# For Job Portal

## User Authentication & Profiles

1. Signup

URL: accounts/api/signup

Method: POST

Headers: Content-type: application/json

Description: Register a new user (default role = jobseeker, can be changed to employer).

Request Body:

{

"username": "string",

"email": "string",

"password": "string",

"role": "jobseeker | employer"

}

Response:

{

"message": "User registered successfully",

"user\_id": "int",

"role": "string"

}

1. Login

URL: accounts/api/login

Method: POST

Headers: Content-type: application/json

Description: Authenticate user and return JWT/Token.

Request Body:

{

"username\_or\_email": "string",

"password": "string"

}

Response:

{

"token": "jwt\_token\_string",

"user\_id": "int",

"role": "string"

}

1. Logout

URL: accounts/api/logout

Method: POST

Headers: Authorization: Bearer <token>

Description: Logout user and invalidate token.

Request Body: {}

Response:

{

"message": "Logged out successfully"

}

1. Update User Profile

URL: accounts/api/profile/update

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Update user profile (jobseeker or employer).

Request Body:

{

"username": "string",

"email": "string",

"bio": "string",

"skills": ["string"], # for jobseeker

"company\_name": "string" # for employer

}

Response:

{

"message": "Profile updated successfully"

}

1. Change Password

URL: accounts/api/change-password

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Allow user to change password.

Request Body:

{

"old\_password": "string",

"new\_password": "string"

}

Response:

{

"message": "Password changed successfully"

}

1. Delete Account

URL: accounts/api/delete

Method: DELETE

Headers: Authorization: Bearer <token>

Description: Permanently delete a user account.

Request Body: {}

Response:

{

"message": "Account deleted successfully"

}

1. Token Obtain (Login with JWT)

URL: accounts/api/token

Method: POST

Headers: Content-type: application/json

Description: Generate JWT access & refresh tokens for authentication.

Request Body:

{

"username": "string",

"password": "string"

}

Response:

{

"access": "jwt\_access\_token",

"refresh": "jwt\_refresh\_token"

}

1. Token Refresh

URL: accounts/api/token/refresh

Method: POST

Headers: Content-type: application/json

Description: Refresh the JWT access token using the refresh token.

Request Body:

{

"refresh": "jwt\_refresh\_token"

}

Response:

{

"access": "new\_jwt\_access\_token"

}

1. Token Blacklist (Logout for JWT Refresh Tokens)

URL: accounts/api/token/blacklist

Method: POST

Headers: Authorization: Bearer <refresh\_token>, Content-type: application/json

Description: Invalidate (blacklist) a refresh token, usually for logout.

Request Body:

{

"refresh": "jwt\_refresh\_token"

}

Response:

{

"message": "Token blacklisted successfully"

}

* Job Listings & Applications

1. Create Job Listing (Employer only)

URL: jobs/api/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer posts a new job listing.

Request Body:

{

"title": "string",

"description": "string",

"requirements": ["string"],

"location": "string",

"salary\_range": "string",

"employment\_type": "full-time | part-time | internship | contract"

}

Response:

{

"message": "Job created successfully",

"job\_id": "int"

}

1. Get All Job Listings (Public)

URL: jobs/api/list

Method: GET

Headers: Content-type: application/json

Description: Fetch all available job listings (public endpoint).

Request Body: {}

Response:

[

{

"job\_id": "int",

"title": "string",

"company\_name": "string",

"location": "string",

"employment\_type": "string"

}

]

1. Get Job Details

URL: jobs/api/detail/<job\_id>

Method: GET

Headers: Content-type: application/json

Description: Retrieve detailed information about a specific job.

Request Body: {}

Response:

{

"job\_id": "int",

"title": "string",

"description": "string",

"requirements": ["string"],

"location": "string",

"salary\_range": "string",

"employment\_type": "string",

"posted\_by": "employer\_id"

}

1. Update Job Listing (Employer only)

URL: jobs/api/update/<job\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer updates their job listing.

