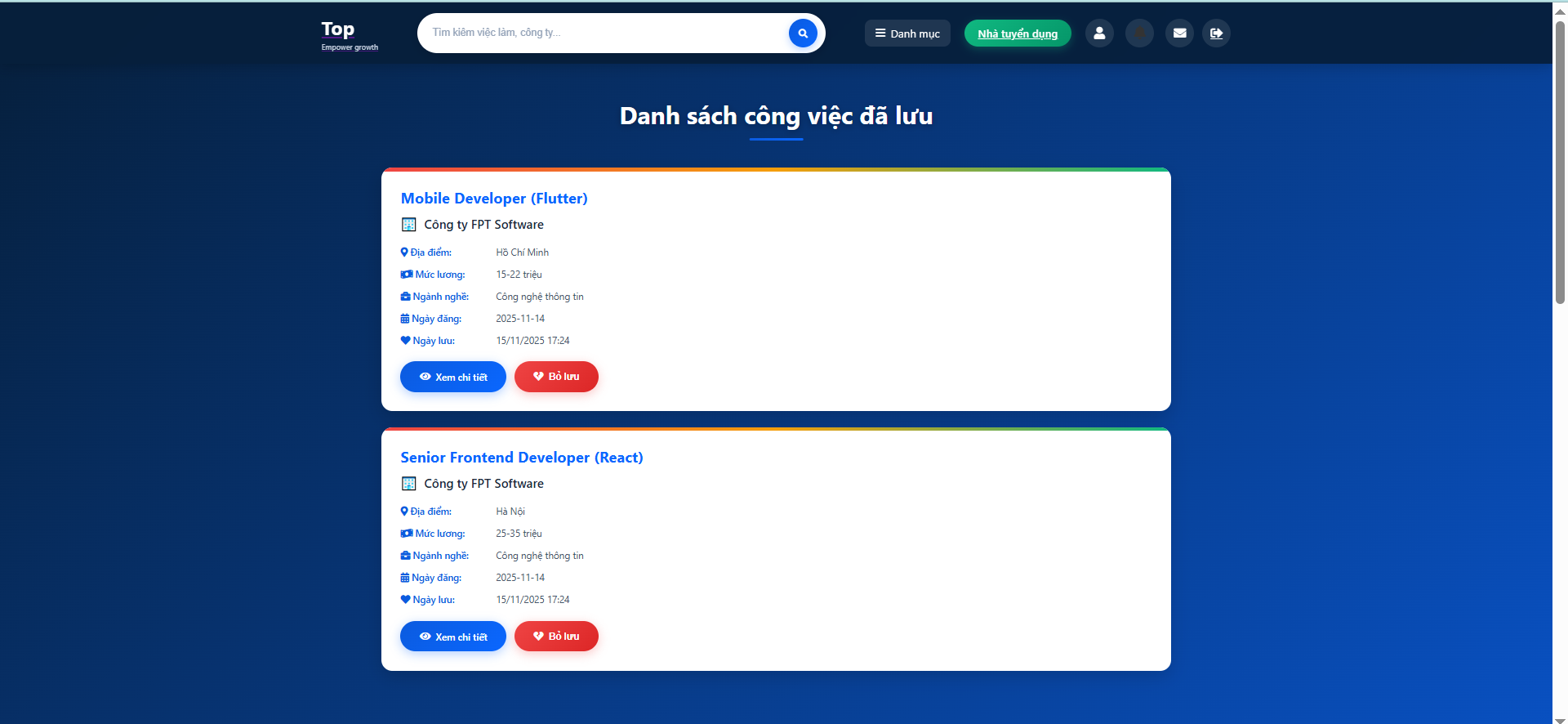
**5/ Check if job is saved:**

SELECT COUNT(\*) FROM SavedJobs

WHERE JobSeekerID = ? AND JobID = ?

#### *b. View saved job*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Job Title | Text Link | Job position title, clickable to view job detail |
| Company Name | Text | Recruiter's company name |
| Location | Text/Icon | Job location from Locations table |
| Salary Range | Text | Salary information (e.g., "15-20 triệu") |
| Industry/Category | Badge | Job category/industry from Categories table |
| Posting Date | Date | Date when job was originally posted |
| Saved Date | DateTime | Date and time when user saved this job. Format: DD/MM/YYYY HH:mm |
| Total Count | Number | Total number of saved jobs displayed at top |
| ***Data Table*** | | |
| SavedJobID | Integer | Primary key, auto-increment identifier |
| JobSeekerID | Integer (FK) | Foreign key reference to JobSeeker table |
| JobID | Integer (FK) | Foreign key reference to Jobs table |
| SavedDate | DATETIME | Timestamp when job was saved. Default: GETDATE() |
| ***Data Actions*** | | |
| View Job Detail | Job Card Click/"View" Button | Navigate to job detail page |
| Unsave Job | Heart Icon/Remove Button | Remove job from saved list. Updates UI and database. |
| Save Job | Heart Icon (from other pages) | Add job to saved jobs list. Requires login. |
| Check Saved Status | API Call | Check if specific job is already saved (via AJAX) |
| Apply to Job | Apply Button | Navigate to job detail page with application option |
| Sort by Date | Sort Option | Sort saved jobs by saved date (default: newest first) |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| SavedJobs | C | INSERT INTO SavedJobs (JobSeekerID, JobID, SavedDate) VALUES (?, ?, GETDATE()) |
| SavedJobs | R | Get all saved jobs for user with JOIN to Jobs, Recruiter, Locations, Categories |
| SavedJobs | R | Check if job is saved: SELECT SavedJobID WHERE JobSeekerID = ? AND JobID = ? |
| SavedJobs | D | DELETE FROM SavedJobs WHERE JobSeekerID = ? AND JobID = ? |
| SavedJobs | R | Count saved jobs: SELECT COUNT(\*) WHERE JobSeekerID = ? |

***SQL Commands:***

**1/ Get all saved jobs with details:**

SELECT sj.SavedJobID, sj.JobSeekerID, sj.JobID, sj.SavedDate,

j.JobTitle, r.CompanyName, l.LocationName, j.SalaryRange,

c.CategoryName as Industry, j.PostingDate

FROM SavedJobs sj

JOIN Jobs j ON sj.JobID = j.JobID

JOIN Recruiter r ON j.RecruiterID = r.RecruiterID

JOIN Locations l ON j.LocationID = l.LocationID

JOIN Categories c ON j.CategoryID = c.CategoryID

WHERE sj.JobSeekerID = ?

ORDER BY sj.SavedDate DESC

**2/ Save a job:**

-- First check if already saved

SELECT SavedJobID FROM SavedJobs

WHERE JobSeekerID = ? AND JobID = ?

-- If not exists, insert

INSERT INTO SavedJobs (JobSeekerID, JobID, SavedDate)

VALUES (?, ?, GETDATE())

**3/ Unsave a job:**

DELETE FROM SavedJobs

WHERE JobSeekerID = ? AND JobID = ?

**4/ Check if job is saved:**

SELECT SavedJobID FROM SavedJobs

WHERE JobSeekerID = ? AND JobID = ?

**5/ Get saved jobs count:**

SELECT COUNT(\*) as count

FROM SavedJobs

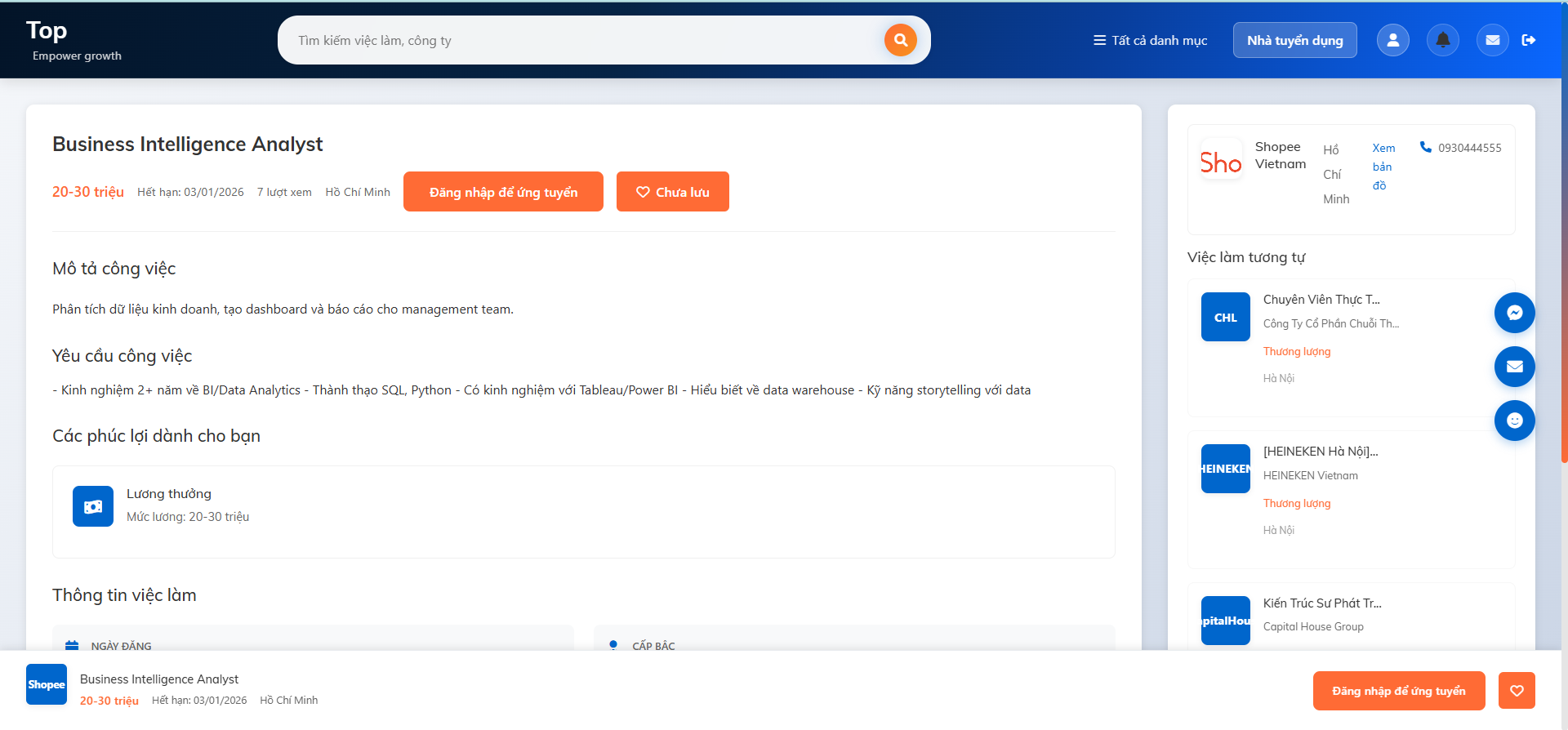
WHERE JobSeekerID = ?

