PixelCrafters

Jobify
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
<23/07/2025>	Draft	Initial draft from template	Nguyễn Lê Hồ Anh Khoa
<23/07/2025>	1.0	Fill in 1. Introduction, 2. Architectural Goals and Constraints, 3. Use case model, overview of 4. Logical View	Nguyễn Lê Hồ Anh Khoa
<23/07/2025>	1.1	Fill in 4.2 Business Layer some controllers	Bùi Minh Duy
<24/07/2025>	1.2	Fill in 4.1 Presentation Layer	Nguyễn Lê Hồ Anh Khoa
<24/07/2025>	1.3	Complete 4.2 Business Layer	Bùi Minh Duy
<27/07/2025>	1.4	Update 4.1.12: Moderation & Admin	Hình Diễm Xuân
<31/07/2025>	1.5	Update 4.1 Presentation Layer	Nguyễn Lê Hồ Anh Khoa
<31/07/2025>	1.6	Update 4.2 Business Layer	Bùi Minh Duy
<01/08/2025>	1.7	Refactor Presentation Layer	Nguyễn Lê Hồ Anh Khoa
<02/08/2025>	1.8	Update 4.2 Business Layer Complete 4.3 Data Layer Add Class Diagram	Bùi Minh Duy

Table of Contents

Software Architecture Document

1. Introduction

The introduction of the Software Architecture Document provides an overview of the entire Software Architecture Document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the Software Architecture Document.

1.1 Purpose

This Software Architecture Document (SAD) provides a comprehensive overview of the architectural design of Jobify, a web-based recruitment platform. The purpose of this document is to outline the system's major components, their interactions, and the architectural decisions made to ensure the system is scalable, secure, maintainable, and aligned with stakeholder requirements. It serves as a foundation for developers, designers, testers, and maintainers to understand and implement the system consistently.

1.2 Scope

Jobify is a multi-role recruitment web application that connects job seekers (candidates) and recruiters (HR representatives) on a unified platform. The system enables users to register, post jobs, apply for jobs, manage job applications, send notifications, and utilize a coin-based system for recruiters. Additionally, moderators can verify job postings, and administrators can broadcast system-wide messages or take managerial actions.

The system includes:

- Candidate and recruiter account management
- Job listing creation and discovery
- Application submission and status tracking
- Role-based authorization (Candidate, Recruiter, Moderator, Admin)
- Wallet and transaction handling using a virtual coin system
- Google OAuth 2.0 authentication
- AI-based job recommendation (planned)
- Notification system

1.3 Definitions, Acronyms, and Abbreviations

Term	**Description**
CRUD	Create, Read, Update, Delete – basic database operations
OAuth 2.0	Open standard for access delegation and user authentication
JWT	JSON Web Token – used for secure API authorization
RBAC	Role-Based Access Control
HR	Human Resources – recruiter role in the system
DB	Database
API	Application Programming Interface
UI	User Interface
UML	Unified Modeling Language

1.4 References

Jobify – Vision Document (v1.5)
Jobify – Use Case Specification (v1.18)
Figma UI design board (Jobify)

1.5 Overview

This document consists of the following main sections:

- Architectural Goals and Constraints: Non-functional requirements and technical limitations that influence the architecture.
- Use-Case Model: A graphical representation of the system's functional requirements.
- Logical View: A breakdown of system components and how they interact.
- Deployment View: Mapping of components to physical nodes.
- Implementation View: Overview of code structure and organization.

2. Architectural Goals and Constraints

This section describes the non-functional requirements, architectural goals, and technical constraints that significantly influence the design of Jobify.

2.1 Architectural Goals

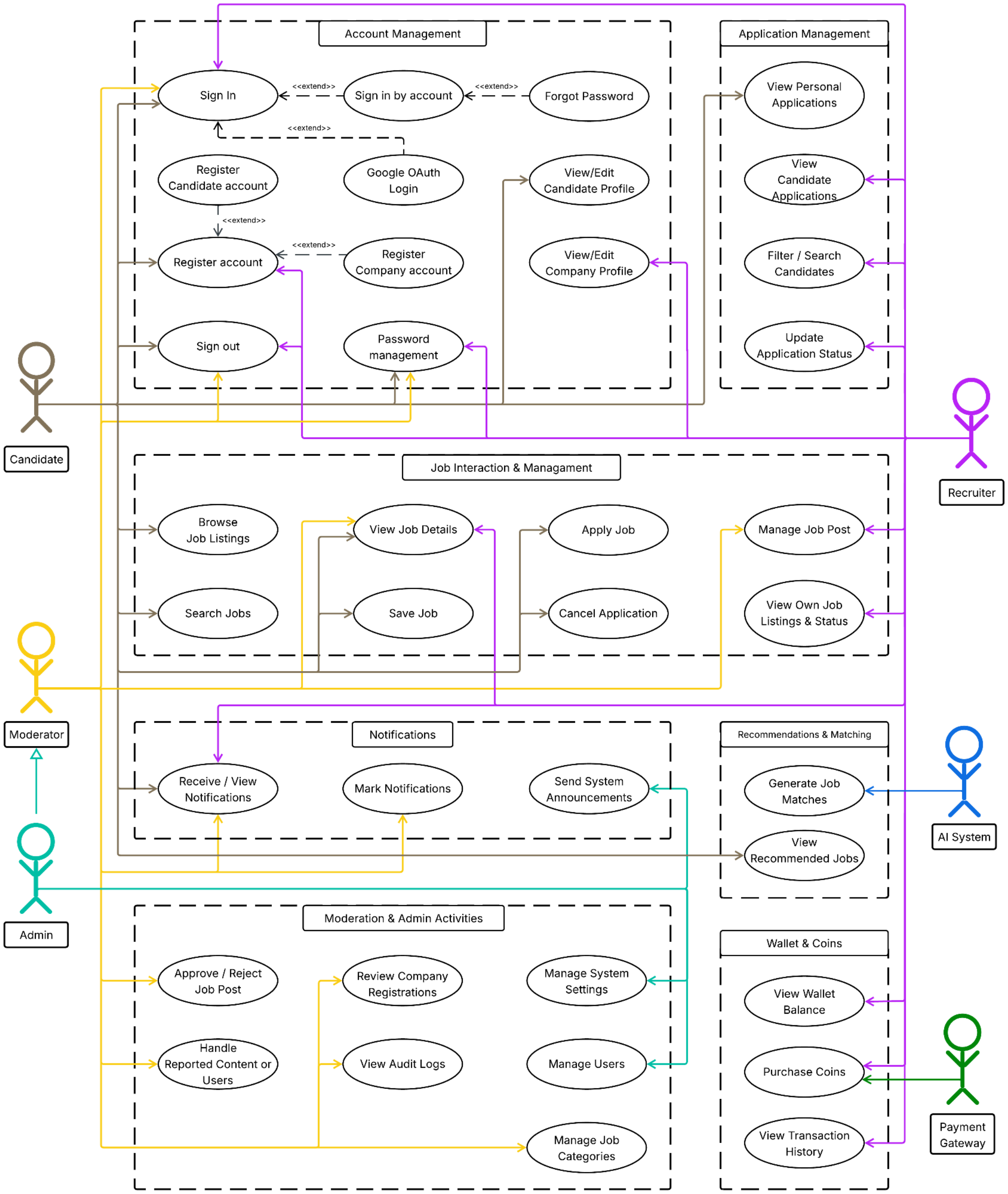
- **Web-based platform:** The system must run reliably across major browsers (Chrome, Firefox, Safari) on both desktop and mobile.
- **Scalability:** Support at least 500 concurrent users under normal usage conditions.
- **Performance:** API responses should complete within 500 milliseconds on average; page load times should not exceed 3 seconds.
- **Usability:** New users must be able to register, apply for jobs, and navigate the platform within 5 minutes without prior instructions.
- **Security:**
  - Authentication via OAuth 2.0 (Google)
  - Encrypted passwords (if not using external login)
  - Role-Based Access Control (RBAC) between Candidate, Recruiter, Moderator, and Admin
  - Secure communication using HTTPS
- **Maintainability:** Code should follow modular and layered architecture (MVC-like). Backend and frontend should be loosely coupled.
- **Extensibility:** Future support for AI-based job recommendations and mobile responsiveness should be considered in the design.
- **Reusability:** Generic components (e.g., job cards, pagination, modal dialogs) should be reusable across different views and user roles.
- **Transaction Logic**: The wallet and coin system must ensure transactional integrity. All coin-related operations must be atomic and avoid inconsistent states in case of failure