Request Body:

{

"title": "string",

"description": "string",

"requirements": ["string"],

"location": "string",

"salary\_range": "string"

}

Response:

{

"message": "Job updated successfully"

}

1. Delete Job Listing (Employer only)

URL: jobs/api/delete/<job\_id>

Method: DELETE

Headers: Authorization: Bearer <token>

Description: Employer deletes their job listing.

Request Body: {}

Response:

{

"message": "Job deleted successfully"

}

1. Apply for a Job (Jobseeker only)

URL: applications/api/apply/<job\_id>

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Jobseeker applies for a job.

Request Body:

{

"resume\_url": "string",

"cover\_letter": "string"

}

Response:

{

"message": "Application submitted successfully",

"application\_id": "int"

}

1. Get Applications for a Job (Employer only)

URL: applications/api/job/<job\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Employer views all applications for their posted job.

Request Body: {}

Response:

[

{

"application\_id": "int",

"jobseeker\_id": "int",

"resume\_url": "string",

"cover\_letter": "string",

"status": "applied | reviewed | shortlisted | rejected"

}

]

1. Get Jobseeker Applications (Jobseeker only)

URL: applications/api/my-applications

Method: GET

Headers: Authorization: Bearer <token>

Description: Jobseeker views all jobs they have applied to.

Request Body: {}

Response:

[

{

"application\_id": "int",

"job\_id": "int",

"job\_title": "string",

"status": "applied | reviewed | shortlisted | rejected"

}

]

1. Update Application Status (Employer only)

URL: applications/api/update/<application\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer updates application status.

Request Body:

{

"status": "reviewed | shortlisted | rejected"

}

Response:

{

"message": "Application status updated"

}

* Advanced Job Search & Filters

1. Filtered Job Search

URL: jobs/api/search

Method: GET

Headers: Content-type: application/json

Description: Search jobs using filters like location, industry, experience level, salary range, and job type.

Request Body: {}

Query Parameters:

location=string

industry=string

experience=entry | mid | senior

salary\_min=int

salary\_max=int

job\_type=full-time | part-time | remote | internship | contract

Example: /jobs/api/search?location=Delhi&industry=IT&experience=mid&job\_type=remote

Response:

[

{

"job\_id": "int",

"title": "string",

"company\_name": "string",

"location": "string",

"industry": "string",

"experience\_level": "string",

"salary\_range": "string",

"job\_type": "string"

}

]

1. Keyword-based Job Search

URL: jobs/api/search/keyword

Method: GET

Headers: Content-type: application/json

Description: Search jobs based on keywords in title or description.

Request Body: {}

Query Parameters:

q=string

Example: /jobs/api/search/keyword?q=python developer

Response:

[

{

"job\_id": "int",

"title": "string",

"company\_name": "string",

"location": "string",

"salary\_range": "string",

"job\_type": "string"

}

]

* Resume Upload & Parsing

1. Resume Upload

URL: resume/api/upload

Method: POST

Headers: Authorization: Bearer <token>, Content-type: multipart/form-data

Description: Upload a resume (PDF/DOCX) for job applications.

Request Body (Form Data):

{

"resume\_file": <file> # Accepts .pdf, .docx

}

Response:

{

"message": "Resume uploaded successfully",

"resume\_url": "string",

"file\_type": "pdf | docx"

}

1. Get Uploaded Resume

URL: resume/api/get/<user\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch the latest uploaded resume for the user.

Request Body: {}

Response:

{

"resume\_url": "string",

"uploaded\_at": "datetime"

}

1. Delete Resume

URL: resume/api/delete

Method: DELETE

Headers: Authorization: Bearer <token>

Description: Delete the uploaded resume from user profile.

Request Body: {}

Response:

{

"message": "Resume deleted successfully"

}

1. AI-based Resume Parsing (Profile Auto-fill)

URL: resume/api/parse

Method: POST

Headers: Authorization: Bearer <token>, Content-type: multipart/form-data

Description: Upload a resume (PDF/DOCX) and extract structured data using AI for automated profile creation.