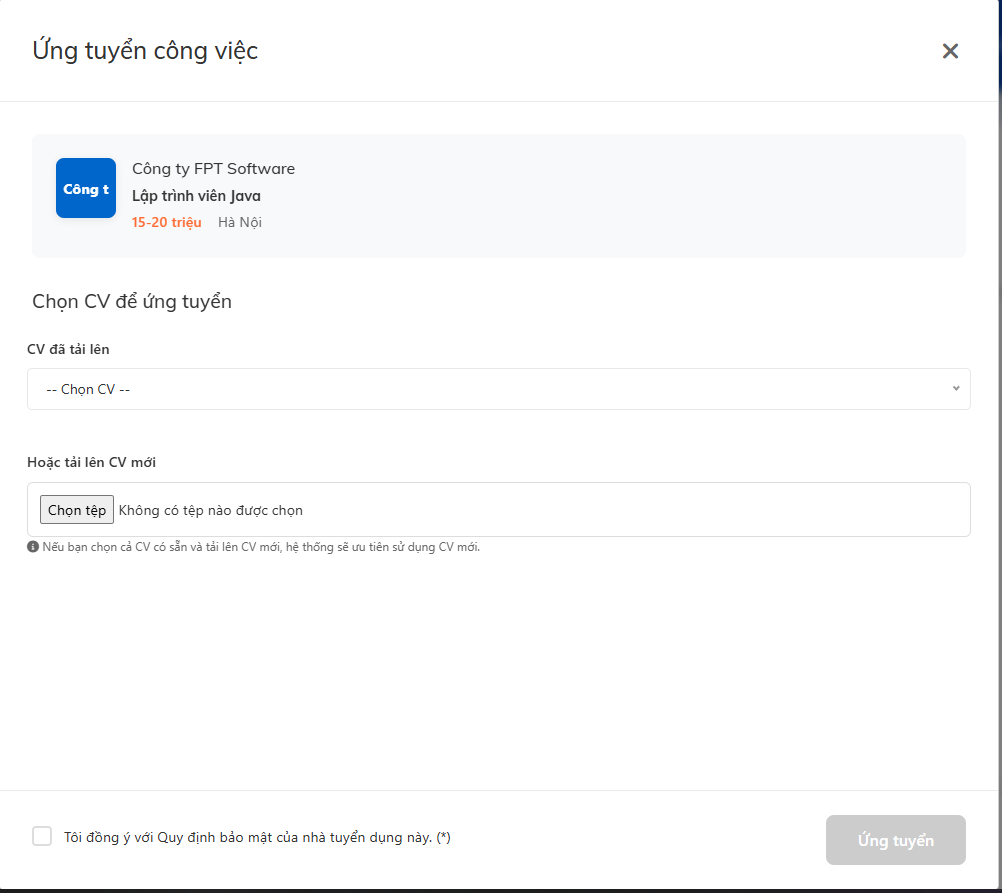
## 7. Job application

### 7.1 Apply for job

#### *a. View job detail, apply job*

##### **UI Design**





| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Job Title | Text Header | Job position title displayed prominently |
| Salary Range | Text Badge | Salary information (e.g., "20-30 triệu") |
| Posting Date | Date | Date when job was posted. Format: DD/MM/YYYY |
| Expiration Date | Date | Job application deadline. Format: "Hết hạn: DD/MM/YYYY" |
| View Count | Number | Number of times job has been viewed |
| Location | Text/Icon | Job location/city name |
| Company Name | Text/Logo | Recruiter's company name with logo |
| Company Logo | Image | Company logo image (CompanyLogoURL) |
| Company Address | Text | Company's physical address |
| Company Size | Text | Number of employees (e.g., "100-500 người") |
| Company Website | Link | Company website URL |
| Company Description | Rich Text | Detailed company information |
| Company Benefits | Rich Text | Employee benefits and perks |
| Job Description | Rich Text | Detailed job description and responsibilities |
| Job Requirements | Rich Text | Required skills, experience, and qualifications |
| Age Requirement | Number | Required age for applicants |
| Hiring Count | Number | Number of positions to fill |
| Job Level | Badge | Job level (Intern, Junior, Senior, Manager, etc.) |
| Job Type | Badge | Employment type (Full-time, Part-time, Contract) |
| Certificates | Badge | Required certifications |
| Category | Badge | Job category/industry |
| CV Selection | Dropdown/Radio | Select CV from existing CVs or upload new one |
| ***Job Table*** | | |
| JobID | Integer | Primary key |
| RecruiterID | Integer (FK) | Reference to Recruiter |
| JobTitle | VARCHAR(255) | Job position title |
| Description | TEXT | Detailed job description |
| Requirements | TEXT | Job requirements |
| SalaryRange | VARCHAR(100) | Salary range text |
| PostingDate | DATETIME | When job was posted |
| ExpirationDate | DATETIME | Application deadline |
| AgeRequirement | Integer | Required age |
| HiringCount | Integer | Number of positions |
| Status | VARCHAR(50) | Job status (Published, Closed, etc.) |
| ViewCount | Integer | Number of views |
| JobLevelID | Integer (FK) | Reference to Types (Level) |
| JobTypeID | Integer (FK) | Reference to Types (JobType) |
| CertificatesID | Integer (FK) | Reference to Types (Certificate) |
| LocationID | Integer (FK) | Reference to Locations |
| CategoryID | Integer (FK) | Reference to Categories |
| ***Application Table*** | | |
| ApplicationID | Integer | Primary key |
| JobID | Integer (FK) | Reference to Jobs |
| CVID | Integer (FK) | Reference to CVs |
| ApplicationDate | DATETIME | When application was submitted |
| Status | VARCHAR(50) | Application status (Pending, Accepted, Rejected) |
| ***Data Actions*** | | |
| Apply Job | "Đăng nhập để ứng tuyển" Button | Opens application modal. Requires login. Allows CV selection or upload. |
| Save Job | Heart Icon "Chưa lưu" | Saves job to saved jobs list (requires login) |
| Unsave Job | Filled Heart Icon | Removes job from saved jobs list |
| Select Existing CV | Radio Button | Choose from user's uploaded CVs |
| Upload New CV | File Upload | Upload new CV file (.pdf, .doc, .docx) for this application |
| View Company Profile | Company Name Link | Navigate to company profile/details page |
| Contact Company | Phone/Contact Info | Display company contact information |
| Increment View Count | Automatic | Increments ViewCount when page loads via stored procedure |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Jobs | R | Get complete job details with all related information |
| Recruiter | R | JOIN to get company information (name, logo, description, benefits, etc.) |
| Locations | R | JOIN to get location name |
| Categories | R | JOIN to get category name |
| Types | R | JOIN to get JobLevel, JobType, Certificate names |
| Jobs | U | Increment ViewCount via stored procedure |
| Applications | C | INSERT new application when user applies |
| Applications | R | Check if user has already applied: SELECT COUNT(\*) WHERE JobID = ? AND JobSeekerID = ? |
| CVs | R | Get user's CV list for selection |
| CVs | C | Insert new CV if user uploads during application |
| SavedJobs | C/D | Save or unsave job |

***SQL Commands:***

**1/ Get job detail with all related information:**

SELECT

j.JobID, j.JobTitle, j.Description, j.Requirements, j.SalaryRange,

j.PostingDate, j.ExpirationDate, j.AgeRequirement, j.HiringCount,

j.Status, j.ViewCount,

r.RecruiterID, r.CompanyName, r.CompanyDescription, r.CompanyLogoURL,

r.Website, r.CompanyAddress, r.CompanySize, r.CompanyBenefits,

l.LocationID, l.LocationName,

c.CategoryID, c.CategoryName,

jlevel.TypeID as JobLevelID, jlevel.TypeName as JobLevelName,

jtype.TypeID as JobTypeID, jtype.TypeName as JobTypeName,

cert.TypeID as CertificateID, cert.TypeName as CertificateName

FROM Jobs j

LEFT JOIN Recruiter r ON j.RecruiterID = r.RecruiterID

LEFT JOIN Locations l ON j.LocationID = l.LocationID

LEFT JOIN Categories c ON j.CategoryID = c.CategoryID

LEFT JOIN Types jlevel ON j.JobLevelID = jlevel.TypeID

LEFT JOIN Types jtype ON j.JobTypeID = jtype.TypeID

LEFT JOIN Types cert ON j.CertificatesID = cert.TypeID

WHERE j.JobID = ?

