Jovio

Backend System Requirements Specification

This document is a personal planning blueprint for the Jovio project. It is created before development to organize my ideas and clearly define what I intend to build. The goal is to have a clear roadmap that guides the development process and avoids randomness.

The project is inspired by the final exam of the Node.js course I completed at **Route Training Center,** with additional features and enhancements beyond the original scope. Jovio is a training project aimed at strengthening my backend skills, with a primary focus on NestJS and building a well-structured, scalable system**.**

Created by: **Dev Mahmoud Zain**
Backend Developer

Feel free to reach out

Email: dev.mahmoud.zain@gmail.com

Whatsapp: 01551975456

1-Collections

- User Collection
- User Settings Collection
- OTP Collection
- JWT Collection
- Company Collection
- Company Settings Collection
- Job opportunity Collection
- Application Collection
- Chat Collection
- Notification Collection

User Collection

Field Name	Type	Description
firstName	string	The user's first name.
lastName	string	The user's last name.
email	string	The primary email address used for login and notifications.
emailConfirmedAt	Date (Optional)	Timestamp indicating when the user verified their email.
newEmail	string (Optional)	Stores a pending email address if the user is in the process of changing it.
phoneNumber	string (Optional)	The user's contact phone number.
provider	ProviderEnum	The authentication provider used (e.g., google, system).
role	RoleEnum	The access level assigned to the user (e.g., admin, user).
status	UserStatusEnum	The professional or employment status of the user.
bio	string (Optional)	A short personal biography or description.
socialLinks	I_UserSocial	An object containing links to social media profiles.
password	string	The hashed password for system authentication.
gender	GenderEnum	The user's gender.
dateOfBirth	Date	The user's date of birth.
changeCredentialsTime	Date	Timestamp of the last time the user updated their credentials.
profilePicture	I_File	Object containing the URL and ID for the profile image.
coverPicture	I_File	Object containing the URL and ID for the cover/banner image.
gallery	I_File[]	An array of files representing the user's image gallery.
resume	I_File	An array of files representing the user's uploaded resumes/CVs.
skills	string[]	List Of User Skills (e.g, Node.js , Nest js)
bannedAt	Date (Optional)	Timestamp of when the user was banned.
bannedUntil	Date (Optional)	The date when the user's ban is set to expire.
bannedReason	string (Optional)	The reason provided for the account ban.
bannedBy	ObjectId (Optional)	Reference ID of the administrator who performed the ban.
freezedAt	Date (Optional)	Timestamp of when the account was frozen (temporary suspension).
freezedUntil	Date (Optional)	The date when the account freeze is set to

		expire.
freezedReason	string (Optional)	The reason provided for freezing the account.
freezedBy	ObjectId (Optional)	Reference ID of the administrator who froze the account.
restoredAt	Date (Optional)	Timestamp of when a banned/frozen account was reactivated.
RestoredBy	ObjectId (Optional)	Reference ID of the administrator who restored the account.

Notes :

- Create an index on userId + isUsed + expiresAt to speed up lookup and verification.

- Do not delete the OTP immediately after it expires; it may be needed for security analysis.

User Settings Collection

Field Name	Type	Description
userId	ObjectId	Reference to the user this settings document belongs to
is2FAEnabled	boolean	Indicates whether two-factor authentication is enabled
backupCodes	string[]	Encrypted backup codes used for account recovery
activeSessions	object[]	Active user sessions (device, ipAddress, lastActiveAt)
notifyOnNewDeviceLogin	boolean	Sends notification when login occurs from a new device
loginBlockedUntil	Date	Timestamp until login is temporarily blocked
profileVisibility	enum	Profile visibility level (public / companies-only / private)
hideEmail	boolean	Hides user email from public and company views
hidePhone	boolean	Hides user phone number from public and company views
emailNotifications	boolean	Enables or disables email notifications
createdAt	Date	Timestamp when settings were created
updatedAt	Date	Timestamp when settings were last updated

Notes :

- **Brute Force Protection**: Each failed attempt increments a counter; at 5 failures, set loginBlockedUntil to now + 15 minutes.

- **Lockout Enforcement:** Any login attempt made while now < loginBlockedUntil must be automatically rejected.

- **Success Reset**: On a successful login, the failed attempts counter must be reset to 0 and loginBlockedUntil set to null.

- **Session Cap**: Limit activeSessions to a maximum of 5–10 entries to prevent the document from exceeding size limits.

- **Data Masking:** If hideEmail or hidePhone is true, the API must strip these fields from the response before it reaches the client.

- **MFA Recovery**: backupCodes must be hashed or encrypted and should only be returned to the user once during setup.