Request Body (Form Data):

{

"resume\_file": <file>

}

Response:

{

"message": "Resume parsed successfully",

"extracted\_data": {

"name": "string",

"email": "string",

"phone": "string",

"skills": ["string"],

"education": [

{

"degree": "string",

"institution": "string",

"year": "string"

}

],

"experience": [

{

"job\_title": "string",

"company": "string",

"duration": "string"

}

]

}

}

1. Apply Parsed Data to Profile

URL: resume/api/parse/apply

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Save extracted resume data directly into the user’s profile.

Request Body:

{

"name": "string",

"email": "string",

"phone": "string",

"skills": ["string"],

"education": [...],

"experience": [...]

}

Response:

{

"message": "Profile updated with parsed resume data"

}

* Application Tracking System (ATS)

1. Review Applications (Employer)

URL: ats/api/review/<job\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Employer fetches all applications for a specific job posting.

Request Body: {}

Response:

[

{

"application\_id": "int",

"jobseeker\_id": "int",

"jobseeker\_name": "string",

"resume\_url": "string",

"cover\_letter": "string",

"status": "applied | reviewed | shortlisted | rejected"

}

]

1. Update Application Status (Employer)

URL: ats/api/update-status/<application\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer updates application status (reviewed, shortlisted, rejected).

Request Body:

{

"status": "reviewed | shortlisted | rejected",

"notes": "string"

}

Response:

{

"message": "Application status updated",

"application\_id": "int",

"new\_status": "string"

}

1. **Schedule Interview**

URL: ats/api/schedule-interview/<application\_id>

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer schedules an interview for a shortlisted candidate.

Request Body:

{

"date": "YYYY-MM-DD",

"time": "HH:MM",

"mode": "online | offline",

"location": "string (if offline)",

"meeting\_link": "string (if online)"

}

Response:

{

"message": "Interview scheduled successfully",

"application\_id": "int",

"interview\_details": {

"date": "string",

"time": "string",

"mode": "string",

"location\_or\_link": "string"

}

}

1. Get Scheduled Interviews (Employer & Jobseeker)

URL: ats/api/interviews/my

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch upcoming interviews for the logged-in user (jobseeker sees their interviews, employer sees all they scheduled).

Request Body: {}

Response:

[

{

"interview\_id": "int",

"application\_id": "int",

"candidate\_name": "string",

"job\_title": "string",

"date": "YYYY-MM-DD",

"time": "HH:MM",

"mode": "online | offline",

"location\_or\_link": "string"

}

]

1. Cancel / Reschedule Interview

URL: ats/api/interview/update/<interview\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer updates or cancels a scheduled interview.

Request Body:

{

"action": "reschedule | cancel",

"new\_date": "YYYY-MM-DD (optional if reschedule)",

"new\_time": "HH:MM (optional if reschedule)",

"new\_mode": "string (optional)",

"new\_location\_or\_link": "string (optional)"

}

Response:

{

"message": "Interview rescheduled/canceled successfully",

"interview\_id": "int"

}

* Company Pages & Reviews

1. Create Company Profile (Employer only)

URL: company/api/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer creates a company profile page.

Request Body:

{

"company\_name": "string",

"industry": "string",

"location": "string",

"website": "string",

"description": "string",

"logo\_url": "string"

}

Response:

{

"message": "Company profile created successfully",

"company\_id": "int"

}

1. Update Company Profile (Employer only)

URL: company/api/update/<company\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employer updates their company profile.

Request Body:

{

"industry": "string",

"location": "string",

"website": "string",

"description": "string",

"logo\_url": "string"

}

Response:

{

"message": "Company profile updated successfully"

}

1. Get Company Profile (Public)

URL: company/api/detail/<company\_id>

Method: GET

Headers: Content-type: application/json

Description: Fetch details of a company profile.