2.2 Technical Constraints

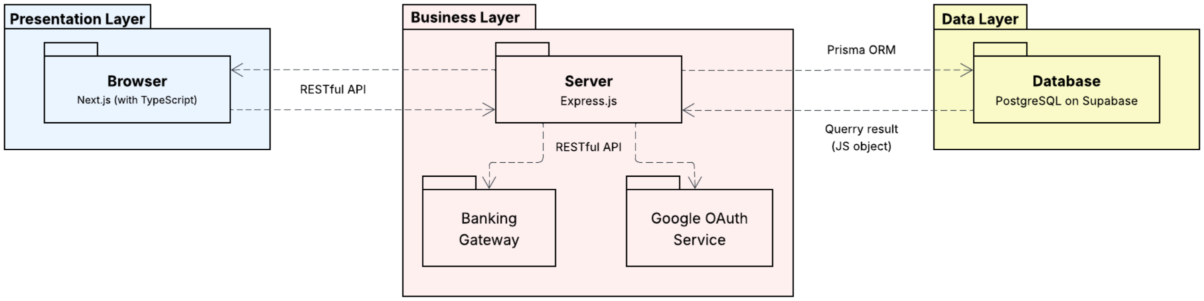
- **Frontend:** Implemented using React + Next.js with the App Router and TailwindCSS for styling and deployed on Vercel.
- **Backend:** Built using Node.js + Express, hosted on cloud infrastructure and deployed on Render.
- **Database:** Uses PostgreSQL via Supabase, a hosted backend-as-a-service that provides instant APIs, auth, and storage.
- **Authentication:** Must be implemented using Google OAuth 2.0, with JWT for session management.
- **Communication protocol:** RESTful APIs over HTTPS between frontend and backend.
- **Deployment model:** Backend and database run on cloud; frontend served statically (Next.js pre-rendering where possible).
- **Development team:** Comprised of 4 members with defined roles (Frontend, Backend, UI/UX, QA); development is done using Git-based collaboration.

- **Project schedule:** Design must be deliverable in phases, aligning with incremental PA submissions.

3. Use-Case Model



4. Logical View

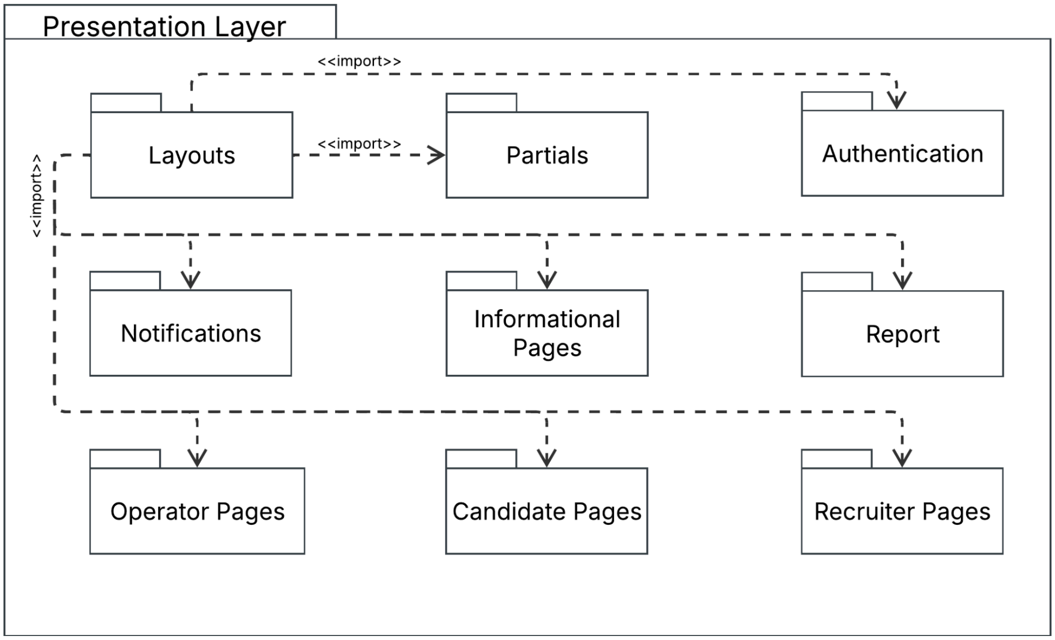


The system architecture of Jobify follows a 3-tier decoupled web architecture, consisting of:

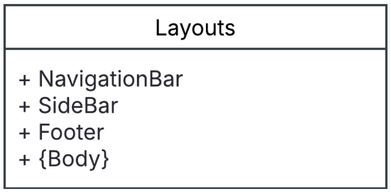
- **Presentation Layer:** Built using Next.js (with TypeScript), a React-based framework responsible for rendering the user interface and interacting with the backend via RESTful API.
- **Business Layer:** Implemented with Express.js, which handles routing, authentication, business logic, and communication with the database.
  - This layer also integrates with:
  - Google OAuth 2.0: for user authentication and account linking.
  - AI Recommendation Service: to fetch personalized job suggestions based on candidate profiles.
  - All communication between frontend and backend uses RESTful APIs
- **Data Layer:** Hosted on Supabase, which provides a cloud-based PostgreSQL database and integrated authentication and storage services.

This separation of concerns allows for high scalability, independent deployment of frontend/backend, easier testing, and potential replacement of individual layers without breaking the entire system.

4.1 Presentation Layer

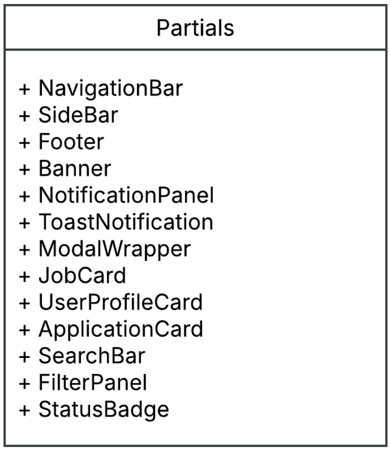


4.1.1 Layouts



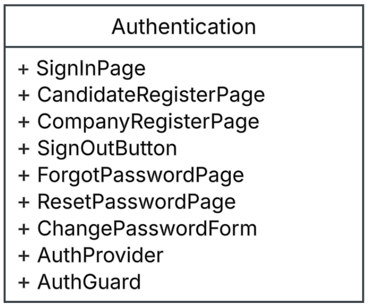
- The overall page framework, responsible for importing components like NavigationBar, SideBar, Footer, and rendering the {Body} content of each specific page.