**2/ Increment job view count (Stored Procedure):**

EXEC sp\_IncrementJobView @JobID = ?

**3/ Check if user already applied:**

SELECT COUNT(\*) FROM Applications

WHERE JobID = ? AND CVID IN (SELECT CVID FROM CVs WHERE JobSeekerID = ?)

**4/ Insert new application:**

INSERT INTO Applications (JobID, CVID, ApplicationDate, Status)

VALUES (?, ?, GETDATE(), 'Pending')

**5/ Get user's CV list:**

SELECT CVID, CVTitle, CVURL, IsActive, CreationDate

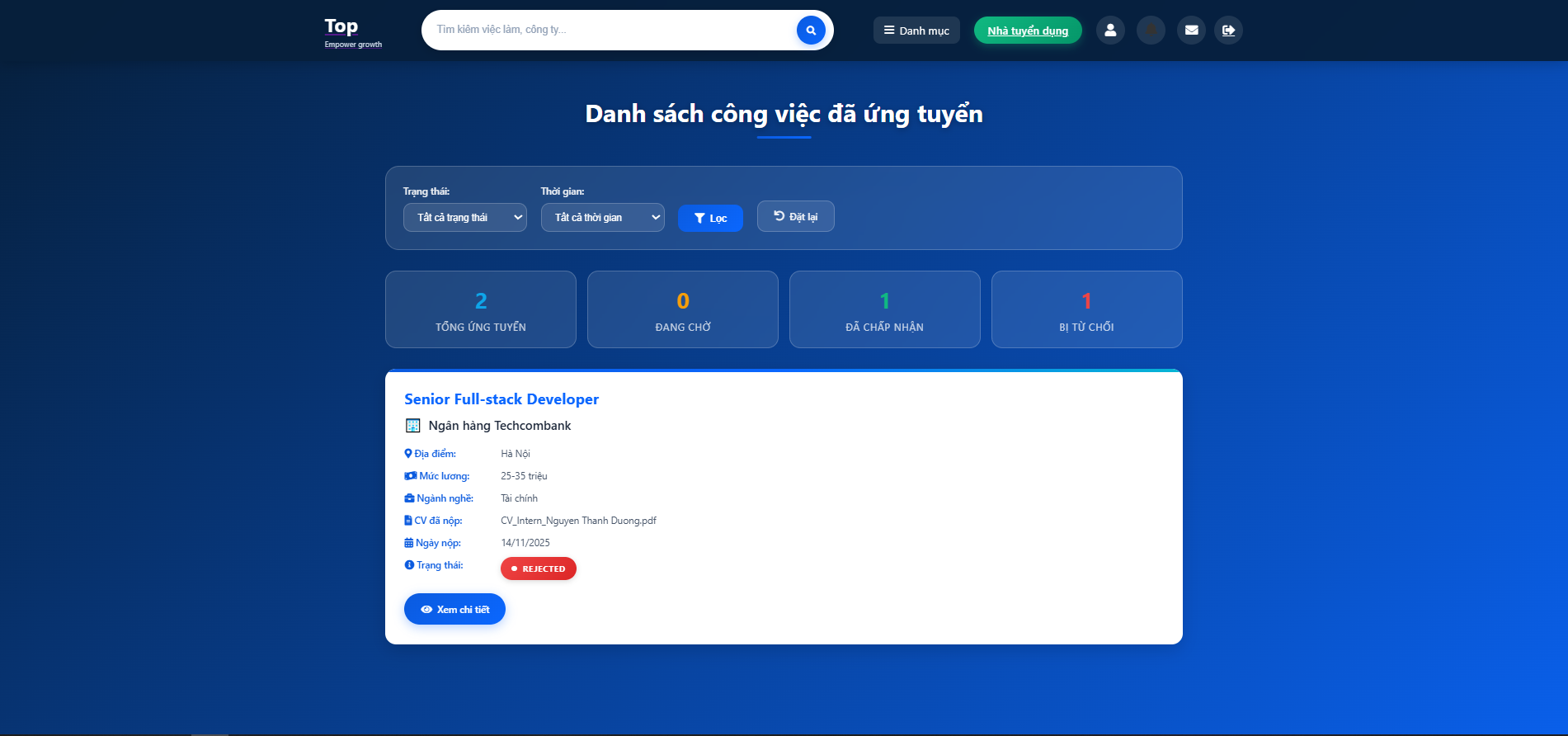
FROM CVs

WHERE JobSeekerID = ?

#### 

#### *b. View applied job*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Job Title | Text Link | Job position title, clickable to view job detail |
| Company Name | Text | Recruiter's company name |
| Location | Text/Icon | Job location from Locations table |
| Industry/Category | Badge | Company industry/category |
| Salary Range | Text | Salary information (e.g., "15-20 triệu") |
| Job Type | Badge | Employment type (Full-time, Part-time, Contract) from Types table |
| Application Date | DateTime | Date and time when user applied. Format: DD/MM/YYYY |
| Status | Badge/Status | Application status: Pending, Accepted, Rejected, Interviewed |
| CV Used | Text | Title of CV used for this application |
| Total Count | Number | Total number of applications |
| Statistics | Number Cards | Breakdown by status (Pending, Accepted, Rejected, Interviewed) |
| ***Data Table*** | | |
| ApplicationID | Integer | Primary key, auto-increment identifier |
| JobID | Integer (FK) | Foreign key reference to Jobs table |
| CVID | Integer (FK) | Foreign key reference to CVs table |
| ApplicationDate | DATETIME | Timestamp when application was submitted. Default: GETDATE() |
| Status | VARCHAR(50) | Application status: Pending, Accepted, Rejected, Interviewed |
| ***Data Actions*** | | |
| View Job Detail | Job Card Click | Navigate to job detail page |
| View Company Profile | Company Name Click | Navigate to company profile page |
| Cancel Application | Delete/Cancel Button | Delete application (only for Pending status). Sends notification. |
| Filter by Status | Dropdown Selection | Filter applications by status |
| Filter by Date Range | Dropdown Selection | Filter applications by application date range |
| View CV | CV Link | View or download the CV used for application |
| Apply to Job | Button (if not applied) | Navigate to job detail page to apply |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Applications | R | Get all applications for user with full details via JOIN |
| Applications | R | Get applications with filters (status, date range) |
| Applications | R | Get application statistics by status |
| Applications | D | Delete application (only Pending status, with JobSeeker verification) |
| Applications | U | Update application status (by recruiter, triggers notification) |
| CVs | R | JOIN to verify JobSeekerID and get CV details |
| Jobs | R | JOIN to get job details |
| Recruiter | R | JOIN to get company information |

***SQL Commands:***

**1/ Get all applied jobs with full details:**

SELECT

a.ApplicationID, a.JobID, a.CVID, a.ApplicationDate, a.Status,

j.JobTitle, t.TypeName as JobType, j.SalaryRange, j.RecruiterID,

r.CompanyName, COALESCE(c.CategoryName, 'Công nghệ') as Industry,

l.LocationName, cv.CVTitle

FROM Applications a

JOIN CVs cv ON a.CVID = cv.CVID

JOIN Jobs j ON a.JobID = j.JobID

JOIN Recruiter r ON j.RecruiterID = r.RecruiterID

JOIN Locations l ON j.LocationID = l.LocationID

LEFT JOIN Types t ON j.JobTypeID = t.TypeID

LEFT JOIN Categories c ON r.CategoryID = c.CategoryID

WHERE cv.JobSeekerID = ?

ORDER BY a.ApplicationDate DESC

**2/ Get applications with filters:**

SELECT ... (same as above)

WHERE cv.JobSeekerID = ?

AND a.Status = ? -- Optional status filter

AND a.ApplicationDate >= DATEADD(day, ?, GETDATE()) -- Optional date range

ORDER BY a.ApplicationDate DESC

**3/ Get application statistics:**

SELECT

COUNT(\*) as total,

SUM(CASE WHEN a.Status = 'Pending' THEN 1 ELSE 0 END) as pending,

SUM(CASE WHEN a.Status = 'Accepted' THEN 1 ELSE 0 END) as accepted,

SUM(CASE WHEN a.Status = 'Rejected' THEN 1 ELSE 0 END) as rejected,

SUM(CASE WHEN a.Status = 'Interviewed' THEN 1 ELSE 0 END) as interviewed

FROM Applications a

JOIN CVs cv ON a.CVID = cv.CVID

WHERE cv.JobSeekerID = ?

**4/ Delete/Cancel application (Pending only):**

-- First verify status and ownership

SELECT a.Status FROM Applications a

JOIN CVs cv ON a.CVID = cv.CVID

WHERE a.ApplicationID = ? AND cv.JobSeekerID = ?

-- If status is Pending, delete

DELETE FROM Applications

WHERE ApplicationID = ?

AND Status = 'Pending'

AND CVID IN (SELECT CVID FROM CVs WHERE JobSeekerID = ?)