Request Body: {}

Response:

{

"company\_id": "int",

"company\_name": "string",

"industry": "string",

"location": "string",

"website": "string",

"description": "string",

"logo\_url": "string",

"average\_rating": "float"

}

1. List All Companies (Public)

URL: company/api/list

Method: GET

Headers: Content-type: application/json

Description: Fetch a list of all companies with basic details.

Request Body: {}

Response:

[

{

"company\_id": "int",

"company\_name": "string",

"industry": "string",

"location": "string",

"average\_rating": "float"

}

]

1. Add Company Review

URL: reviews/api/add/<company\_id>

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employees/jobseekers leave a review & rating for a company.

Request Body:

{

"rating": 1-5,

"review\_text": "string",

"pros": "string",

"cons": "string"

}

Response:

{

"message": "Review added successfully",

"review\_id": "int"

}

1. Get Company Reviews (Public)

URL: reviews/api/company/<company\_id>

Method: GET

Headers: Content-type: application/json

Description: Fetch all reviews for a company.

Request Body: {}

Response:

[

{

"review\_id": "int",

"user\_id": "int",

"rating": 1-5,

"review\_text": "string",

"pros": "string",

"cons": "string",

"created\_at": "datetime"

}

]

1. Update Review (Author only)

URL: reviews/api/update/<review\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: User updates their own review.

Request Body:

{

"rating": 1-5,

"review\_text": "string",

"pros": "string",

"cons": "string"

}

Response:

{

"message": "Review updated successfully"

}

1. Delete Review (Author only)

URL: reviews/api/delete/<review\_id>

Method: DELETE

Headers: Authorization: Bearer <token>

Description: User deletes their own review.

Request Body: {}

Response:

{

"message": "Review deleted successfully"

}

* Notifications & Alerts

1. Set Notification Preferences

URL: notifications/api/preferences/update

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: User sets preferences for email & push notifications.

Request Body:

{

"email\_notifications": true,

"push\_notifications": true,

"job\_alerts": true,

"application\_updates": true,

"interview\_reminders": true

}

Response:

{

"message": "Notification preferences updated successfully"

}

1. Get Notification Preferences

URL: notifications/api/preferences/get

Method: GET

Headers: Authorization: Bearer <token>

Description: Retrieve user’s notification settings.

Request Body: {}

Response:

{

"email\_notifications": true,

"push\_notifications": false,

"job\_alerts": true,

"application\_updates": true,

"interview\_reminders": false

}

1. Get All Notifications (In-App)

URL: notifications/api/list

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch all in-app notifications for a user.

Request Body: {}

Response:

[

{

"notification\_id": "int",

"type": "job\_alert | application\_update | interview",

"title": "string",

"message": "string",

"is\_read": false,

"created\_at": "datetime"

}

]

1. **Mark Notification as Read**

URL: notifications/api/mark-read/<notification\_id>

Method: PUT

Headers: Authorization: Bearer <token>

Description: Mark a single notification as read.

Request Body: {}

Response:

{

"message": "Notification marked as read"

}

1. Delete Notification

URL: notifications/api/delete/<notification\_id>

Method: DELETE

Headers: Authorization: Bearer <token>

Description: Delete a notification from user’s inbox.

Request Body: {}

Response:

{

"message": "Notification deleted successfully"

}

* Payment Gateway for Premium Listings

1. Create Payment for Featured Job

URL: payments/api/job/checkout

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Initiate a payment request for featuring a job posting.

Request Body:

{

"job\_id": "int",

"plan": "basic | standard | premium", # Different highlight levels

"amount": "float",

"currency": "string"

}

Response:

{

"payment\_id": "string",

"job\_id": "int",

"amount": "float",

"currency": "string",

"payment\_url": "string" # Redirect to payment gateway

}

1. Verify Payment for Job Feature

URL: payments/api/job/verify

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Verify the payment transaction after job listing purchase.

Request Body:

{

"payment\_id": "string",

"transaction\_id": "string",

"status": "success | failed"

}

Response:

{

"message": "Payment verified successfully",

"job\_id": "int",

"featured\_until": "datetime"

}

1. Get Featured Job Listings (Public)

URL: jobs/api/featured

Method: GET

Headers: Content-type: application/json

Description: Fetch all featured job listings (highlighted on homepage).