4.1.2 Partials (Reusable Components)



- NavigationBar: The main navigation bar, containing logo, menus (depending on user role), profile, and notifications.
- SideBar: The secondary navigation bar, often used for dashboards or admin panels.
- Footer: The page footer, containing contact info and links to auxiliary pages (About Us, Support, etc.).
- Banner: Displays announcements, ads, or highlighted information.
- NotificationPanel: A quick-access panel showing the latest notifications.
- ToastNotification: Small pop-up notifications for events (success, error, etc.).
- ModalWrapper: A modal container for pop-ups, forms, confirmations, etc.
- JobCard: A compact card displaying summary information about a job (used in job listings).
- UserProfileCard: A compact card showing candidate information (used in candidate lists or when recruiters view profiles).
- ApplicationCard: A compact card showing application information (used in application lists).
- SearchBar: A search bar for jobs, candidates, etc.
- FilterPanel: A filter panel for refining search results.
- StatusBadge: Displays status (e.g., Applied, Approved, Rejected, etc.).

4.1.3 Authentication

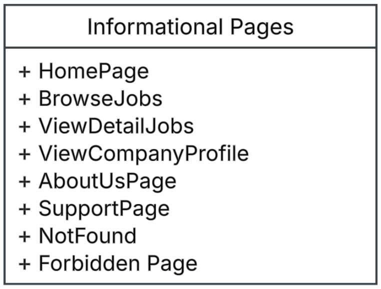


Authentication

+ SignInPage
+ CandidateRegisterPage
+ CompanyRegisterPage
+ SignOutButton
+ ForgotPasswordPage
+ ResetPasswordPage
+ ChangePasswordForm
+ AuthProvider
+ AuthGuard

- SignInPage: Login form for all users (candidate, recruiter, moderator, admin)
- CandidateRegisterPage: Registration page for candidate accounts.
- CompanyRegisterPage: Registration page for company accounts.
- SignOutButton: Logout button.
- ForgotPasswordPage: Form to request an email to reset password.
- ResetPasswordPage: Access by url received via email with token.
- ChangePasswordForm: Form for changing password when already logged in.
- AuthProvider: Provides authentication context for the app (commonly with React Context/Provider).
- AuthGuard: Route guard, only allows access if the user is authenticated/authorized.

4.1.4 Informational Pages

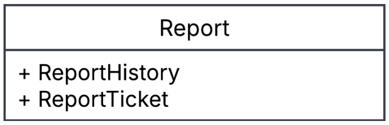


Informational Pages

+ HomePage
+ BrowseJobs
+ ViewDetailJobs
+ ViewCompanyProfile
+ AboutUsPage
+ SupportPage
+ NotFound
+ Forbidden Page

- HomePage: Page to view newest jobs and top industry on platform.
- BrowseJobs (including SearchJob): Page to browse and search job listings with filters and keywords.
- ViewDetailJobs: Displays full job details including description, requirements, benefits, and apply button.
- ViewCompanyProfile: Shows public company profile including logo, description, website, and active job posts.
- AboutUsPage: About page for Jobify.
- SupportPage: Support, contact and FAQ page.
- NotFound: 404 page shown when route or resource is not found.
- Forbidden Page: 403 page shown when the user is unauthorized to access the requested content.

4.1.5 Report



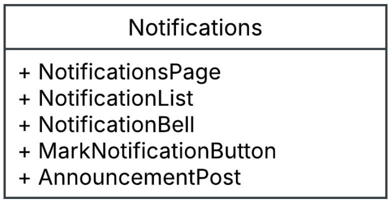
Report

+ ReportHistory
+ ReportTicket

- ReportHistory: Displays a list of all reports submitted by the user (candidates or recruiters).

- ReportTicket: Shows detailed information of a specific report, including target entity, reason, status, and resolution (if reviewed).

4.1.6 Notifications

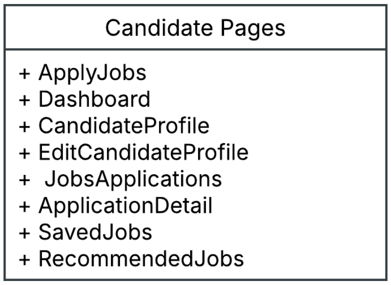


Notifications

+ NotificationsPage
+ NotificationList
+ NotificationBell
+ MarkNotificationButton
+ AnnouncementPost

- NotificationsPage: Page listing all notifications.
- NotificationList: List of notifications.
- NotificationBell: Notification bell icon (usually in the NavigationBar).
- MarkNotificationButton: Button to mark notifications as read or dismiss them.
- AnnouncementPost: Displays a system announcement.

4.1.7 Candidate Pages

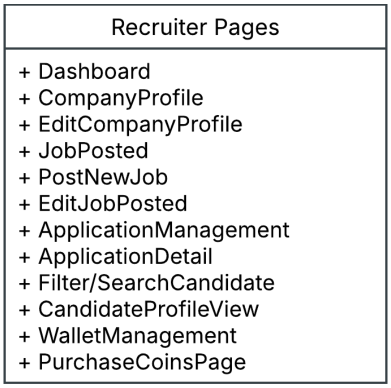


Candidate Pages

+ ApplyJobs
+ Dashboard
+ CandidateProfile
+ EditCandidateProfile
+ JobsApplications
+ ApplicationDetail
+ SavedJobs
+ RecommendedJobs

- ApplyJobs: Form or action to submit a job application with resume and optional cover letter.
- Dashboard: Candidate’s personal dashboard showing overview information, recent jobs and newest notifications.
- CandidateProfile: View mode of the candidate’s full profile for self.
- EditCandidateProfile: Form to edit personal info, skills, experience, education, etc.
- JobsApplications: List of all job applications submitted by the candidate.
- ApplicationDetail: Detailed view of a specific application, including status, cover letter and resume.
- SavedJobs: List of jobs the candidate has saved.
- RecommendedJobs: Showing jobs recommended for the candidate.

4.1.8 Recruiter Pages

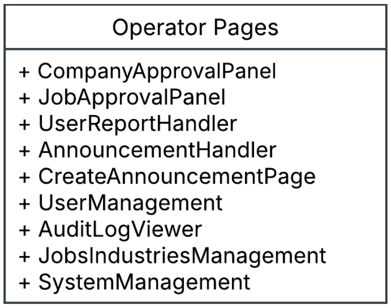


Recruiter Pages

+ Dashboard
+ CompanyProfile
+ EditCompanyProfile
+ JobPosted
+ PostNewJob
+ EditJobPosted
+ ApplicationManagement
+ ApplicationDetail
+ Filter/SearchCandidate
+ CandidateProfileView
+ WalletManagement
+ PurchaseCoinsPage

- Dashboard: Recruiter dashboard showing job stats, applications, and company overview.
- CompanyProfile: View mode of the company’s full profile for self.
- EditCompanyProfile: Form to edit company details.
- JobPosted: List of jobs posted by the recruiter with status, stats, and management actions.
- PostNewJob: Form to create a new job posting.
- EditJobPosted: Form to update an existing job post.
- ApplicationManagement: Manage job applications for posted jobs, including status updates and review.
- ApplicationDetail: View details of a specific application submitted to a job post.
- Filter/SearchCandidate: Filter and search through candidates who applied, based on criteria like skills or experience.
- CandidateProfileView: View full profile of a specific applicant as a recruiter.
- WalletManagement: View and manage company coin balance.
- PurchaseCoinsPage: Page to purchase additional coins for job posting or platform features.