**5/ Check if already applied:**

SELECT COUNT(\*) FROM Applications a

JOIN CVs c ON a.CVID = c.CVID

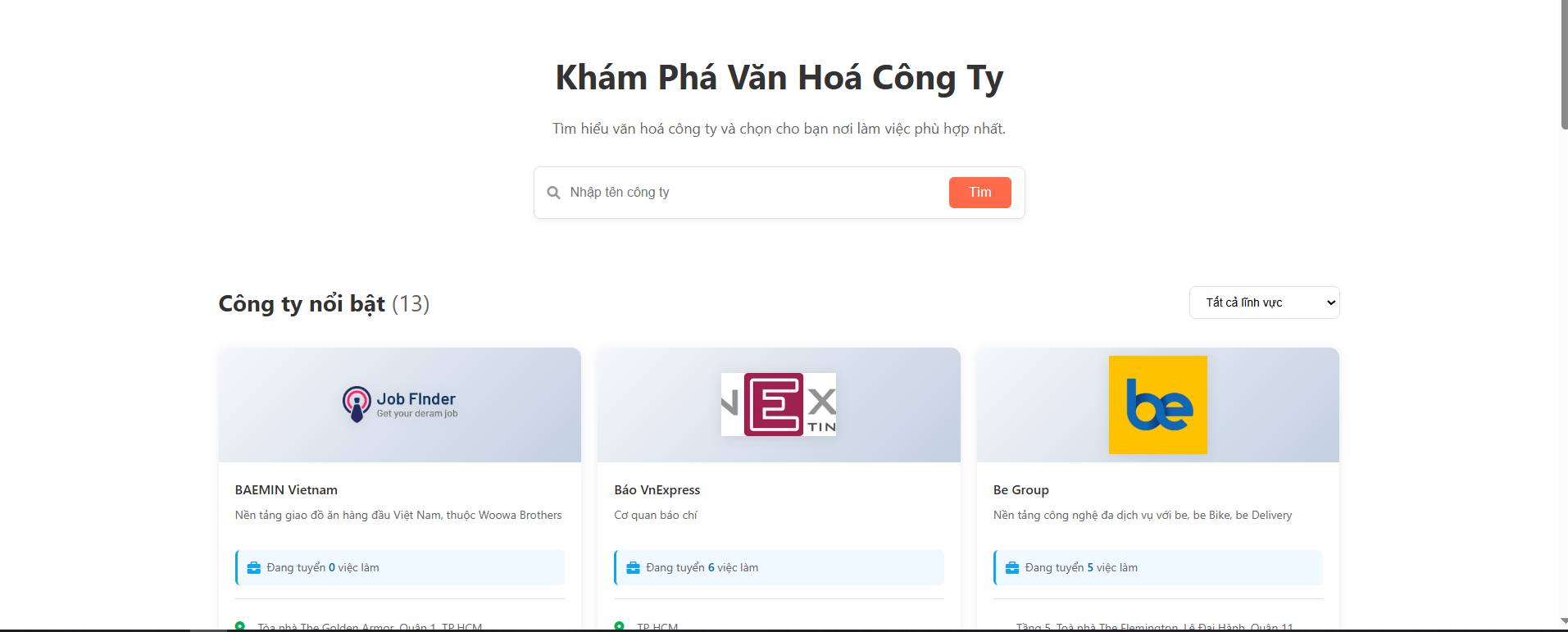
WHERE a.JobID = ? AND c.JobSeekerID = ?

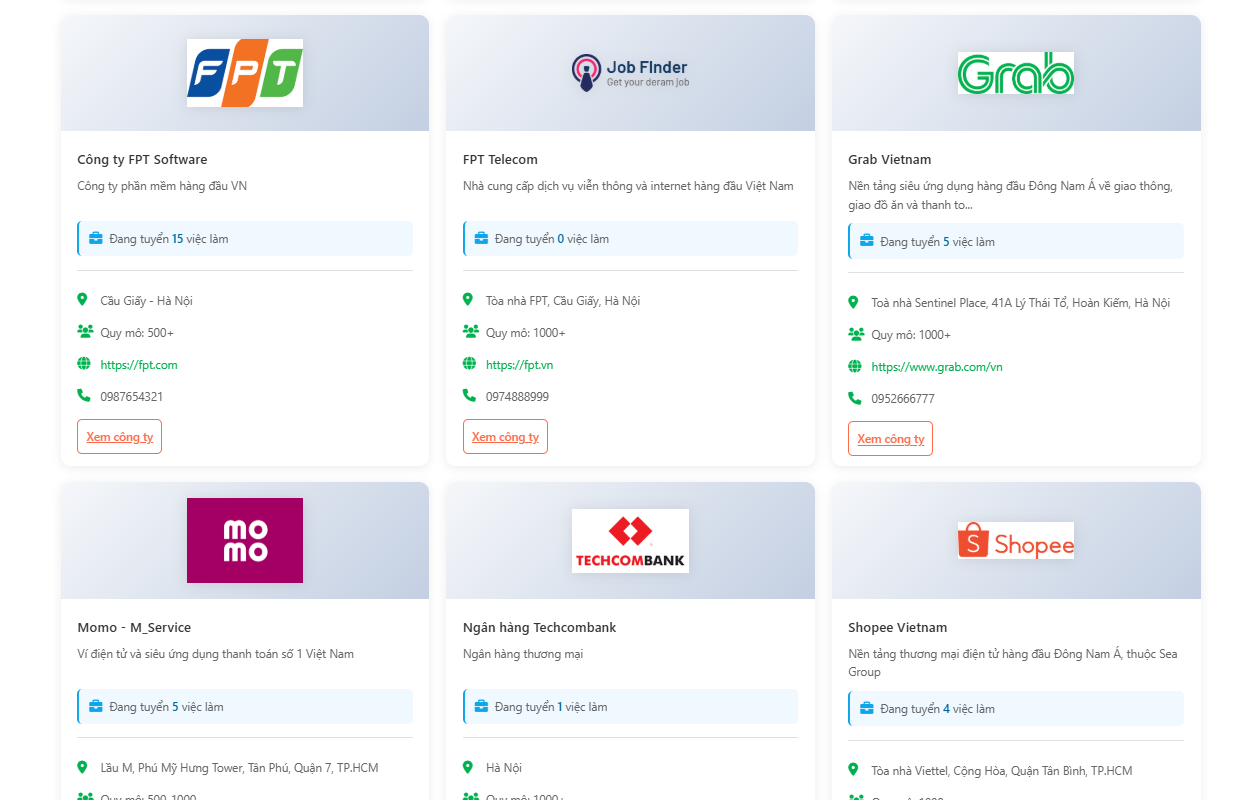
## 8. Company information

### 8.1 View all company

#### *a. View all company*

##### **UI Design**





| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Search Box | Text Input | Search companies by company name. Trims whitespace and handles empty values. |
| Category Filter | Dropdown | Filter companies by industry category. Options from Categories table. |
| Company Name | Text/Link | Company name, clickable to view company detail page |
| Company Logo | Image | Company logo image (CompanyLogoURL) |
| Company Description | Text (Truncated) | Brief company description with "Read more" option |
| Company Address | Text | Company's physical address location |
| Company Size | Text | Number of employees (e.g., "100-500", "500-1000", "1000+") |
| Job Count | Number Badge | Number of currently published jobs from this company |
| Company Benefits | Text List | Employee benefits and perks offered |
| Website | Link | Company website URL |
| ***Recruiter Table*** | | |
| RecruiterID | Integer | Primary key, auto-increment recruiter identifier |
| Email | VARCHAR(255) | Company/recruiter email address |
| Password | VARCHAR(255) | Hashed password (MD5) |
| Phone | VARCHAR(20) | Company contact phone number |
| CompanyName | VARCHAR(255) | Official company name |
| CompanyDescription | TEXT | Detailed company description and culture |
| CompanyLogoURL | VARCHAR(500) | Path to company logo image |
| Website | VARCHAR(500) | Company website URL |
| Img | VARCHAR(500) | Additional company image |
| CategoryID | Integer (FK) | Foreign key reference to Categories table |
| Status | VARCHAR(50) | Account status (Active, Inactive, Pending) |
| CompanyAddress | VARCHAR(500) | Company physical address |
| CompanySize | VARCHAR(100) | Employee count range |
| ContactPerson | VARCHAR(255) | Contact person name |
| CompanyBenefits | TEXT | Employee benefits and perks |
| CompanyVideoURL | VARCHAR(500) | Company introduction video URL |
| Taxcode | VARCHAR(50) | Company tax code |
| RegistrationCert | VARCHAR(500) | Business registration certificate |
| ***Jobs Table ( for counting )*** | | |
| JobID | Integer | Primary key |
| RecruiterID | Integer (FK) | Reference to Recruiter |
| Status | VARCHAR(50) | Job status (Published, Closed, Draft) |
| ***Data Actions*** | | |
| Search Companies | Search Button/Icon | Submit search form with company name keyword |
| Filter by Category | Dropdown Selection | Filter companies by selected industry category |
| View Company Detail | Company Card Click | Navigate to detailed company profile page |
| View Company Jobs | "X việc làm" Link | Navigate to job list filtered by this company |
| Clear Filters | Clear/Reset Button | Remove all search and filter criteria |
| Visit Website | Website Link | Open company website in new tab |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Recruiter | R | Get all active companies with job count |
| Recruiter | R | Search companies by name with job count |
| Recruiter | R | Filter companies by category with job count |
| Recruiter | R | Combined search by name AND category with job count |
| Jobs | R | LEFT JOIN and COUNT to get number of published jobs per company |
| Categories | R | Get category list for filter dropdown |

***SQL Commands:***

**1/ Get all active companies with job count:**

SELECT R.\*, COUNT(J.JobID) AS JobCount

FROM Recruiter R

LEFT JOIN Jobs J ON R.RecruiterID = J.RecruiterID AND J.Status = 'Published'

WHERE R.Status = 'Active'

GROUP BY R.RecruiterID, R.Email, R.Password, R.Phone, R.CompanyName,

R.CompanyDescription, R.CompanyLogoURL, R.Website, R.Img,

R.CategoryID, R.Status, R.CompanyAddress, R.CompanySize,

R.ContactPerson, R.CompanyBenefits, R.CompanyVideoURL,

R.Taxcode, R.RegistrationCert

ORDER BY R.CompanyName

**2/ Search companies by name with job count:**

SELECT R.\*, COUNT(J.JobID) AS JobCount

FROM Recruiter R

LEFT JOIN Jobs J ON R.RecruiterID = J.RecruiterID AND J.Status = 'Published'

WHERE R.Status = 'Active' AND R.CompanyName LIKE ?