OTP Collection

Field Name	Type	Description
userId	ObjectId	Reference to the user this OTP belongs to
otp	string	One-time password code
type	string	OTP type (e.g., "email_verification", "password_reset")
expiresAt	Date	Expiration timestamp of the OTP
createdAt	Date	Timestamp when the OTP was created
isUsed	boolean	Whether the OTP has been used or not
usedAt	Date	Timestamp when the OTP was used (if applicable)
attempts	number	Number of attempts tried with this OTP (optional, for security)
blockedUntil	Date	If attempts exceed 5, user is blocked until this timestamp (e.g., 10 minutes from last attempt)

Notes :

- Each failed attempt increments the attempts count and checks if it has reached 5.

- If attempts reach 5, set blockedUntil = now + 10 minutes.

- Any attempt made before blockedUntil is automatically rejected.

JWT Collection

Field Name	Type	Description
userId	ObjectId	Reference to the user this token belongs to
token	string	Hashed Value Of The JWT
type	string	Token type (e.g., "access", "refresh")
createdAt	Date	Timestamp when the token was issued
expiresAt	Date	Expiration timestamp for the token
revoked	boolean	Whether the token has been revoked before expiration
revokedAt	Date	Timestamp when the token was revoked (if applicable)
deviceInfo	object	Normalized device information (device type, OS, browser)
ipAddress	string	IP address from which the token was issued
userAgent	string	Raw user agent string for session tracking

DeviceInfo Structure:

```
{
  "type": "desktop | mobile | tablet",
  "os": "Linux | Windows | macOS | Android | iOS",
  "browser": "Chrome | Firefox | Safari | Chromium-based"
}
```

Notes :

- All issued access and refresh tokens are stored in the database.
- Each token represents a single authenticated session.
- Token validation requires:
  - Valid signature
  - Not expired
  - Not revoked
  - Existing record in the database
- Tokens are revoked instead of being deleted immediately to allow auditing and security analysis.
- Expired tokens may be cleaned up periodically using a background job.

Company Collection

Field Name	Type	Description
name	string	Company official name
slug	string	URL-friendly unique identifier
email	string	Official company email
phone	string	Company contact phone number
website	string	Company official website
description	string	Company overview and description
industry	string	Company industry or field
size	string	Company size (e.g. 1-10, 11-50, 51-200, 200+)
foundedAt	Date	Company founding date
location	object	Company main location { country, city ,street }
isVerified	boolean	Indicates whether the company is verified
verificationStatus	string	pending | approved | rejected
verifiedAt	Date	Timestamp when company was verified
createdAt	Date	Timestamp when company was created
updatedAt	Date	Timestamp when company was last updated
ownerId	ObjectId	User who owns the company
admins	ObjectId[]	Users who can manage the company
createdBy	ObjectId	User who created the company
logo	object	Company logo { url:string, public_id:string }
CoverImage	object	Company cover image { url:string, public_id:string }
Gallery	object[]	company images
socialLinks	object[]	[{name:string , url:string}]
status	string	active | suspended | banned
suspendedAt	Date	Timestamp when company was suspended
suspendedBy	ObjectId	Admin who suspended the company
bannedAt	Date	Timestamp when company was banned
bannedBy	ObjectId	Admin who banned the company
restoredAt	Date	Timestamp when company was restored
restoredBy	ObjectId	Admin who restored the company

Notes :

- Only verified companies can post job opportunities.

- Company status affects visibility and job posting permissions.

- Soft actions (suspend / ban) are preferred over deletion

Company Settings Collection

Field Name	Type	Description
companyId	ObjectId	Reference to the company this settings document belongs to
allowJobPosting	boolean	Allows or blocks the company from creating new job opportunities
allowApplications	boolean	Enables or disables receiving job applications
autoCloseJobs	boolean	Automatically closes job opportunities after their expiration date
notifyOnNewApplication	boolean	Sends notification when a new application is submitted
notifyOnNewMessage	boolean	Sends notification when a new chat message is received
applicationAutoReply	boolean	Enables automatic reply to applicants after submission
autoReplyMessage	string	Default message sent to applicants (if auto-reply enabled)
visibility	enum	Company visibility level (public / limited / hidden)
createdAt	Date	Timestamp when settings were created
updatedAt	Date	Timestamp when settings were last updated

Notes :

- **Job Posting Control**: If allowJobPosting is false, the "Create Job" button should be disabled and the API must reject new job submissions.

- **Automation Logic:** If autoCloseJobs is true, a daily background cron job should compare currentDate with the job's expiryDate and set the job status to Closed.

- **Auto-Reply Trigger**: When an application is submitted, if applicationAutoReply is true, the system must immediately send the autoReplyMessage to the applicant.

- **Visibility Restrictions**:
  - Public: Profile and jobs are searchable.
  - Limited: Only registered users can see profile details.
  - Hidden: Company profile and jobs are not returned in search results.

- **Notification Filter**: Check notifyOnNewApplication and notifyOnNewMessage before triggering email or push notification services to respect company preferences.