4.1.9 Operator Pages

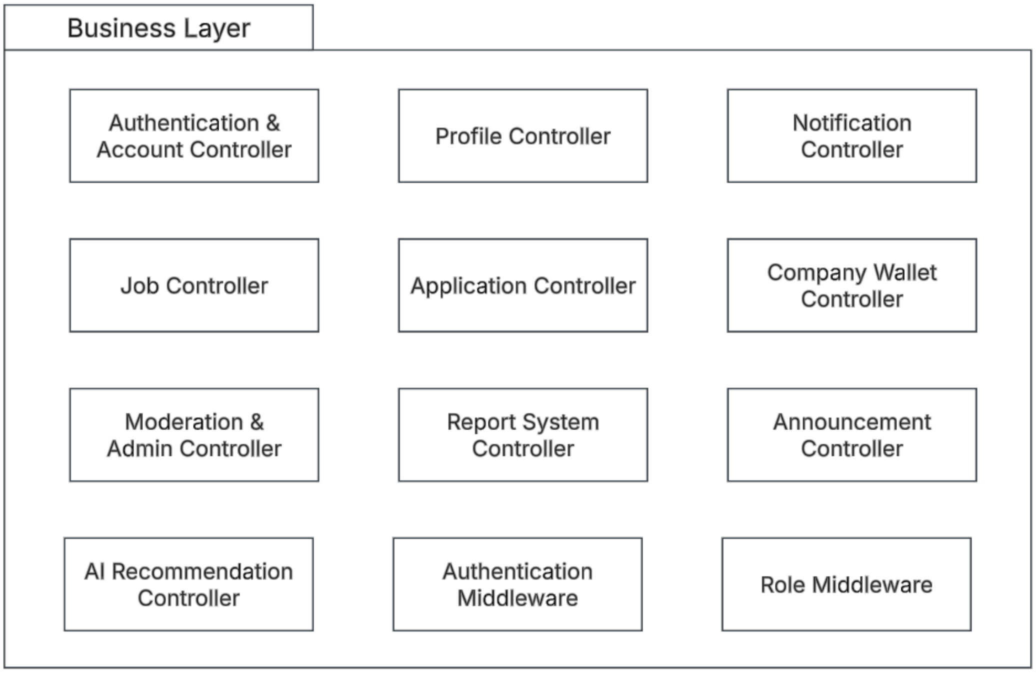


Operator Pages

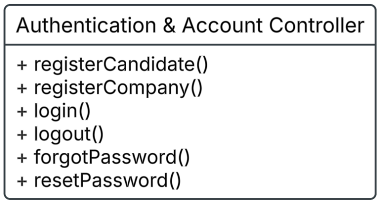
+ CompanyApprovalPanel
+ JobApprovalPanel
+ UserReportHandler
+ AnnouncementHandler
+ CreateAnnouncementPage
+ UserManagement
+ AuditLogViewer
+ JobsIndustriesManagement
+ SystemManagement

- CompanyApprovalPanel: Panel for approving registration of companies.
- JobApprovalPanel: Panel for approving job post.
- UserReportHandler: Handles user/content reports.
- AnnouncementHandler: Admin panel for viewing, managing, and scheduling system announcements.
- CreateAnnouncementPage: Form to create a system-wide announcement for users.
- UserManagement: Admin interface for managing user accounts, roles, status (e.g., ban, activate).
- AuditLogViewer: Views system audit logs.
- JobsIndustriesManagement: CRUD management of job industry tags.
- SystemManagement: Form for configuring system settings.

4.2 Business Layer



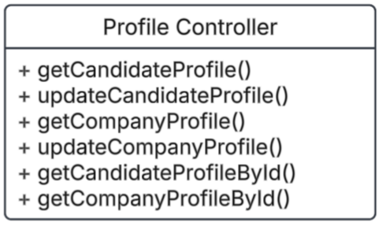
4.2.1 Authentication & Account Controller



Description: Manages all account-related actions such as user registration, login, logout, and password recovery. Provides secure authentication and access management for both candidates and company recruiters, with separate registration flows for individual users and companies.

- registerCandidate(): Register a new user account as a candidate.
- registerCompany(): Register a new user account as a company recruiter, including company details.
- login(): Authenticate users with valid credentials.
- logout(): Log out the currently authenticated user and invalidate their session.
- forgotPassword(): Send a recovery email with a token to reset the password.
- resetPassword(): Reset the password using the provided verification token.

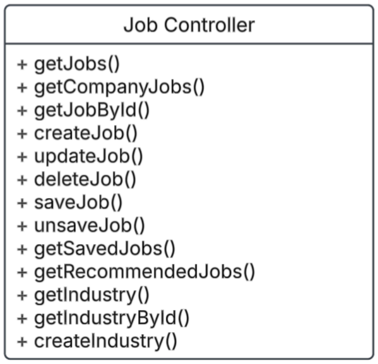
4.2.2 Profile Controller (Candidate & Company)



Description: Handles profile information for both candidates and companies. Allows users to view and update their own profiles, as well as view public profiles of companies.

- getCandidateProfile(): Retrieve the current candidate's profile information.
- updateCandidateProfile(): Update personal details of the logged-in candidate.
- getCompanyProfile(): Retrieve the current company's profile information.
- updateCompanyProfile(): Update company profile details of the logged-in recruiter.
- getCandidateProfileById(): View a candidate’s profile in the applicant list by their unique ID.
- getCompanyProfileById(): View a company’s public profile by their unique ID.

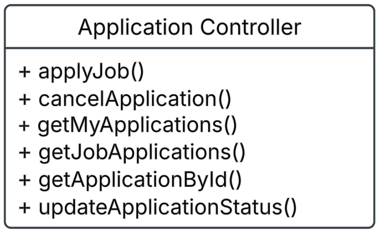
4.2.3 Job Controller



Description: Handles all operations related to job postings, job discovery, and candidate interactions such as saving jobs and browsing personalized job recommendations. Supports job management for recruiters and facilitates job exploration for candidates across various industries.

- getJobs(): Browse all available jobs with filtering and pagination options.
- getCompanyJobs(): Retrieve all job posts created by the currently authenticated recruiter.
- getJobById(): Get full details of a specific job post by its ID.
- createJob(): Recruiter posts a new job opening.
- updateJob(): Edit details of an existing job post.
- deleteJob(): Remove a job listing from the system.
- saveJob(): Save a job post to the current candidate's saved list.
- unsaveJob(): Remove a job post from the saved list.
- getSavedJobs(): View all jobs that the current candidate has saved.
- getRecommendedJobs(): Show personalized job recommendations based on user profile.
- getIndustry(): Retrieve a list of all job industries.
- getIndustryById(): Get details of a specific industry by its ID.
- createIndustry(): Add a new industry to the system.