GROUP BY R.RecruiterID, ... (all columns)

ORDER BY R.CompanyName

**3/ Filter companies by category with job count:**

SELECT R.\*, COUNT(J.JobID) AS JobCount

FROM Recruiter R

LEFT JOIN Jobs J ON R.RecruiterID = J.RecruiterID AND J.Status = 'Published'

WHERE R.Status = 'Active' AND R.CategoryID = ?

GROUP BY R.RecruiterID, ... (all columns)

ORDER BY R.CompanyName

**4/ Search by name AND category with job count:**

SELECT R.\*, COUNT(J.JobID) AS JobCount

FROM Recruiter R

LEFT JOIN Jobs J ON R.RecruiterID = J.RecruiterID AND J.Status = 'Published'

WHERE R.Status = 'Active'

AND R.CompanyName LIKE ?

AND R.CategoryID = ?

GROUP BY R.RecruiterID, ... (all columns)

ORDER BY R.CompanyName

**5/ Get categories for filter dropdown:**

SELECT CategoryID, CategoryName

FROM Categories

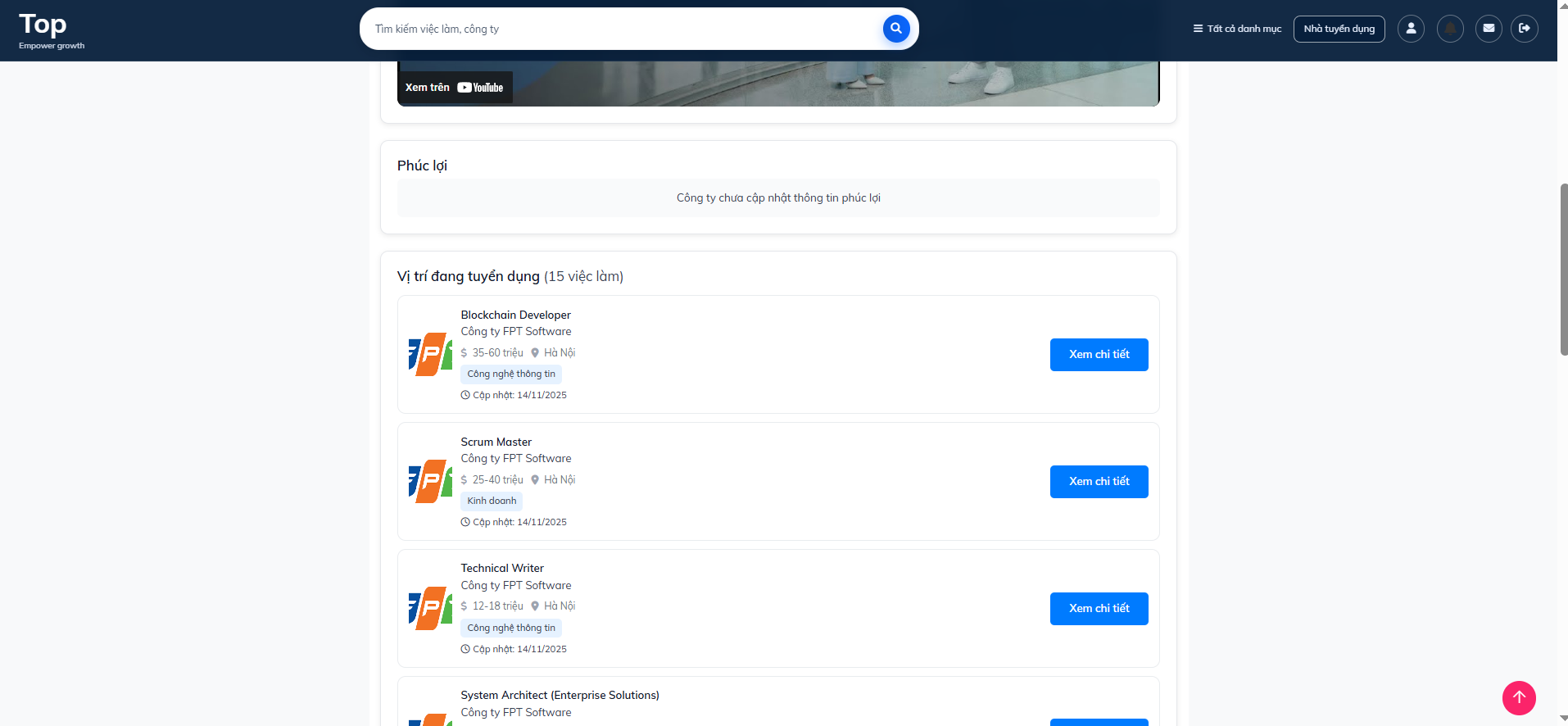
ORDER BY CategoryName

### 8.2 View company detail

#### *a. View company detail*

##### **UI Design**





| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Company Name | Text Header | Company's official name displayed prominently |
| Company Logo | Image | Company logo (CompanyLogoURL). Falls back to 3-letter abbreviation if no image |
| Company Description | Rich Text | Detailed company information, culture, and overview |
| Company Address | Text | Company's physical address location |
| Company Size | Text | Number of employees (e.g., "100-500", "500-1000") |
| Website | Link | Company website URL, clickable to open in new tab |
| Contact Person | Text | Company contact person name |
| Phone | Text | Company contact phone number |
| Email | Text | Company/recruiter email address |
| Tax Code | Text | Company tax registration number |
| Company Benefits | List | Employee benefits separated by pipe ( |
| Company Video | Video Embed | Company introduction/recruitment video (CompanyVideoURL) |
| Category | Badge | Company industry/category from Categories table |
| Job Listings | Card List | List of published jobs from this company |
| Job Count | Number | Total number of published jobs |
| ***Job listing Fields*** | | |
| **Field Name** | **Field Type** | **Description** |
| Job Title | Text Link | Job position title, clickable to job detail |
| Salary Range | Text | Salary range (e.g., "15-20 triệu") |
| Location | Text | Job location from Locations table |
| Category | Text | Job category from Categories table |
| Posting Date | Date | Date when job was posted |
| Expiration Date | Date | Application deadline |
| ***Recruiter Table*** | | |
| RecruiterID | Integer | Primary key |
| Email | VARCHAR(255) | Company email |
| Phone | VARCHAR(20) | Contact phone |
| CompanyName | VARCHAR(255) | Official company name |
| CompanyDescription | TEXT | Detailed company description |
| CompanyLogoURL | VARCHAR(500) | Path to company logo |
| Website | VARCHAR(500) | Company website URL |
| CategoryID | Integer (FK) | Reference to Categories |
| Status | VARCHAR(50) | Account status (Active, Inactive) |
| CompanyAddress | VARCHAR(500) | Physical address |
| CompanySize | VARCHAR(100) | Employee count range |
| ContactPerson | VARCHAR(255) | Contact person name |
| CompanyBenefits | TEXT | Benefits list (pipe-separated) |
| CompanyVideoURL | VARCHAR(500) | Company video URL |
| Taxcode | VARCHAR(50) | Tax registration code |
| ***Jobs Table*** | | |
| JobID | Integer | Primary key |
| RecruiterID | Integer (FK) | Reference to Recruiter |
| JobTitle | VARCHAR(255) | Job position title |
| Description | TEXT | Job description |
| Requirements | TEXT | Job requirements |
| SalaryRange | VARCHAR(100) | Salary range |
| PostingDate | DATETIME | When job was posted |
| ExpirationDate | DATETIME | Application deadline |
| LocationID | Integer (FK) | Reference to Locations |
| CategoryID | Integer (FK) | Reference to Categories |
| Status | VARCHAR(50) | Job status (Published, Closed, Draft) |
| ***Data Actions*** | | |
| View Job Detail | Job Card Click/"View" Button | Navigate to job detail page for specific job |
| Visit Company Website | Website Link | Open company website in new tab/window |
| Watch Company Video | Video Player | Play company introduction/recruitment video |
| Contact Company | Email/Phone Link | Initiate contact via email or phone |
| Filter Jobs by Category | Filter Option | Filter displayed jobs by category (if implemented) |
| Filter Jobs by Location | Filter Option | Filter displayed jobs by location (if implemented) |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Recruiter | R | SELECT \* FROM Recruiter WHERE RecruiterID = ? - Get company information |
| Jobs | R | SELECT \* FROM Jobs WHERE RecruiterID = ? ORDER BY PostingDate DESC - Get all jobs |
| Jobs | R | Filter to only Published status jobs for display |
| Locations | R | SELECT \* FROM Locations WHERE LocationID = ? - Get location names for jobs |
| Categories | R | SELECT \* FROM Categories WHERE CategoryID = ? - Get category names |

***SQL Commands:***

**1/ Get company information:**

SELECT \* FROM Recruiter WHERE RecruiterID = ?