Job Opportunity Collection

Field Name	Type	Description
title	string	Job title
slug	string	URL-friendly unique identifier
description	string	Full job description
requirements	String[]	Required skills and qualifications
responsibilities	string[]	Job responsibilities
employmentType	string	full-time / part-time / internship / freelance / contract
workType	string	on-site / remote / hybrid
experienceLevel	string	junior / mid-level / senior / lead
salary	object	Salary details { min:number, max:number, currency:string, isHidden:boolean }
vacancies	number	Number of open positions
location	object	Job location { country, city ,street }
industry	string	Job industry / field
tags	string[]	Keywords for search and filtering
status	string	draft / published / closed / paused
isUrgent	boolean	Marks urgent hiring
isFeatured	boolean	Highlighted job
publishedAt	Date	Timestamp when job was published
expiresAt	Date	Job expiration date
viewsCount	number	Number of job views
applicationsCount	number	Number of applications
companyId	ObjectId	Reference to the company
createdBy	ObjectId	User (company admin) who created the job
updatedBy	ObjectId	User who last updated the job
createdAt	Date	Creation timestamp
updatedAt	Date	Last update timestamp

Notes :

- Jobs with `status = draft` are visible only to company admins.
- Jobs with `status = published` are public.
- `expiresAt` automatically closes the job when reached
-Closed / expired jobs cannot receive applications.
`-applicationsCount` and `viewsCount` are incremented automatically (no manual edits).
- Salary visibility controlled via `salary.isHidden`.
`-Indexes recommended:`
    - `companyId + status`
    - `tags`
    - `employmentType`
    - `workType`
    - `experienceLevel`

Application Collection

Field Name	Type	Description
userId	ObjectId	Reference to the user applying
jobId	ObjectId	Reference to the job opportunity
companyId	ObjectId	Reference to the company (redundant for quick queries)
status	string	pending / reviewed / accepted / rejected / withdrawn
resume	object	User’s uploaded CV { url: string, public_id: string }
coverLetter	string	Optional cover letter text
appliedAt	Date	Timestamp when the application was submitted
reviewedAt	Date	Timestamp when the application was first reviewed
reviewedBy	ObjectId	Company admin who reviewed the application
notes	string	Internal notes by company admin
interviewDate	Date	Scheduled interview date (optional)
rejectedAt	Date	Timestamp if application rejected
withdrawnAt	Date	Timestamp if user withdraws application
createdAt	Date	Creation timestamp
updatedAt	Date	Last update timestamp

Notes :

- Users can apply only once per job.
- Only active / published jobs accept applications.
- Company admins can change status and add notes.
- status flow: pending → reviewed → accepted/rejected | pending → withdrawn
- Users can withdraw before job closes.
- Shortlisting and interviews are optional but tracked.

- Notifications can be triggered on:

  - Application submission
  - Status change
  - Interview scheduling

Notification Collection

Field Name	Type	Description
userId	ObjectId	Reference to the user receiving the notification
type	string	Notification type (e.g. application_submitted, application_status_changed, job_posted, interview_scheduled, system)
title	string	Short notification title
message	string	Notification message content
data	object	Optional payload (jobId, applicationId, companyId, etc.)
isRead	boolean	Indicates whether the notification was read
readAt	Date	Timestamp when notification was read
isArchived	boolean	Indicates if notification was archived
priority	string	low / normal / high
channel	string	in-app / email / push (future-proof)
createdAt	Date	Creation timestamp
expiresAt	Date	Optional expiration date for notification

Notes :

- Users can mark notifications as read or archived.
- Archived notifications are hidden from main inbox but not deleted.
- High priority notifications can trigger additional channels (email / push).
- Expired notifications are ignored by default.

Common Notification Types :

- `job_posted`
- `application_submitted`
- `application_reviewed`
- `application_accepted`
- `application_rejected`
- `interview_scheduled`
- `company_verified`
- `system`

Chat Collection

Chat functionality is scoped to job applications and is only available between a candidate and a company after an application reaches a review or shortlist state.

Field Name	Type	Description
applicationId	ObjectId	Reference to the related job application
jobId	ObjectId	Reference to the job opportunity
companyId	ObjectId	Reference to the company
candidateId	ObjectId	Reference to the applicant user
participants	ObjectId[]	Chat participants (candidate + company admins)
isActive	boolean	Indicates whether chat is active
closedAt	Date	Timestamp when chat was closed
closedBy	ObjectId	User who closed the chat
lastMessageAt	Date	Timestamp of the last message
createdAt	Date	Creation timestamp
updatedAt	Date	Last update timestamp

Message Collection

Field Name	Type	Description
chatId	ObjectId	Reference to the chat
senderId	ObjectId	User who sent the message
senderRole	string	candidate / company
content	string	Message content
attachments	object[]	Optional files { url, public_id, type }
isRead	boolean	Indicates whether message was read
readAt	Date	Timestamp when message was read
createdAt	Date	Message creation timestamp

Notes :

- Chat is created only after application status becomes reviewed or shortlisted.
- Only participants listed in participants can send messages.
- Chat is automatically closed when:Application is rejected Or Job is closed or expired .
- Closed chats are read-only.
- One chat per application .
- Notifications should be triggered on:
    - New message
    - Chat closed
    -
- Chat Lifecycle : Application reviewed → Chat opened → Messaging → Chat closed

2-Authorization & Access Control

This phase describes the authorization strategy used in the system.
It does not represent database collections, but rather defines how roles and permissions are evaluated during runtime to control access to system resources.