4.2.4 Application Controller

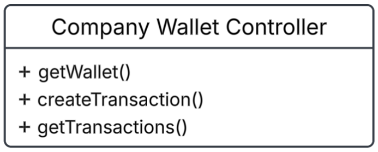


Description: Handles all operations related to submitting, managing, and reviewing job applications from both candidate and recruiter perspectives. Supports application submissions, status tracking, and managing applicant lists for job postings.

- applyJob(): Candidates apply to a specific job posting.
- cancelApplication(): Withdraw a previously submitted job application.

- getMyApplications(): Retrieve a list of all job applications submitted by the current candidate.
- getJobApplications(): Recruiters view all applications submitted for a specific job they posted.
- getApplicationById(): View detailed information of a specific application (accessible by both candidate and recruiter).
- updateApplicationStatus(): Recruiters update the application status (e.g., reviewing, accepted, rejected).

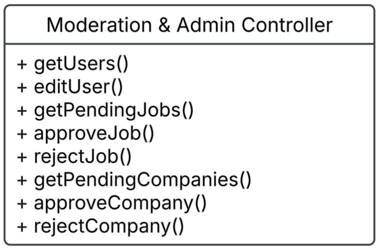
4.2.5 Company Wallet Controller



Description: Handles wallet operations for companies, including coin transactions and balance management.

- getWallet(): Retrieve the current coin balance of the company.
- createTransaction(): Deposit or spend coins from the wallet.
- getTransactions(): View transaction history of the company.

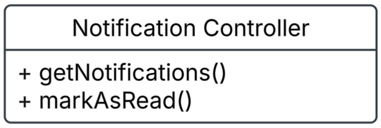
4.2.6 Moderation & Admin Controller



Description: Grants moderators and administrators control over platform content and user management. Handles reviewing and moderating pending job posts and company registrations, as well as managing user accounts.

- getUsers(): Retrieve a list of all user accounts with filtering and pagination options.
- editUser(): Update or modify user account information and roles.
- getPendingJobs(): Retrieve a list of job posts pending moderation approval.
- approveJob(): Approve a submitted job post and make it publicly visible.
- rejectJob(): Reject a submitted job post and optionally provide feedback to the recruiter.
- getPendingCompanies(): Retrieve a list of companies awaiting registration approval.
- approveCompany(): Approve a company's registration and activate their account.
- rejectCompany(): Reject a company's registration request with feedback or reasons.

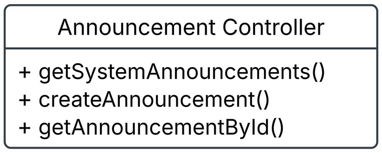
4.2.7 Notification Controller



Description: Handles user-specific notifications generated by system events, such as job application updates or account actions. It ensures users stay informed of activities relevant to them.

- getUserNotifications(): List notifications for the current user.
- markAsRead(): User marks a notification as read.

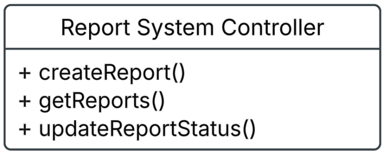
4.2.8 Announcement Controller



Description: Manages system-wide announcements created by administrators. Announcements are visible to all or specific user roles and used for broadcasting important messages like updates or platform notices.

- getSystemAnnouncements(): View all system-wide announcements.
- createAnnouncement(): Admin creates a new announcement.
- getAnnouncementDetail(): View the content of a specific announcement.

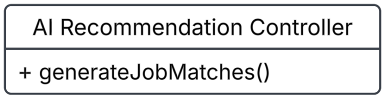
4.2.9 Report System Controller



Description: Allows candidates to report inappropriate jobs or companies and enables moderators/admins to manage these reports.

- createReport(): Submit a report about a job post or company.
- getReports(): Admin or moderator views submitted reports.
- updateReportStatus(): Update the review status of a report.

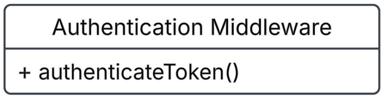
4.2.10 AI Recommendation Controller



Description: Provides personalized job recommendations for candidates by interacting with an internal AI/ML service that calculates relevance scores. This controller is responsible for initiating the matching process, ensuring candidates receive up-to-date job suggestions based on their profile.

- generateJobMatches(): Trigger the job matching algorithm for the current candidate. This process recalculates relevance scores and updates personalized job recommendations based on the latest profile data.

4.2.11 Authentication Middleware



Description: Handles JWT-based authentication to ensure only logged-in users can access protected routes. Extracts and verifies the access token from the Authorization header.

- authenticateToken(): Validates JWT token, decodes user information, and attaches it to the request object. Returns 401 if no token is provided and 403 if the token is invalid or expired.