**2/ Get all jobs by recruiter:**

SELECT \* FROM Jobs

WHERE RecruiterID = ?

ORDER BY PostingDate DESC

**3/ Get location name:**

SELECT LocationID, LocationName

FROM Locations

WHERE LocationID = ?

**4/ Get category name:**

SELECT CategoryID, CategoryName

FROM Categories

WHERE CategoryID = ?

**5/ Get company category info:**

SELECT c.CategoryID, c.CategoryName

FROM Recruiter r

JOIN Categories c ON r.CategoryID = c.CategoryID

WHERE r.RecruiterID = ?

## 9. Authentication

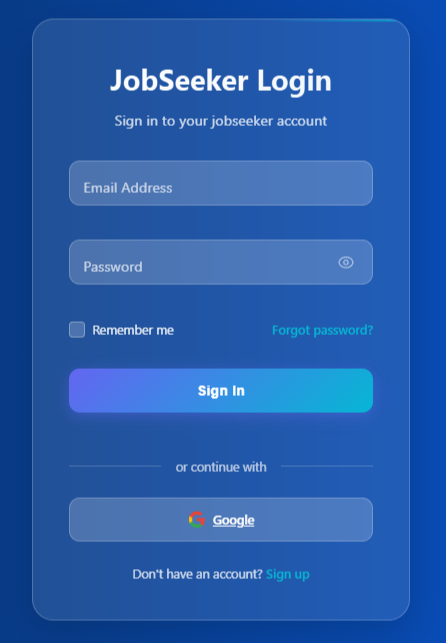
### 9.1 Login

#### *a. JobSeeker Login*

This screen allows user to be authenticated to the system screens/functionalities.

Related use cases:

* [UC02\_Login System](#_h2xksjvozg4w)



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Email\* | Text Box | This is for user to input valid email address for logging in |
| Password\* | Password Box | This is for user to input password for logging in |
| Sign In | Button | User clicks to authenticate him/herself into the system with provided email & password |
| Sign up | Button | User clicks to redirect to the User Register page for registering new user account to access the system |
| Forgot Password? | Hyperlink | User clicks to redirect to the Password Reset page for resetting his/her forgot password |
| Login with Google | Hyperlink | Allow user to login with his/her Google account |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| User | R | Verify UserName & Password information |

***SQL Commands:***

1/ Verify UserName & Password information

SELECT \* FROM JobSeeker WHERE Email = ? AND Password = ? AND Status = 'Active'

#### *b. Recruiter Login*

This screen allows user to be authenticated to the system screens/functionalities.

Related use cases:

* [UC02\_Login System](#_h2xksjvozg4w)



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Email\* | Text Box | This is for user to input valid email address for logging in |
| Password\* | Password Box | This is for user to input password for logging in |
| Sign In | Button | User clicks to authenticate him/herself into the system with provided email & password |
| Sign up | Button | User clicks to redirect to the User Register page for registering new user account to access the system |
| Forgot Password? | Hyperlink | User clicks to redirect to the Password Reset page for resetting his/her forgot password |
| Back to home page | Hyperlink | User clicks to redirect to the Home page. |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| User | R | Verify UserName & Password information |

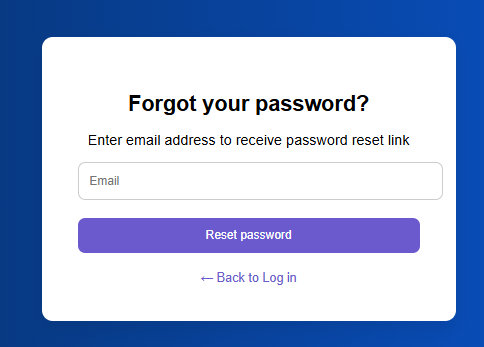
***SQL Commands:***

1/ Verify UserName & Password information

SELECT \* FROM Recruiter WHERE Email = ? AND Password = ? AND Status = 'Active'

### 9.2 Reset password

#### *a. Request forget Password Screen*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Email\* | Text Box | This is for user to input valid email address for logging in |
| Reset Password | Button | The user clicks on the system to confirm the email and send a password reset email to the entered email. |
| Back to Log in | Hyperlink | Users click on the system will be directly transferred to the login page. |

##### **Database Access**

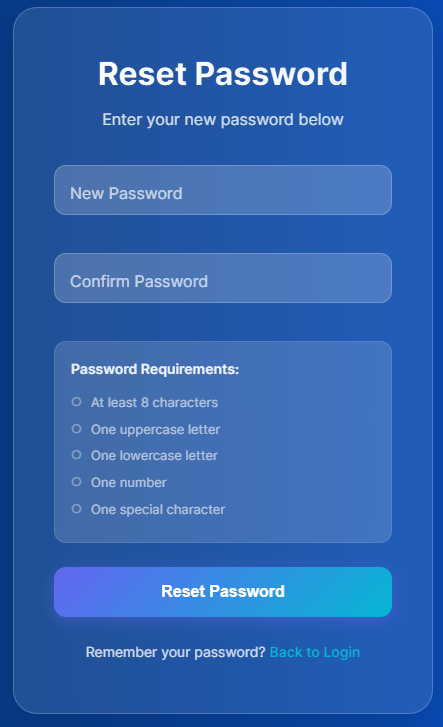
| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Reset Password | R | Query the current list of emails from the database. |

***SQL Commands:***

1/ Query the current list of emails from the database.

SELECT \* FROM ? WHERE Email = ?

#### *b. Reset Password Screen*



##### **UI Design**

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| New Password | Text Box | This is where the user enters the new password. |
| Confirm Password | Text Box | This is where the user confirms the new password. |
| Reset Password | Button | User clicks on the system to reset password |
| Back to Log in | Hyperlink | Users click on the system will be directly transferred to the login page. |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Reset Password | U | The system updates the password according to the new password. |

***SQL Commands:***

1/ The system updates the password according to the new password.

UPDATE ?

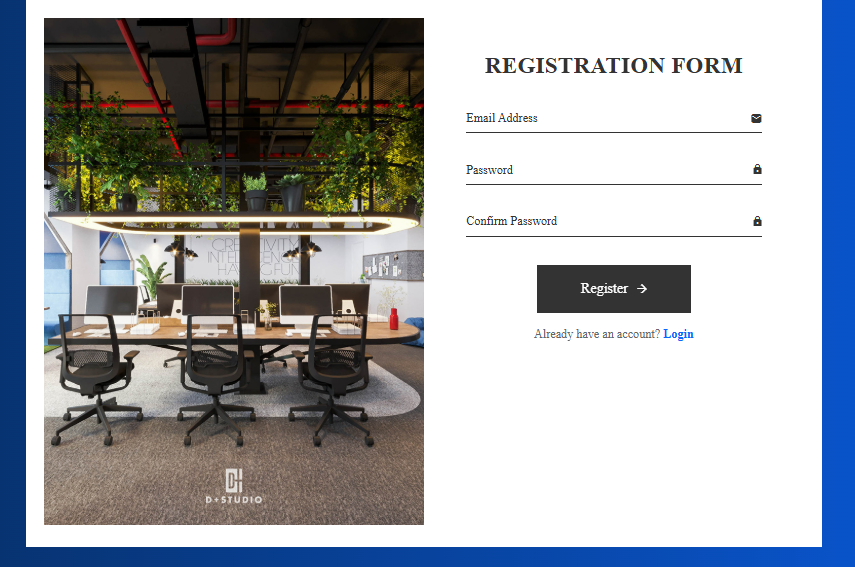
SET Password = ?

WHERE ID = ?

### 9.3 Register account

#### *a. JobSeeker Register*

##### **UI Design**



| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Email | Text Box | This is where you enter the email address of the candidate you want to register |
| Password | Text Box | This is where you enter the password |
| Confirm Password | Text Box | This is where you confirm the password again |
| Register | Button | The system proceeds to add account data to the system. |
| Login | Hyperlink | Users click on the system will be directly transferred to the login page. |

##### **Database Access**

| **Table** | **CRUD** | **Description** |
| --- | --- | --- |
| Register | RC | The system proceeds to add account data to the system. |

***SQL Commands:***

1.The system proceeds to add account data to the system.