System Roles

System roles define the global access level of a user across the platform.

Role	Description
user	Regular user (job seeker)
admin	Platform administrator
super_admin	Full system access and control

Company Roles (Scoped Roles)

Role	Description
company_owner	Creator of the company with full control
company_admin	Manages jobs, applications, and company data

Core Permissions

Permissions define specific actions that can be performed on system resources.

Permission	Description
create_company	Create a company profile
update_company	Update company information
manage_company_admins	Add or remove company admins
create_job	Create job opportunity
update_job	Update job details
publish_job	Publish job
pause_job	Pause job
close_job	Close job
review_application	Review job applications
update_application_status	Accept or reject applications
access_chat	Access application-related chat
send_message	Send chat messages
verify_company	Verify company profile
suspend_company	Suspend company
ban_company	Ban company

Permission Evaluation Flow

Access to any protected resource is granted only if all the following conditions are met:

- **The user is authenticated.**
- **The user has the required system role or company-scoped role.**
- **The user is associated with the target resource (ownership or membership).**
- **The target entity is in a valid state.**

Example – Create Job Opportunity

- POST /jobs

Conditions:

User role: company_owner or company_admin

- **Company status: active**
- **Company verification: approved**

Entity State Constraints

Authorization decisions may depend on the current state of the entity.

Entity	State Constraint
Company	Must be verified and active
Job	Must be published to accept applications
Application	Must not be rejected or withdrawn
Chat	Must be active to allow messaging

Guard Strategy

Authorization logic is implemented through layered checks:

- **Authentication validation – ensures the user is logged in and token is valid.**
- **Role-based checks – verifies the user's system or company role.**
- **Ownership and association checks – ensures the user is associated with the target resource.**
- **Entity state validation – confirms the entity is in a valid state to perform the action.**

Each request must pass all required checks before reaching business logic.

Design Principles

Authorization logic must be centralized and reusable.

- **Permissions should not be hardcoded inside controllers.**
- **Roles define who the user is.**
- **Permissions define what the user can do.**
- **Access decisions are evaluated at runtime, based on role, ownership, and entity state.**
- **Authorization failures must return meaningful errors (e.g., 403 Forbidden).**

3-API Endpoints Map

This phase provides a complete mapping of all API endpoints required for the Jovio system.
It serves as a reference for backend development, describing each route with its method, authentication requirement, permissions, request data, and purpose.

The goal is to have a clear and organized blueprint, using a module-based structure, so development is consistent, predictable, and aligned with the data model and business rules defined in Phases 1 and

Modules

- **Auth Module**
  Handles all authentication and authorization flows.

- **User Module**
  Handles user profile management, viewing and updating personal information, and retrieving user data (no authentication logic here).

- **Company Module**
  Handles company profiles, verification, admins, ownership, suspension, banning, and CRUD operations on companies.

- **Job Opportunity Module**
  Manages job postings by companies, including job details, status, requirements, and category.

- **Application Module**
  Handles user applications to jobs, tracking status, timestamps, and links to the relevant user and job.

- **Chat Module**
  Manages chat sessions between users and companies related to job applications, including participants and messages.

- **Notification Module**
  Sends and stores notifications for users and companies regarding system events, job applications, or messages.

Auth Module

1- Registration & Verification :

1- Register – System Provider

- **Method / URL: POST /auth/register**
- **Auth: ❌**
- **Permission: ❌**
- **Description: Creates a new user account. Requires { firstName, lastName, email, password, gender, phone, dateOfBirth }. User must activate the account via OTP.**

2- Confirm Email – System Provider

- **Method / URL: POST /auth/confirm-email**
- **Auth: ❌**
- **Permission: ❌**
- **Description: Confirms email via OTP to activate the account for login.**

3- Register / Login – Google Provider

- **Method / URL: POST /auth/google**
- **Auth: ❌**
- **Permission: ❌**
- **Description: Registers/logs in via Google OAuth, automatically confirming the email.**

2- Authentication & Session Management :

1- Login – System Provider

- **Method / URL:** POST /auth/login
- **Auth:** ❌
- **Permission:** ❌
- **Description:** Authenticates credentials; returns JWT tokens and records device/IP info or send 2FA code to user email.