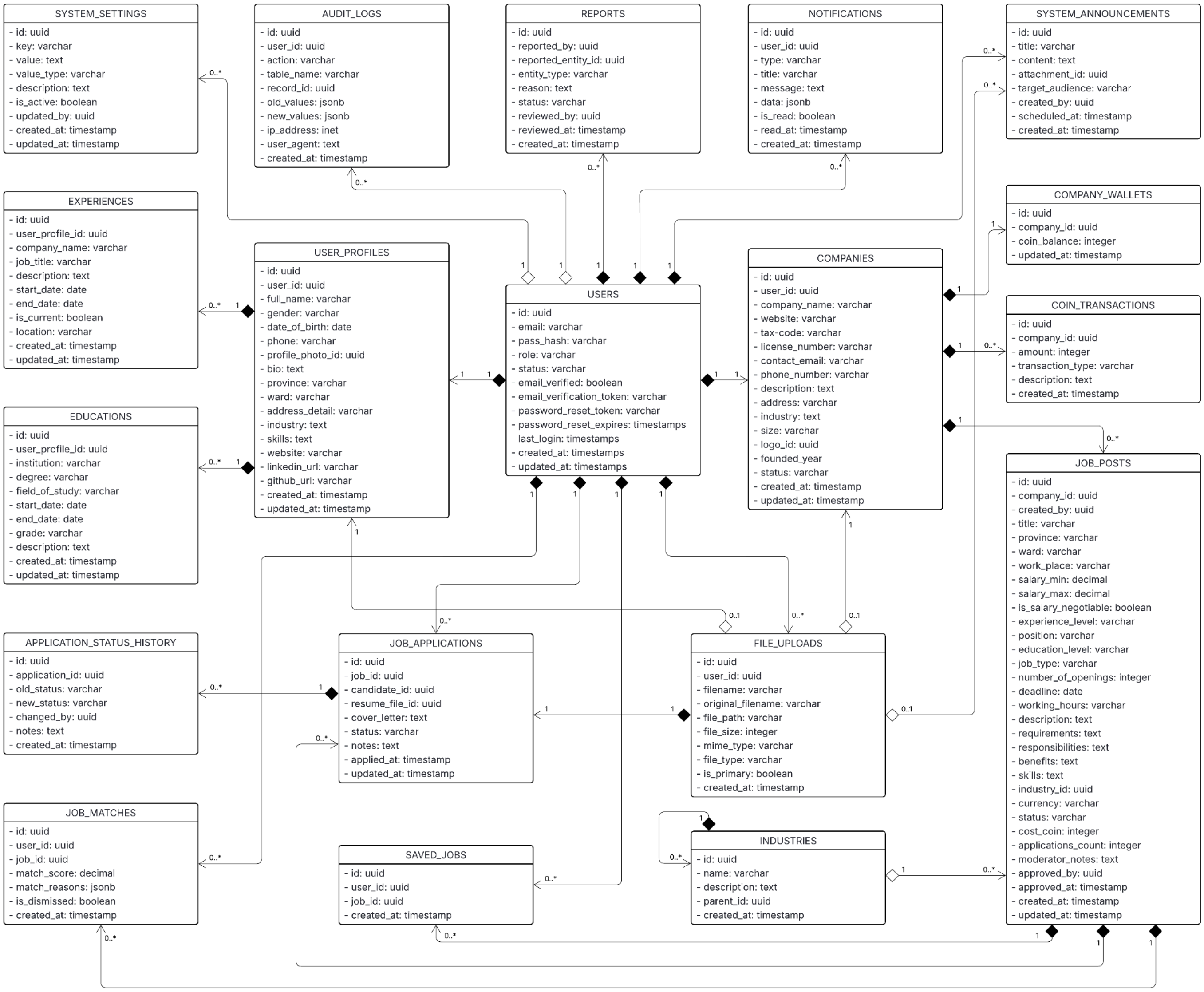
4.2.12 Role Middleware

Role Middleware
+ authorizeRoles()

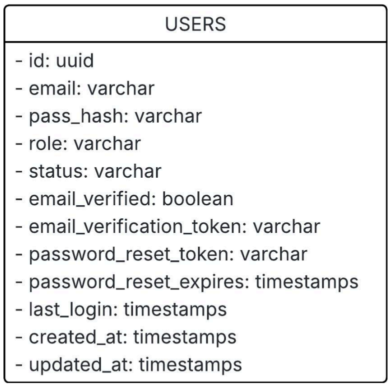
Description: Role-based access control middleware to restrict route access to users with specific roles (e.g., admin, recruiter, candidate).

- authorizeRoles(): Checks if the authenticated user's role is included in the list of allowed roles. Returns 401 if the user role is missing and 403 if the user does not have the necessary permissions.

4.3 Data Layer



4.3.1 Users:

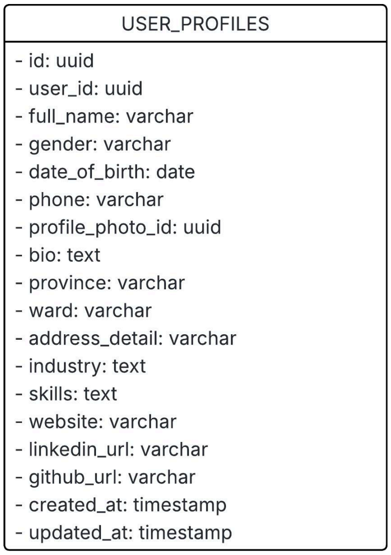


USERS

- id: uuid
- email: varchar
- pass_hash: varchar
- role: varchar
- status: varchar
- email_verified: boolean
- email_verification_token: varchar
- password_reset_token: varchar
- password_reset_expires: timestamps
- last_login: timestamps
- created_at: timestamps
- updated_at: timestamps

Description: Manages core authentication and identity information for all platform users, including candidates, recruiters, and administrators. Stores login credentials, account status, and security tokens.

4.3.2 User_profiles:

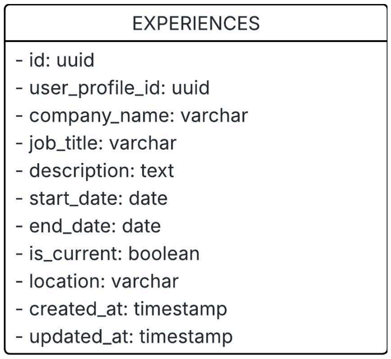


USER_PROFILES

- id: uuid
- user_id: uuid
- full_name: varchar
- gender: varchar
- date_of_birth: date
- phone: varchar
- profile_photo_id: uuid
- bio: text
- province: varchar
- ward: varchar
- address_detail: varchar
- industry: text
- skills: text
- website: varchar
- linkedin_url: varchar
- github_url: varchar
- created_at: timestamp
- updated_at: timestamp

Description: Contains detailed profile information for individual users such as personal details, contact info, biography, and links to portfolios or social profiles. Linked one-to-one with *users*.

4.3.3 Experiences:

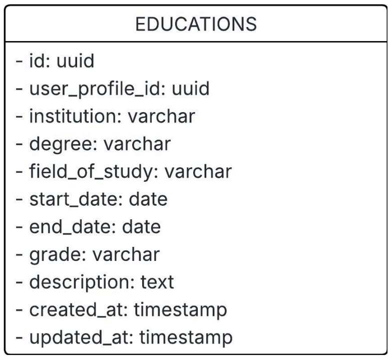


EXPERIENCES

- id: uuid
- user_profile_id: uuid
- company_name: varchar
- job_title: varchar
- description: text
- start_date: date
- end_date: date
- is_current: boolean
- location: varchar
- created_at: timestamp
- updated_at: timestamp

Description: Records a user's professional work history including company names, job titles, descriptions, and employment duration. Part of the *user_profiles*.

4.3.4 Educations:

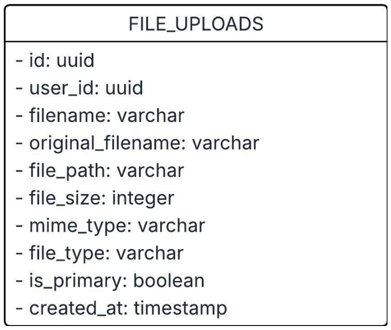


EDUCATIONS

- id: uuid
- user_profile_id: uuid
- institution: varchar
- degree: varchar
- field_of_study: varchar
- start_date: date
- end_date: date
- grade: varchar
- description: text
- created_at: timestamp
- updated_at: timestamp

Description: Stores educational background of users, including institution details, degrees, and study periods. Part of the *user_profiles*.

4.3.5 File_uploads:

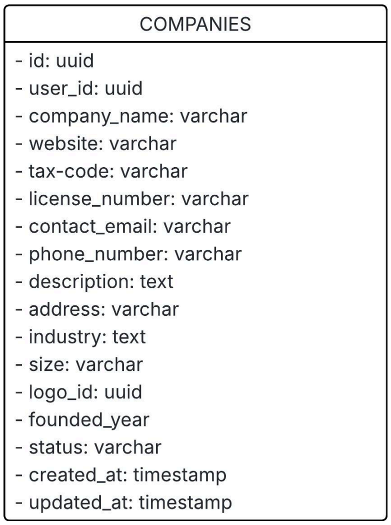


FILE_UPLOADS

- id: uuid
- user_id: uuid
- filename: varchar
- original_filename: varchar
- file_path: varchar
- file_size: integer
- mime_type: varchar
- file_type: varchar
- is_primary: boolean
- created_at: timestamp

Description: Manages user-uploaded files like resumes, profile photos, and attachments in announcements. Supports various file types and usage contexts.