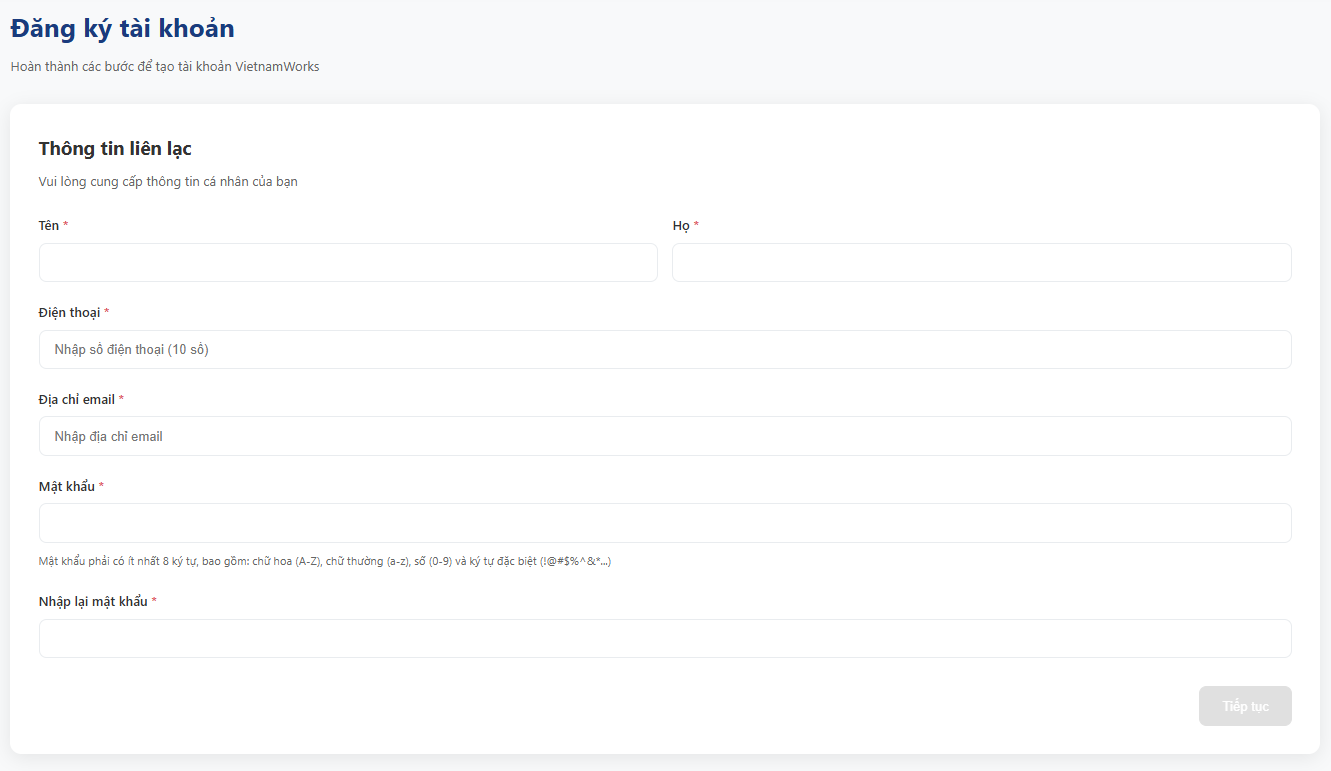
SELECT COUNT(\*) FROM JobSeeker WHERE Email = ?

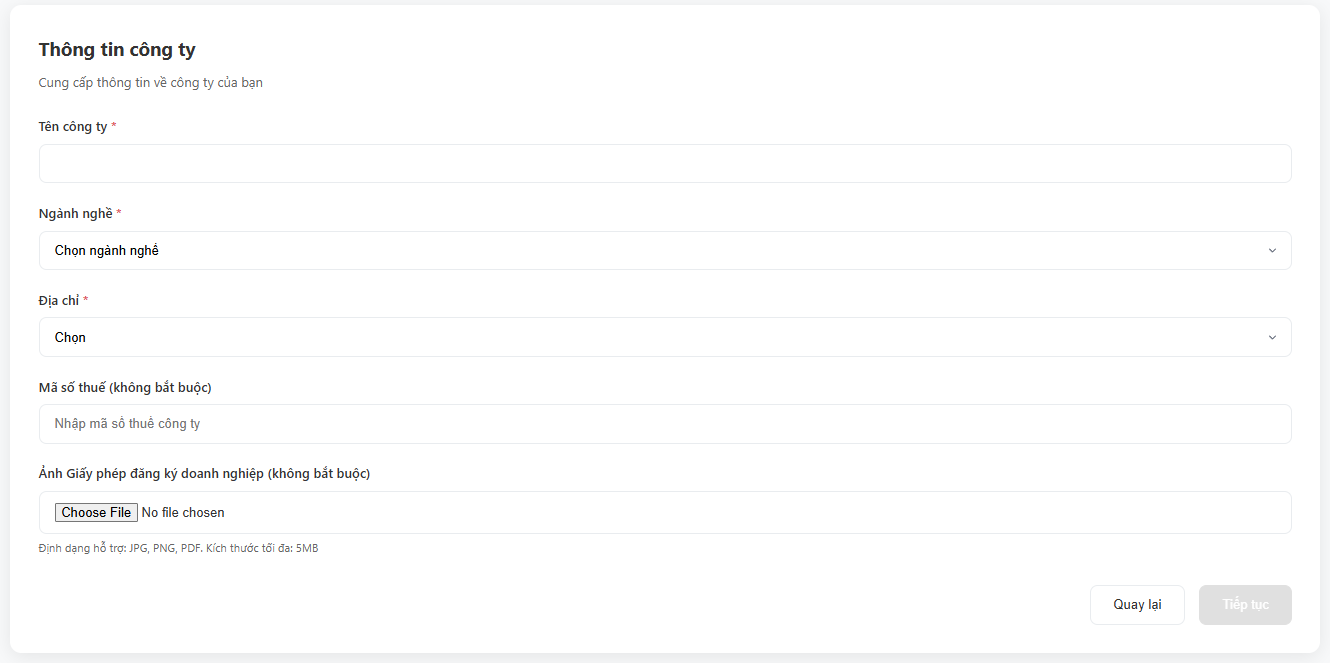
INSERT INTO JobSeeker (Email, Password, Status)

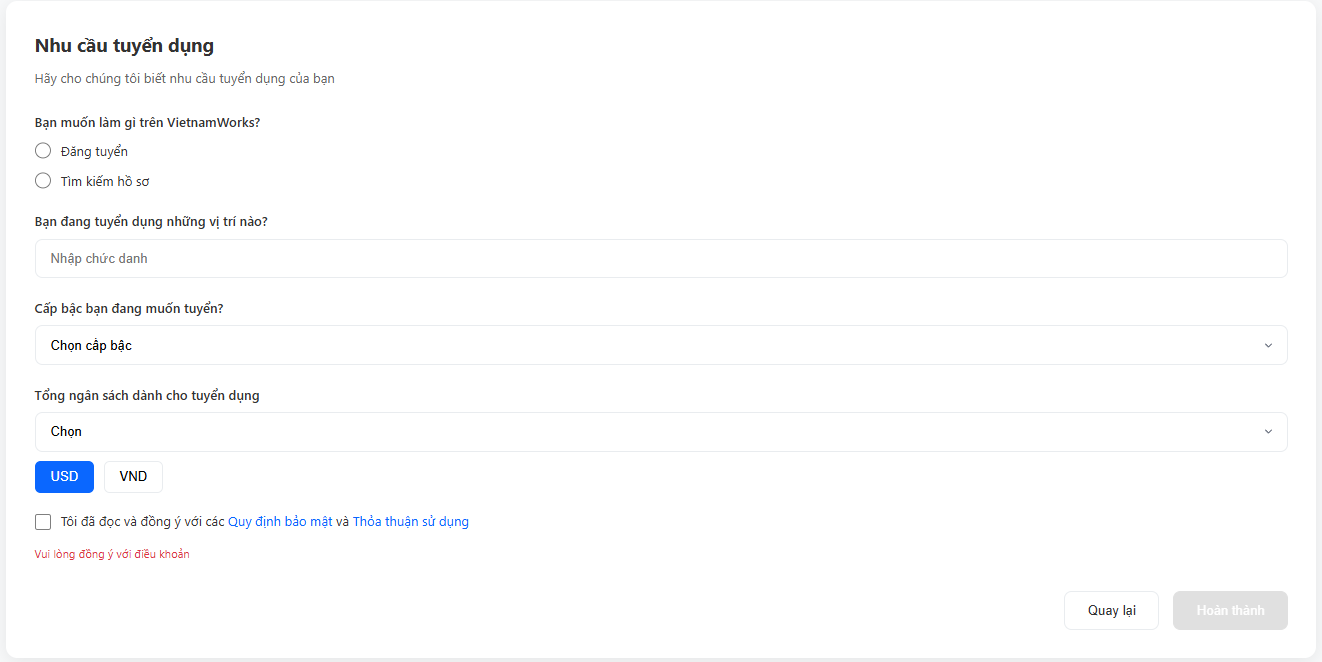
VALUES (?, ?, ?)

#### *b. Recruiter Register*

##### **UI Design**







| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| First Name \* | Text Box | User’s first name. Letters and spaces only. Max 50 characters. Regex: ^[A-Za-zÀ-ỹ\s]{1,50}$ |
| Last Name \* | Text Box | User’s last name. Letters and spaces only. Max 50 characters. |
| Phone \* | Text Box | User's phone number. Must be 10-11 digits. Validates using regex: ^\d{10,11}$ |
| Email \* | Text Box (Read-only) | User’s registered email address. This field cannot be edited. |
| Password \* | Password Box | Minimum 8 characters: uppercase, lowercase, number, special character. |
| Confirm Password \* | Password Box | Must match the Password field. |
| Company Name \* | Text Box | Name of the company. Max 255 characters. |
| Industry \* | Dropdown | Select business industry. References Industry table. |
| Address \* | Dropdown | Select location/city. References Location table. |
| Tax Code | Text Box | Company tax code (10–13 digits). Regex: ^\d{10,13}$ |
| Business License Image | File Upload | Upload Business Registration Certificate. Supports JPG/PNG/PDF, maximum size 5MB. |
| Action Type \* | Radio Button | Options: “Post Job” or “Search Resumes”. |
| Job Title | Text Box | Job title the employer wants to recruit. Max 100 characters. |
| Level | Dropdown | Desired job level. References JobLevel table. |
| Budget | Dropdown | Select total recruitment budget. References BudgetRange table |
| Currency | Toggle Button | Options: USD or VND. |
| Agree to Terms \* | Checkbox | User must check to proceed. |

##### **Database Access**

| Table | CRUD | Description |
| --- | --- | --- |
| User | CR | Create new user account & validate email uniqueness |
| Company | CU | Save and update company information |
| Recruitment | C | Store recruitment requirement data |
| Industry | R | Read list of industries for dropdown |
| Location | R | Read list of locations/cities for dropdown |
| JobLevel | R | Read list of job levels for dropdown |

***SQL Commands:***

1.Create new user account & validate email uniqueness

SELECT COUNT(\*) FROM Recruiter WHERE Email = ?