2- Verify 2FA Code

- **Method / URL:** POST /auth/2fa/verify
- **Auth:** ❌ (Pending 2FA state)
- **Permission:** ❌
- **Description:** Validates the 2FA code during the login flow if 2FA is enabled.

2- Refresh Token

- **Method / URL:** POST /auth/refresh
- **Auth:** ✅
- **Permission:** user
- **Description:** Generates a new access token using a valid refresh token.

3- Logout

- **Method / URL:** POST /auth/logout
- **Auth:** ✅
- **Permission:** user
- **Description:** Revokes current JWT or clears cookies.

4- Get User Sessions

- **Method / URL:** GET /auth/sessions
- **Auth:** ✅
- **Permission:** user
- **Description:** Retrieves all active JWT sessions with device and IP metadata.

5- Revoke Session

- **Method / URL:** POST /auth/sessions/revoke
- **Auth:** ✅
- **Permission:** user
- **Description:** Revokes a specific session or all sessions.

1- Security & Recovery :

1- Request Password Reset

- **Method / URL:** POST /auth/request-password-reset
- **Auth:** ❌
- **Permission:** ❌
- **Description:** Sends a password-reset OTP to the registered email.

2- Reset Password

- **Method / URL:** POST /auth/reset-password
- **Auth:** ❌
- **Permission:** ❌
- **Description:** Resets password using valid OTP and new password.

4- Request Email Change

- **Method / URL:** POST /auth/request-email-change
- **Auth:** ✅
- **Permission:** user
- **Description:** Starts the process by sending an OTP to the **new** email.

5- Confirm Email Change

- **Method / URL:** POST /auth/confirm-email-change
- **Auth:** ✅
- **Permission:** user
- **Description:** Updates the user's email after OTP verification.

6- Generate Backup Codes

- **Method / URL:** POST /auth/2fa/backup-codes
- **Auth:** ✅
- **Permission:** user
- **Description:** Generates and returns a new set of encrypted backup codes for account recovery.

User Module

1- Users Retrieval :

1- Get Own Profile

- **Method / URL:** GET /users/me
- **Auth:** ✅
- **Permission:** user
- **Description:** Retrieves the profile of the currently authenticated user.

2- Get User Profile

- **Method / URL:** GET /users/profile/:id
- **Auth:** ✅
- **Permission:** user
- **Description:** Retrieves the profile information of a user by ID.
- **Response:** User object with public and non-sensitive fields.

3- Search Users

- **Method / URL:** GET /users
- **Auth:** ✅
- **Permission:** user
- **Description:** Returns a paginated list of users with filters for email, name, status, and role.

2- Personal Profile Management :

1- Update Own Profile

- **Method / URL:** PATCH /users/profile
- **Auth:** ✅
- **Permission:** user
- **Description:** Updates the authenticated user's profile partially.
- **Fields include:** { firstName?, lastName?, phone?, bio?, socialLinks?, gender?, dateOfBirth? }

2- Upload Profile Picture

- **Method / URL:** POST /users/profile/picture
- **Auth:** ✅
- **Permission:** user
- **Description:** Uploads or updates the user's profile picture.
- **Request:** Accepts an image file.

3- Upload Cover Picture

- **Method / URL:** POST /users/profile/cover
- **Auth:** ✅
- **Permission:** user
- **Description:** Uploads or updates the user’s cover picture.
- **Request:** Accepts an image file.

4- Upload / Replace Resume

- **Method / URL:** PATCH /users/resume/upload
- **Auth:** ✅
- **Permission:** user
- **Description:** Uploads or updates the user’s resume in the `resume` field.
- **Request:** Accepts a PDF file.

5- Delete Resume

- **Method / URL:** DELETE /users/resume/upload
- **Auth:** ✅
- **Permission:** user
- **Description:** Deletes the user’s resume from the profile.

6- Delete Gallery Item

- **Method / URL:** DELETE /users/me/gallery/:public-id
- **Auth:** ✅
- **Permission:** user
- **Description:** Removes a specific image from the user's gallery using its unique ID.

7. Add / Update User Skills

- **Method / URL:** `PATCH /users/me/skills`
- **Auth:** ✅
- **Permission:** `user`
- **Description:** Updates the user's skill set. Since the `User Collection` uses a `bio` or tags for search, this endpoint allows users to submit an array of strings representing their professional expertise (e.g., `["NestJS", "TypeScript", "MongoDB"]`).

8- Get My Skills

- **Method / URL:** `GET /users/me/skills`
- **Auth:** ✅
- **Permission:** `user`
- **Description:** Retrieves the current list of skills associated with the authenticated user's profile to display in the UI management section.

9- Delete Specific Skill

- **Method / URL:** `DELETE /users/me/skills/:skillName`
- **Auth:** ✅
- **Permission:** `user`
- **Description:** Allows a user to remove a specific skill from their profile list.