Request Body: {}

Response:

[

{

"job\_id": "int",

"title": "string",

"company\_name": "string",

"featured\_until": "datetime"

}

]

1. Subscribe to Plan

URL: payments/api/subscription/checkout

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Initiate subscription payment for recruiters/companies.

Request Body:

{

"plan": "monthly | quarterly | yearly",

"amount": "float",

"currency": "string"

}

Response:

{

"subscription\_id": "string",

"plan": "string",

"amount": "float",

"payment\_url": "string"

}

1. Verify Subscription Payment

URL: payments/api/subscription/verify

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Verify subscription payment after purchase.

Request Body:

{

"subscription\_id": "string",

"transaction\_id": "string",

"status": "success | failed"

}

Response:

{

"message": "Subscription activated successfully",

"subscription\_id": "string",

"valid\_until": "datetime"

}

1. Get Active Subscription

URL: payments/api/subscription/active

Method: GET

Headers: Authorization: Bearer <token>

Description: Retrieve details of the logged-in company’s active subscription.

Request Body: {}

Response:

{

"subscription\_id": "string",

"plan": "string",

"started\_at": "datetime",

"valid\_until": "datetime",

"status": "active | expired"

}

1. Cancel Subscription

URL: payments/api/subscription/cancel/<subscription\_id>

Method: PUT

Headers: Authorization: Bearer <token>

Description: Cancel an active subscription (no refund after cut-off).

Request Body: {}

Response:

{

"message": "Subscription cancelled successfully",

"subscription\_id": "string"

}

For Professional Community

* Networking & Connections

1. Send Connection Request

URL: community/api/connections/request

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Send a connection request to another user.

Request Body:

{

"receiver\_id": "int"

}

Response:

{

"message": "Connection request sent",

"request\_id": "string",

"status": "pending"

}

1. Accept/Reject Connection Request

URL: community/api/connections/respond/<request\_id>

Method: PUT

Headers: Authorization: Bearer <token>

Description: Accept or reject a pending connection request.

Request Body:

{

"action": "accept | reject"

}

Response:

{

"message": "Connection accepted",

"connection\_id": "string"

}

1. Get My Connections

URL: community/api/connections/my

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch a list of all confirmed connections for the logged-in user.

Request Body: {}

Response:

[

{

"user\_id": "int",

"name": "string",

"headline": "string",

"profile\_picture": "string"

}

]

1. Follow a Professional

URL: community/api/follow/<user\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Follow a professional without sending connection request.

Request Body: {}

Response:

{

"message": "Now following this user",

"user\_id": "int"

}

1. Direct Messaging (1-on-1 Chat)

URL: community/api/messages/send

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Send a private message to a connection.

Request Body:

{

"receiver\_id": "int",

"message": "string"

}

Response:

{

"message\_id": "string",

"status": "sent",

"timestamp": "datetime"

}

1. Fetch Chat History

URL: community/api/messages/history/<user\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Retrieve chat history with a specific user.

Request Body: {}

Response:

[

{

"message\_id": "string",

"sender\_id": "int",

"receiver\_id": "int",

"message": "string",

"timestamp": "datetime"

}

]

1. Create Forum Post

URL: community/api/forums/post

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Create a discussion post in a professional community forum.

Request Body:

{

"title": "string",

"content": "string",

"tags": ["string"]

}

Response:

{

"post\_id": "string",

"message": "Post created successfully"

}

1. Get Forum Posts

URL: community/api/forums/posts

Method: GET

Headers: Content-type: application/json

Description: Fetch latest forum posts (paginated).

Request Body: {}

Response:

[

{

"post\_id": "string",

"title": "string",

"content": "string",

"author": "string",

"tags": ["string"],

"created\_at": "datetime",

"likes": "int",

"comments\_count": "int"

}

]

1. Comment on Forum Post

URL: community/api/forums/comment

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Add a comment to a forum post.