SELECT COUNT(\*) FROM Recruiter WHERE Phone = ?

INSERT INTO Recruiter (Email, Password, Phone, CompanyName, CompanyAddress, Status, ContactPerson, CategoryID, Taxcode, RegistrationCert)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

2.Read list of locations/cities for dropdown

SELECT \* FROM Locations

ORDER BY LocationName

3.Read list of industries for dropdown

SELECT \* FROM Categories

WHERE ParentCategoryID IS NULL

ORDER BY CategoryName

4.Read list of job levels for dropdown

SELECT \* FROM Types

WHERE TypeCategory = 'Level'

ORDER BY TypeName

# IV. Appendix

## 1. Assumptions & Dependencies

AS-1: Database Availability and Performance

- The MySQL/SQL Server database will maintain 99.5% uptime during business hours (7 AM - 11 PM).

- The database can handle concurrent transactions from at least 500 active users without significant performance degradation.

- Database backup and recovery procedures are in place and tested regularly.

AS-2: User Internet Connectivity

- Job seekers and recruiters have access to stable internet connections with minimum speeds of 2 Mbps for basic functionality.

- Mobile users have access to 3G/4G or WiFi connectivity for accessing the platform.

AS-3: User Device Compatibility

- Users have access to modern web browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+) released within the last 2 years.

- Mobile devices run Android 8.0+ or iOS 12+ for optimal mobile experience.

AS-4: Email Service Availability

- Email service providers (Gmail, Yahoo, Outlook, etc.) will accept and deliver system-generated emails.

- Users provide valid and active email addresses during registration.

- Email delivery occurs within 5 minutes under normal circumstances.

AS-5: Payment Gateway Integration

- Third-party payment gateways (VNPay, MoMo, ZaloPay, or similar) maintain their service availability and API compatibility.

- Payment processing occurs in real-time with immediate confirmation.

AS-6: Content Moderation Capacity

- Staff members are available during business hours to review and approve job postings within 24 hours.

- Automated content filtering tools can identify and flag inappropriate content with 80% accuracy.

AS-7: File Storage and CV Processing

- Sufficient server storage space is available for storing CVs and company documents (estimated 10GB initial, scaling as needed).

- PDF and common document formats (DOC, DOCX) can be processed and displayed reliably.

AS-8: User Competency

- Job seekers have basic computer literacy to create profiles, upload CVs, and apply for jobs.

- Recruiters have sufficient technical knowledge to post jobs, manage applications, and use company profile features.

- Admin staff can perform basic content moderation and system management tasks.

AS-9: Company Information Accuracy

- Recruiters provide legitimate and accurate company information during registration.

- Companies have the legal right to recruit for positions they post.

AS-10: Scalability Requirements

- The system architecture supports horizontal scaling to accommodate 10x growth in user base.

- Cloud infrastructure or hosting provider can allocate additional resources on demand.

DE-1: Java Development Environment

- The system depends on Java SE 8+ runtime environment for backend operations.

- Apache Tomcat 9.0+ or compatible servlet container for deployment.

- JSP/Servlet technology stack remains supported and maintained.

DE-2: Database Management System

- Dependency on MySQL 8.0+ or SQL Server 2016+ for data persistence.

- JDBC drivers maintain compatibility with chosen database system.

- Database connection pooling libraries (e.g., HikariCP, Apache DBCP) function reliably.

DE-3: External Payment Integration

- Integration with Vietnamese payment gateways (VNPay, MoMo, ZaloPay, or international alternatives).

- Payment gateway APIs remain stable and backward-compatible.

- Merchant accounts are approved and maintained in good standing.

DE-4: Email Service Provider

- SMTP server or email service API (SendGrid, AWS SES, or similar) for sending notifications.

- Email service maintains deliverability rates above 95%.

DE-5: Web Hosting Infrastructure

- Reliable hosting provider or cloud service (AWS, Azure, Google Cloud, or local Vietnam providers).

- SSL/TLS certificates for secure HTTPS connections.

- Domain name registration and DNS management.

DE-6: Frontend Libraries and Frameworks

- Bootstrap 4.x/5.x for responsive UI components.

- jQuery for DOM manipulation and AJAX calls.

- Chart.js or similar libraries for dashboard visualizations.

DE-7: File Upload and Storage

- File upload libraries compatible with Java servlet specifications.

- Storage solution (local filesystem, Amazon S3, or similar) for document persistence.

DE-8: Security Components

- Password hashing libraries (BCrypt, PBKDF2) for secure credential storage.

- Session management frameworks for maintaining user authentication state.

- CAPTCHA service (reCAPTCHA v2/v3) to prevent automated abuse.

DE-9: Development Tools

- NetBeans IDE or compatible Java IDE for development and maintenance.

- Apache Ant or Maven for build automation.

- Version control system (Git) for source code management.

DE-10: Third-Party APIs (Optional)

- Social media login providers (Google, Facebook, LinkedIn) if implementing OAuth integration.

- SMS gateway for mobile verification (if implementing phone number verification).

- Map services (Google Maps API) for displaying company locations.

DE-11: Browser Compatibility

- Modern browsers maintain support for HTML5, CSS3, and ES6 JavaScript features.

- Browser vendors continue security updates for their supported versions.

DE-12: Legal and Compliance

- Compliance with Vietnam's labor laws and online recruitment regulations.

- Adherence to personal data protection regulations (similar to GDPR).

- Terms of service and privacy policy reviewed by legal counsel.

## 2. Limitations & Exclusions

LIM-1: No Advanced AI-Powered Matching

- The system will NOT include sophisticated AI/ML algorithms for job-candidate matching in the initial release.

- Basic keyword-based search and filtering will be provided instead.

- Advanced recommendation systems may be considered for future versions.

LIM-2: No Video Interview Capabilities

- The platform will NOT support built-in video conferencing or virtual interview rooms.

- Users must use external tools (Zoom, Google Meet, Microsoft Teams) for interviews.

- The system may provide links to schedule external video calls but won't host them.

LIM-3: No Background Check Integration

- The system will NOT integrate with background verification or credential checking services.

- Recruiters are responsible for conducting their own due diligence on candidates.

- Document verification remains a manual process.

LIM-4: No Applicant Tracking System (ATS) Advanced Features

- The system will NOT include complex recruitment pipeline management with customizable stages.

- Basic application status tracking (Applied, Reviewed, Accepted, Rejected) will be provided.

- Advanced workflow automation is excluded from this version.

LIM-5: No Multi-Language Support

- The initial version will support Vietnamese language only.

- English language support may be considered for future releases.

- User-generated content (job descriptions, CVs) can be in any language but UI remains Vietnamese.

LIM-6: No Skills Assessment or Testing Platform

- The system will NOT include built-in skills testing or assessment capabilities.

- No coding challenges, psychometric tests, or aptitude assessments are included.

- Third-party assessment tools must be used externally.

LIM-7: No Salary Benchmarking or Analytics

- The system will NOT provide industry salary comparison data or compensation analytics.

- Salary information in job postings is optional and user-provided only.

LIM-8: No Recruitment Agency Features

- The system is designed for direct employer-to-candidate relationships.

- Multi-company management for recruitment agencies is NOT supported.

- Headhunters or staffing agencies must create separate accounts for each client company.

LIM-9: Limited Mobile Application

- No native iOS or Android applications in the initial release.

- Mobile access is provided through responsive web design only.

- Native apps may be considered for future development phases.

LIM-10: File Size and Format Restrictions

- CV uploads are limited to PDF, DOC, DOCX formats only.

- Maximum file size for CV uploads: 5MB per file.

- Image uploads for company profiles limited to JPG, PNG formats, maximum 2MB.

LIM-11: Browser Support Limitations

- Legacy browsers (Internet Explorer 11 and below) are NOT officially supported.

- Optimal experience requires modern browsers released within the last 2 years.

- Some features may degrade gracefully on older browsers but are not guaranteed to work.

LIM-12: No Offline Functionality

- The system requires active internet connection for all operations.

- No offline mode or local data caching is provided.

- Progressive Web App (PWA) features are not included in initial release.

LIM-13: Limited Real-Time Features

- No real-time chat or instant messaging between recruiters and candidates.

- Notifications are delivered via email and in-app badges (checked on page refresh).

- WebSocket or real-time communication protocols are not implemented initially.

LIM-14: API and Integration Limitations

- No public API for third-party integrations in the initial release.

- Integration with external HR systems or job boards is NOT supported.

- Data import/export must be done through provided admin interfaces only.

LIM-15: Search Result Limits

- Job search results are paginated with maximum 50 results per page.

- Advanced search filters are limited to predefined categories (location, job type, salary range).

- Free-text search may not support complex Boolean queries or advanced operators.

LIM-16: Concurrent User Limits

- The system is architected to support up to 1,000 concurrent users in the initial deployment.

- Performance beyond this threshold has not been tested and may require infrastructure upgrades.

LIM-17: Storage Limitations

- Initial storage allocation: 100GB for all CVs and company documents.

- Older files may need to be archived if storage limits are reached.