3- Account Settings & Security :

1- Update User Settings

- **Method / URL:** PATCH /auth/settings
- **Auth:** ✅
- **Permission:** user
- **Description:** Updates preferences such as profileVisibility, hideEmail, hidePhone, and emailNotifications.

2- Update Password (Authenticated)

- **Method / URL:** PATCH /auth/password
- **Auth:** ✅
- **Permission:** user
- **Description:** Allows the logged-in user to change their password by providing the current and new password.

3- Toggle Two-Factor Authentication (2FA)

- **Method / URL:** POST /auth/2fa/toggle
- **Auth:** ✅
- **Permission:** user
- **Description:** Enables or disables 2FA (is2FAEnabled). Requires password verification for security.

5- Freeze Account

- **Method / URL:** DELETE /users/me
- **Auth:** ✅
- **Permission:** user
- **Description:** Soft-deletes the authenticated user account by marking it as frozen/deleted.

Company Module

1- Companies Retrieval :

1- Get Company Profile

- **Method / URL:** GET /companies/:id
- **Auth:** ✅
- **Permission:** public / user / admin
- **Description:** Retrieves company profile information by ID. Only verified companies may have full details exposed publicly.

2- List Companies

- **Method / URL:** GET /companies
- **Auth:** ❌
- **Permission:** public / user / admin
- **Description:** Returns a paginated list of companies with optional filters (name,industry, size, location, status).

3- Search Companies by Job Availability

- **Method / URL:** GET /companies/search/jobs
- **Auth:** ❌
- **Permission:** public / user
- **Description:** Searches companies that currently have active job opportunities. Supports filters like ?jobTitle=, ?category=, ?location=.

2- Company Profile Management :

1- Create Company

- **Method / URL:** POST /companies
- **Auth:** ✅
- **Permission:** user
- **Description:** Creates a new company profile. Requires { name, email, phone, website, description, industry, size, foundedAt, location, logo?, coverImage?, socialLinks? }. Sets `ownerId` to the authenticated user and status to `pending` verification.

2- Update Company

- **Method / URL:** PATCH /companies/:id
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Updates company details partially. Fields include { name?, email?, phone?, website?, description?, industry?, size?, foundedAt?, location?, logo?, coverImage?, socialLinks? }.

4- Freeze Company

- **Method / URL:** DELETE /companies/:id
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Soft-deletes the company (marks as inactive / deleted, does not remove from DB).

5- Upload Company Logo

- **Method / URL:** POST /companies/:id/logo
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Specifically handles the upload or update of the company logo.

6- Upload Company Cover Image

- **Method / URL:** POST /companies/:id/cover
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Specifically handles the upload or update of the company cover image.

7- Add Gallery Image

- **Method / URL:** POST /companies/:id/gallery
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Adds a new image to the company's `gallery` array.

8- Delete Gallery Image

- **Method / URL:** DELETE /companies/:id/gallery/:public-id
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Removes a specific image from the company's gallery.

3- Access & Team Management :

1- Add Company Admin

- **Method / URL:** POST /companies/:id/admins
- **Auth:** ✅
- **Permission:** owner
- **Description:** Adds a user (by email or ID) to the a`dmins` list to allow them to manage jobs and applications for this company.

2- Remove Company Admin

- **Method / URL:** DELETE /companies/:id/admins/:userId
- **Auth:** ✅
- **Permission:** owner
- **Description:** Removes a user from the company's admin list.

4- Company Settings

1-Update Company Settings

- **Method / URL:** PATCH /companies/:id/settings
- **Auth:** ✅
- **Permission:** owner / admin
- **Description:** Updates preferences in the **Company Settings Collection** (e.g., receiveApplicationEmails, autoResponseEnabled, publicContactInfo).

5- System Administration

1- Verify Company

- **Method / URL:** POST /companies/:id/verify
- **Auth:** ✅
- **Permission:** admin / super_admin
- **Description:** Approves or rejects a company verification request. Sets `verificationStatus` and `verifiedAt`.

2- Suspend Company

- **Method / URL:** POST /companies/:id/suspend
- **Auth:** ✅
- **Permission:** admin / super_admin
- **Description:** Suspends a company account, sets `suspendedAt` and `suspendedBy`. Suspended companies cannot post jobs.

3- Restore Company

- **Method / URL:** POST /companies/:id/restore
- **Auth:** ✅
- **Permission:** admin / super_admin
- **Description:** Restores a previously suspended or soft-deleted company. Sets `restoredAt` and `restoredBy`.

4- Ban Company

- **Method / URL:** POST /companies/:id/ban
- **Auth:** ✅
- **Permission:** super_admin
- **Description:** Permanently bans a company account. Sets `bannedAt` and `bannedBy`.

5- Unban Company

- **Method / URL:** POST /companies/:id/unban
- **Auth:** ✅
- **Permission:** super_admin
- **Description:** Unbans a company account. Sets unbannedAt and unbannedBy.