4.3.6 Companies:

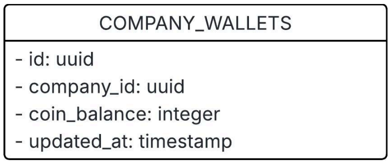


COMPANIES

- id: uuid
- user_id: uuid
- company_name: varchar
- website: varchar
- tax-code: varchar
- license_number: varchar
- contact_email: varchar
- phone_number: varchar
- description: text
- address: varchar
- industry: text
- size: varchar
- logo_id: uuid
- founded_year
- status: varchar
- created_at: timestamp
- updated_at: timestamp

Description: Represents recruiter organizations on the platform. Contains company-specific details like name, industry, license info, size, and description. Each company is owned by a user.

4.3.7 Company_wallets:

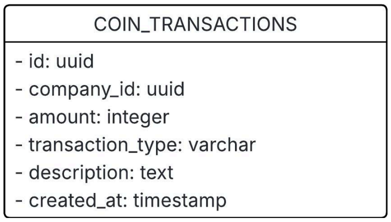


COMPANY_WALLETS

- id: uuid
- company_id: uuid
- coin_balance: integer
- updated_at: timestamp

Description: Represents coin-based credit balances for companies used for transactions like posting jobs.

4.3.8 Coin_transcations:

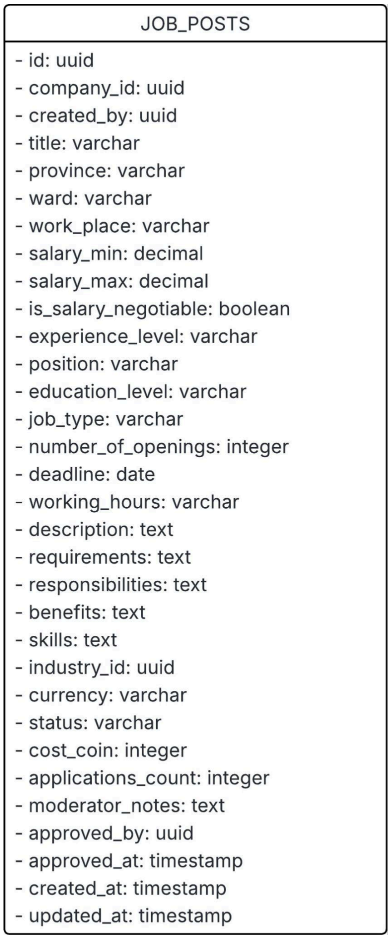


COIN_TRANSACTIONS

- id: uuid
- company_id: uuid
- amount: integer
- transaction_type: varchar
- description: text
- created_at: timestamp

Description: Tracks coin-related transactions in company wallets, including purchasing credits and system deductions for job posts.

4.3.9 Job_posts:

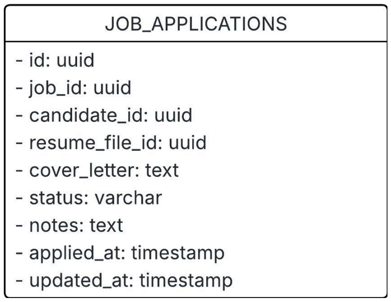


JOB_POSTS

- id: uuid
- company_id: uuid
- created_by: uuid
- title: varchar
- province: varchar
- ward: varchar
- work_place: varchar
- salary_min: decimal
- salary_max: decimal
- is_salary_negotiable: boolean
- experience_level: varchar
- position: varchar
- education_level: varchar
- job_type: varchar
- number_of_openings: integer
- deadline: date
- working_hours: varchar
- description: text
- requirements: text
- responsibilities: text
- benefits: text
- skills: text
- industry_id: uuid
- currency: varchar
- status: varchar
- cost_coin: integer
- applications_count: integer
- moderator_notes: text
- approved_by: uuid
- approved_at: timestamp
- created_at: timestamp
- updated_at: timestamp

Description: Represents job listings created by companies. Contains job necessary details (like title, salary range, job description, etc) and current moderation status.

4.3.10 Job_applications:

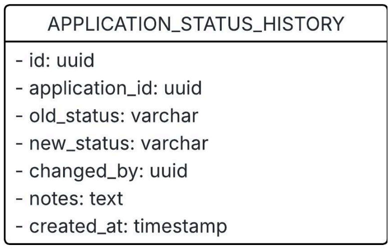


JOB_APPLICATIONS

- id: uuid
- job_id: uuid
- candidate_id: uuid
- resume_file_id: uuid
- cover_letter: text
- status: varchar
- notes: text
- applied_at: timestamp
- updated_at: timestamp

Description: Tracks job applications submitted by candidates. Stores application content, resume references, statuses, and submission timestamps.

4.3.11 Application_status_history:

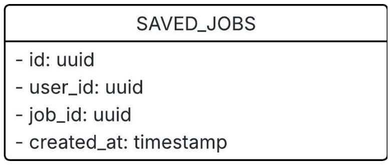


APPLICATION_STATUS_HISTORY

- id: uuid
- application_id: uuid
- old_status: varchar
- new_status: varchar
- changed_by: uuid
- notes: text
- created_at: timestamp

Description: Logs historical status changes of job applications, including who made the change and any notes attached. Supports application lifecycle tracking.

4.3.12 Saved_jobs:

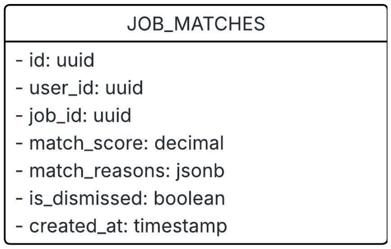


SAVED_JOBS

- id: uuid
- user_id: uuid
- job_id: uuid
- created_at: timestamp

Description: Stores records of jobs that candidates have saved/bookmarked for later reference.

4.3.13 Job_matches:

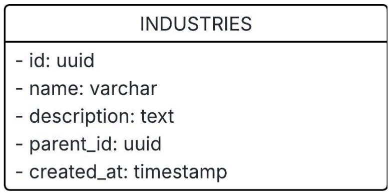


JOB_MATCHES

- id: uuid
- user_id: uuid
- job_id: uuid
- match_score: decimal
- match_reasons: jsonb
- is_dismissed: boolean
- created_at: timestamp

Description: Contains AI-generated job match results for candidates, including relevance scores and reasons. Supports job recommendation features.

4.3.14 Industries:

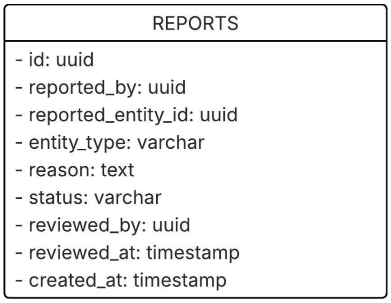


INDUSTRIES

- id: uuid
- name: varchar
- description: text
- parent_id: uuid
- created_at: timestamp

Description: Defines job industry categories for classifying job posts and companies. Supports hierarchical relationships for parent-child industry groupings.