Request Body:

{

"post\_id": "string",

"comment": "string"

}

Response:

{

"comment\_id": "string",

"message": "Comment added successfully"

}

1. Like/Unlike Forum Post

URL: community/api/forums/like/<post\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Like or unlike a forum post.

Request Body: {}

Response:

{

"message": "Post liked",

"likes\_count": "int"

}

* Groups & Events

1. Create a Group

URL: community/api/groups/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Create an industry-specific group.

Request Body:

{

"name": "string",

"description": "string",

"industry": "string",

"privacy": "public | private"

}

Response:

{

"group\_id": "string",

"message": "Group created successfully"

}

1. Join a Group

URL: community/api/groups/join/<group\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Request to join a group (auto-approve for public groups).

Request Body: {}

Response:

{

"message": "Join request sent/approved",

"status": "pending | approved"

}

1. Post in Group

URL: community/api/groups/post

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Post content inside a specific group.

Request Body:

{

"group\_id": "string",

"title": "string",

"content": "string",

"attachments": ["string"]

}

Response:

{

"post\_id": "string",

"message": "Post added to group"

}

1. Get Group Posts

URL: community/api/groups/posts/<group\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch latest posts in a group.

Request Body: {}

Response:

[

{

"post\_id": "string",

"author": "string",

"title": "string",

"content": "string",

"created\_at": "datetime"

}

]

1. Leave Group

URL: community/api/groups/leave/<group\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Leave a group.

Response:

{

"message": "You have left the group"

}

1. Create Event

URL: community/api/events/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Create an event (webinar, meetup, job fair).

Request Body:

{

"title": "string",

"description": "string",

"event\_type": "webinar | meetup | job\_fair",

"location": "string",

"start\_time": "datetime",

"end\_time": "datetime",

"registration\_required": true

}

Response:

{

"event\_id": "string",

"message": "Event created successfully"

}

1. Register for Event

URL: community/api/events/register/<event\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Register for a specific event.

Request Body: {}

Response:

{

"message": "Registered successfully",

"event\_id": "string",

"status": "confirmed"

}

1. Get Upcoming Events

URL: community/api/events/upcoming

Method: GET

Headers: Content-type: application/json

Description: Fetch a list of upcoming events.

Response:

[

{

"event\_id": "string",

"title": "string",

"event\_type": "webinar",

"location": "string",

"start\_time": "datetime",

"end\_time": "datetime",

"registered\_users": "int"

}

]

1. Event Discussions (Chat/Forum for Event)

URL: community/api/events/discussion/<event\_id>

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Post a discussion message in an event’s forum.

Request Body:

{

"message": "string"

}

Response:

{

"discussion\_id": "string",

"message": "Discussion added"

}

1. Event Attendance

URL: community/api/events/attendees/<event\_id>

Method: GET

Headers: Authorization: Bearer <token>

Description: Get list of registered attendees for an event.

Response:

[

{

"user\_id": "int",

"name": "string",

"headline": "string"

}

]

* Skill Endorsements & Recommendations

1. Endorse a Skill

URL: community/api/skills/endorse

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Endorse a skill on another user's profile.

Request Body:

{

"user\_id": "string", // person receiving the endorsement

"skill\_id": "string" // e.g., Python, ReactJS

}

Response:

{

"message": "Skill endorsed successfully",

"endorsement\_id": "string",

"skill\_id": "string",

"endorsed\_by": "string"

}

1. Get Endorsements for a User

URL: community/api/skills/endorsements/<user\_id>

Method: GET

Headers: Content-type: application/json

Description: Get all skill endorsements for a user.

Response:

[

{

"skill\_id": "string",

"skill\_name": "string",

"endorsement\_count": "int",

"endorsed\_by": [

{"user\_id": "string", "name": "string"}

]

}

]

1. Remove an Endorsement

URL: community/api/skills/remove\_endorsement

Method: POST

Headers: Authorization: Bearer <token>

Description: Remove an endorsement you gave.