Job Opportunity Module

1- Job Posting & Management

1- Create Job Opportunity

- **Method / URL:** POST /jobs
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Creates a new job posting. Requires { title, description, requirements, location, salary, category, type, deadline }. Automatically links the job to the user's company and sets status to `active`.

2- Update Job Opportunity

- **Method / URL:** PATCH /jobs/:id
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Partially updates job details. Fields include { title?, description?, requirements?, location?, salary?, category?, type?, deadline? }.

3- Update Job Status

- **Method / URL:** PATCH /jobs/:id/status
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Specifically updates the job availability status (draft / published / closed / paused).

4- Delete Job Opportunity

- **Method / URL:** DELETE /jobs/:id
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Soft-deletes a job posting. Marks it as `deleted` in the DB so it no longer appears in searches but remains for historical application data.

2- Discovery & Filtering :

1- Get Job Opportunity

- **Method / URL:** GET /jobs/:id
- **Auth:** ❌
- **Permission:** public / user
- **Description:** Retrieves full details of a specific job, including the company profile and application requirements.

2- List Job Opportunities

- **Method / URL:** GET /jobs
- **Auth:** ❌
- **Permission:** public / user
- **Description:** Returns a paginated list of all active jobs. Supports basic filters: `?companyId=`, `?category=`, `?type=`.

3- Search & Advanced Filter

- **Method / URL:** GET /jobs/search
- **Auth:** ❌
- **Permission:** public / user
- **Description:** Advanced search using query parameters. Supports: `?keyword=` (searches title/description), `?minSalary=`, `?maxSalary=`, `?location=`, and `?remote=true`.

4- Get All Job Categories

- **Method / URL:** GET /jobs/categories
- **Auth:** ❌
- **Permission:** public / user
- **Description:** Returns a list of all unique job categories currently available in the system to help users filter their search.

5- Get Job Matches by User Skills

- **Method / URL: GET /applications/matches/me**
- **Auth:** ✅
- **Permission: user**
- **Description:** Analyzes the user's managed skills and compares them against active job requirements to return the best-fitting opportunities.

4- System Administration :

1- List All Jobs

- **Method / URL:** GET /admin/jobs
- **Auth:** ✅
- **Permission:** system-admin / super_admin
- **Description:** Allows global admins to see all jobs including `closed`, `deleted`, or `flagged` postings across all companies.

2- Flag/Remove Job

- **Method / URL:** PATCH /admin/jobs/:id/moderation
- **Auth:** ✅
- **Permission:** system-admin / super_admin
- **Description:** Allows admins to force-close or hide a job posting if it violates community standards or is identified as spam.

Jop Application Module

4- Submission & Candidate Management :

1- Apply to Job

- **Method / URL:** POST /applications
- **Auth:** ✅
- **Permission:** user
- **Description:** Allows a user to apply for a specific job opportunity. Requires `{ jobId, coverLetter?, resume? }`. The system should automatically pull the user's latest resume from their profile if not provided.

2- Withdraw Application

- **Method / URL:** DELETE /applications/:id
- **Auth:** ✅
- **Permission:** user (owner) / admin
- **Description:** Allows the applicant to withdraw their application before the job closes. Performs a soft-delete by setting `withdrawnAt`.

3- List My Applications

- **Method / URL:** GET /applications/me
- **Auth:** ✅
- **Permission:** user
- **Description:** Returns all applications submitted by the currently authenticated user. Supports pagination and filtering by status.

4- Recruitment & Review Workflow :

1- Get Application Details

- **Method / URL:** GET /applications/:id
- **Auth:** ✅
- **Permission:** user (owner) / company owner / company admin / admin
- **Description:** Retrieves details of a specific application, including status, timestamps, and candidate/job info.

2- Update Application Status

- **Method / URL:** PATCH /applications/:id/status
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Updates the status following the allowed flow: `pending` → `reviewed` → `accepted/rejected`. Sets `reviewedAt` and `reviewedBy` on the first review.

3- Schedule Interview

- **Method / URL:** PATCH /applications/:id/interview
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Sets the `interviewDate` and triggers an `interview_scheduled` notification to the candidate.

4- Add Internal Admin Notes

- **Method / URL:** PATCH /applications/:id/notes
- **Auth:** ✅
- **Permission:** company owner / company admin
- **Description:** Allows recruiters to save internal `notes` about a candidate that are not visible to the applicant.

5- Search & Administration :

1- List Job Applications

- **Method / URL:** GET /applications/job/:jobId
- **Auth:** ✅
- **Permission:** company owner / company admin / admin
- **Description:** Returns all applications for a specific job opportunity. Supports pagination and filtering by status.

2- Search Applications

- **Method / URL:** GET /applications/search
- **Auth:** ✅
- **Permission:** company owner / company admin / admin
- **Description:** Searches applications by user name, job title, status, or date using query parameters.