4.3.15 Reports:

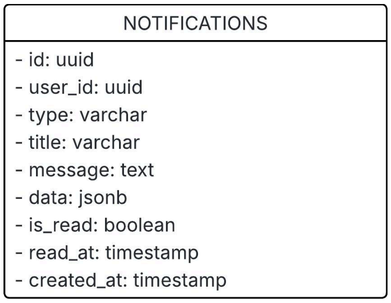


REPORTS

- id: uuid
- reported_by: uuid
- reported_entity_id: uuid
- entity_type: varchar
- reason: text
- status: varchar
- reviewed_by: uuid
- reviewed_at: timestamp
- created_at: timestamp

Description: Handles user-submitted reports regarding inappropriate content or behavior. Tracks review status and moderation actions taken on reports.

4.3.16 Notifications:

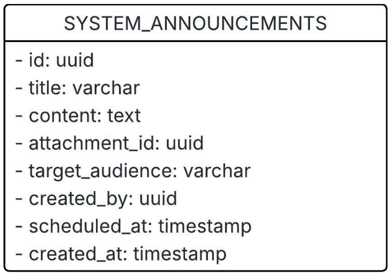


NOTIFICATIONS

- id: uuid
- user_id: uuid
- type: varchar
- title: varchar
- message: text
- data: jsonb
- is_read: boolean
- read_at: timestamp
- created_at: timestamp

Description: Manages user-specific notifications for system events, actions, and alerts. Stores message content, type, and read status.

4.3.17 System_announcements:

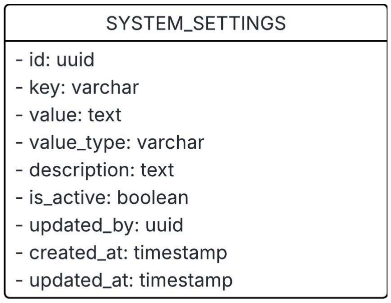


SYSTEM_ANNOUNCEMENTS

- id: uuid
- title: varchar
- content: text
- attachment_id: uuid
- target_audience: varchar
- created_by: uuid
- scheduled_at: timestamp
- created_at: timestamp

Description: Manages platform-wide announcements sent to users, with optional file attachments. Supports targeting specific user groups.

4.3.18 System_settings:

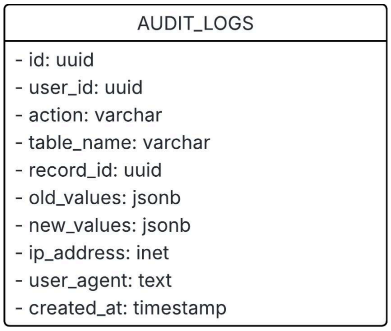


SYSTEM_SETTINGS

- id: uuid
- key: varchar
- value: text
- value_type: varchar
- description: text
- is_active: boolean
- updated_by: uuid
- created_at: timestamp
- updated_at: timestamp

Description: Stores global platform configurations and settings, such as feature toggles and system parameters, modifiable by admins.

4.3.19 Audit_logs:

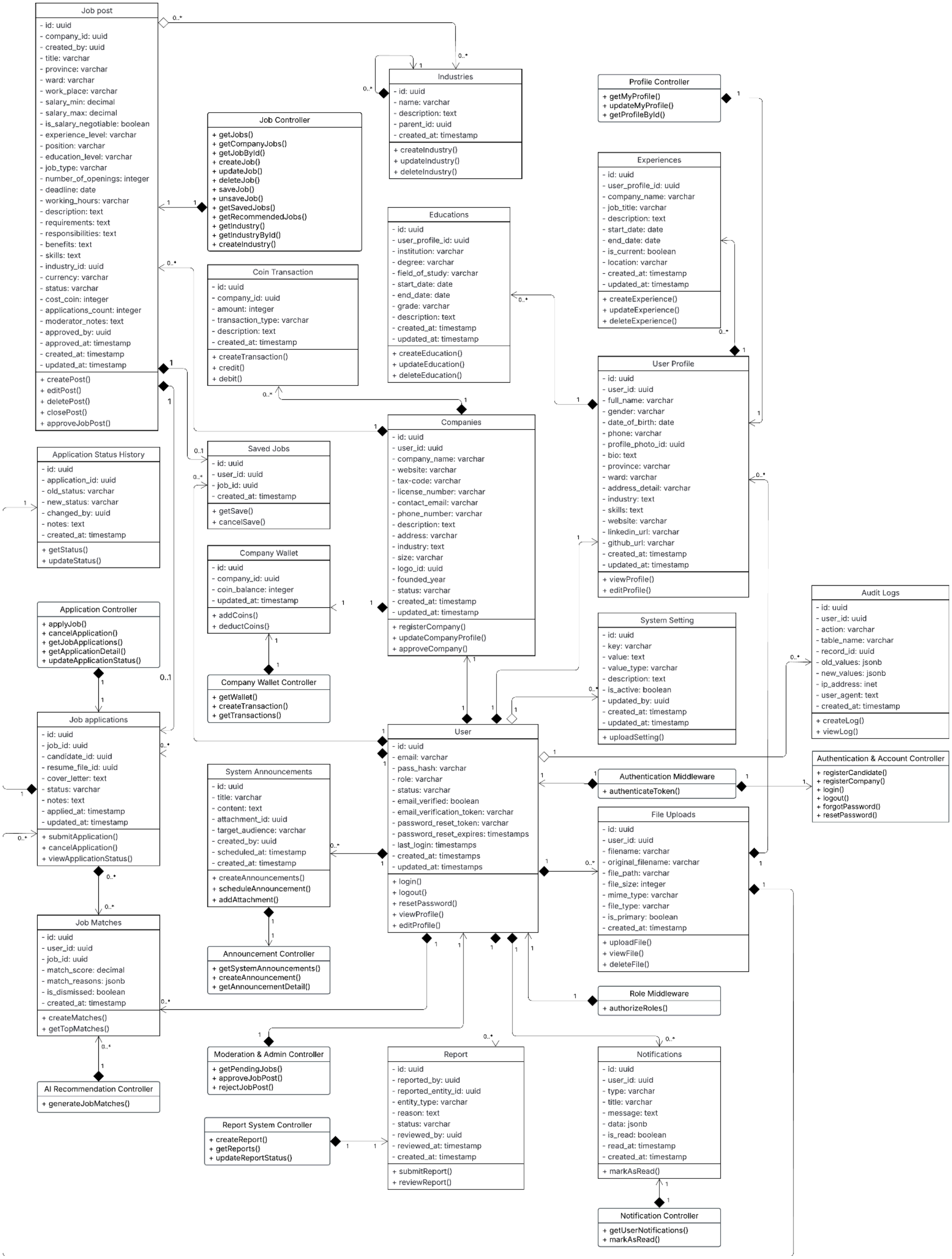


AUDIT_LOGS

- id: uuid
- user_id: uuid
- action: varchar
- table_name: varchar
- record_id: uuid
- old_values: jsonb
- new_values: jsonb
- ip_address: inet
- user_agent: text
- created_at: timestamp

Description: Records system activity logs for auditing purposes, including changes to data records, user actions, and access metadata like IP addresses.

4.4 Class Diagram



5. Deployment

[Leave this section blank for PA4.

In this section, describe how the system is deployed by mapping the components in Section 4 to machines running them. For example, your mobile app is running on a mobile device (Android, iOS, etc), your server runs all components on the server side including the database]

6. Implementation View

[Leave this section blank for PA4.

In this section, provide folder structures for your code for all components described in Section 4.]