3- Admin – List All Applications

- **Method / URL:** GET /applications/user/:userId
- **Auth:** ✅
- **Permission:** admin
- **Description:** Allows a system administrator to view all applications submitted by a specific user across different companies.

Chat Module

1- Chat Lifecycle & Management :

1- Create Chat

- **Method / URL:** `POST /chats`
- **Auth:** ✅
- **Permission:** `company owner / company admin`
- **Description:** Creates a new chat session linked to a specific application.
- **Constraint:** Chat can only be initiated after the application status becomes `reviewed` or `shortlisted`.

2- Get Chat Details

- **Method / URL:** `GET /chats/:id`
- **Auth:** ✅
- **Permission:** `participants / admin`
- **Description:** Retrieves metadata for a specific chat, including participant IDs and the linked `applicationId`.

3- Close Chat Session

- **Method / URL:** `POST /chats/:id/close`
- **Auth:** ✅
- **Permission:** `company owner / company admin / admin`
- **Description:** Manually closes a chat session, setting `closedAt` and `closedBy`.
- **System Logic:** Chats are automatically closed by the system if the application is rejected or the job expires. Closed chats become read-only.

2- Messaging :

1- Get Message History

- **Method / URL:** `GET /chats/:id/messages`
- **Auth:** ✅
- **Permission:** `participants / admin`
- **Description:** Retrieves a paginated list of messages for a chat.

2- Mark Messages as Read

- **Method / URL:** `PATCH /chats/:id/read`
- **Auth:** ✅
- **Permission:** `participants`
- **Description:** Updates `isRead` to `true` and sets `readAt` for all messages received by the authenticated user in that chat.

3- Delete Message

- **Method / URL:** `DELETE /chats/:id/messages/:messageId`
- **Auth:** ✅
- **Permission:** `sender / admin`
- **Description:** Performs a soft-delete of a specific message.

3- Search & Lists :

1- **List My Chats (Candidate)**

- **Method / URL:** `GET /chats/me`
- **Auth:** ✅
- **Permission:** `user`
- **Description:** Lists all active and closed chats for the logged-in candidate.

2- **List Company Chats**

- **Method / URL:** `GET /chats/company/:companyId`
- **Auth:** ✅
- **Permission:** `company owner / company admin`
- **Description:** Lists all chats associated with a specific company.

3- Search Chats & Messages

- **Method / URL:** `GET /chats/search`
- **Auth:** ✅
- **Permission:** `participants / admin`
- **Description:** Filters chats or specific message content by keywords, dates, or application status.

Notification Module

1- Search & Lists :

1- List My Notifications (User/Candidate)

- **Method / URL:** `GET /notifications/me`
- **Auth:** ✅
- **Permission:** `user`
- **Description:** Lists all notifications for the authenticated user. Supports pagination, filtering by `isRead` or `isArchived` status, and sorting by `priority`.

2- List Company Notifications

- **Method / URL:** `GET /notifications/company/:companyId`
- **Auth:** ✅
- **Permission:** `company owner / company admin / admin`
- **Description:** Lists all notifications for a specific company. Useful for tracking applications and system alerts.

3- Get Unread Notifications

- **Method / URL:** `GET /notifications/unread`
- **Auth:** ✅
- **Permission:** `user / company owner / company admin`
- **Description:** Returns the notifications where `isRead` is false.

4- List My Archived Notifications

- **Method / URL:** `GET /notifications/archived`
- **Auth:** ✅
- **Permission:** `recipient`
- **Description:** Retrieves all notifications where `isArchived` is set to `true`.
- **Details:** This allows users to view their "Archive" folder, keeping the main inbox clean while retaining historical data for records like `application_status_changed` or `interview_scheduled`.

2- Status Management :

1- Mark Notification as Read

- **Method / URL:** `PATCH /notifications/:id/read`
- **Auth:** ✅
- **Permission:** `recipient`
- **Description:** Marks a specific notification as read by setting `isRead: true` and updating the `readAt` timestamp.

2- Mark All as Read

- **Method / URL:** `PATCH /notifications/read-all`
- **Auth:** ✅
- **Permission:** `user / company owner`
- **Description:** A bulk action to mark all unread notifications for the current user or company as read.

3-Toggle Archive Notification

- **Method / URL:** `PATCH /notifications/:id/toggle-archive`
- **Auth:** ✅
- **Permission:** `recipient`
- **Description:** Sets `isArchived: true or false ,` This hides the notification from the main inbox but keeps it in the database for the user's "Archive" view.

2- Cleanup & System :

Delete Notification

- **Method / URL:** `DELETE /notifications/:id`
- **Auth:** ✅
- **Permission:** `recipient / admin`
- **Description:** Performs a soft-delete of the notification.

Admin – Global Notifications

- **Method / URL:** `POST /notifications/system-broadcast`
- **Auth:** ✅
- **Permission:** `admin / super_admin`
- **Description:** Allows administrators to send a "system" type notification to all users or all companies.