Request Body:

{

"endorsement\_id": "string"

}

Response:

{

"message": "Endorsement removed successfully"

}

1. Leave Recommendation

URL: community/api/recommendations/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employers leave professional recommendations for users.

Request Body:

{

"user\_id": "string", // candidate being recommended

"employer\_id": "string", // company/HR account

"relationship": "string", // e.g., 'Manager', 'HR Recruiter'

"recommendation\_text": "string"

}

Response:

{

"recommendation\_id": "string",

"message": "Recommendation submitted successfully"

}

1. Get User Recommendations

URL: community/api/recommendations/<user\_id>

Method: GET

Headers: Content-type: application/json

Description: Fetch all recommendations for a given user.

Response:

[

{

"recommendation\_id": "string",

"employer": {

"employer\_id": "string",

"company\_name": "string",

"contact\_name": "string"

},

"relationship": "string",

"recommendation\_text": "string",

"created\_at": "datetime"

}

]

1. Employer Verify/Withdraw Recommendation

URL: community/api/recommendations/manage

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Employers can update or withdraw recommendations.

Request Body:

{

"recommendation\_id": "string",

"action": "update | withdraw",

"updated\_text": "string (optional)"

}

Response:

{

"message": "Recommendation updated/withdrawn successfully"

}

* Content Sharing & Blogs

1. Create a Blog/Article

URL: community/api/blogs/create

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: User creates a blog/article.

Request Body:

{

"title": "string",

"content": "string",

"tags": ["string"],

"visibility": "public | connections\_only"

}

Response:

{

"blog\_id": "string",

"message": "Blog created successfully"

}

1. Get All Blogs (Feed)

URL: community/api/blogs/feed

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch latest blogs/posts for a user feed (based on connections + public posts).

Response:

[

{

"blog\_id": "string",

"title": "string",

"author": {

"user\_id": "string",

"name": "string",

"profile\_pic": "string"

},

"tags": ["string"],

"content\_preview": "string",

"created\_at": "datetime"

}

]

1. Get Single Blog

URL: community/api/blogs/<blog\_id>

Method: GET

Headers: Content-type: application/json

Description: Fetch a full blog/article by ID.

Response:

{

"blog\_id": "string",

"title": "string",

"content": "string",

"tags": ["string"],

"author": {

"user\_id": "string",

"name": "string",

"profile\_pic": "string"

},

"likes": 15,

"comments\_count": 5,

"created\_at": "datetime"

}

1. Update Blog

URL: community/api/blogs/update/<blog\_id>

Method: PUT

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Update an existing blog/article.

Request Body:

{

"title": "string (optional)",

"content": "string (optional)",

"tags": ["string"] (optional)

}

Response:

{

"message": "Blog updated successfully"

}

1. Delete Blog

URL: community/api/blogs/delete/<blog\_id>

Method: DELETE

Headers: Authorization: Bearer <token>

Description: Delete a blog/article.

Response:

{

"message": "Blog deleted successfully"

}

1. Like/Unlike a Blog

URL: community/api/blogs/like

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Like or unlike a blog.

Request Body:

{

"blog\_id": "string",

"action": "like | unlike"

}

Response:

{

"message": "Blog liked/unliked successfully"

}

1. Comment on Blog

URL: community/api/blogs/comment

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Add a comment on a blog/article.

Request Body:

{

"blog\_id": "string",

"comment\_text": "string"

}

Response:

{

"comment\_id": "string",

"message": "Comment added successfully"

}

* Mentorship Program

1. Register as Mentor

URL: community/api/mentorship/register-mentor

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: A professional registers as a mentor.

Request Body:

{

"expertise": ["string"], // e.g., ["Software Engineering", "Data Science"]

"experience\_years": "number",

"available\_slots": "number", // max mentees

"bio": "string"

}

Response:

{

"mentor\_id": "string",

"message": "Mentor profile created successfully"

}

1. Browse Available Mentors

URL: community/api/mentorship/mentors

Method: GET

Headers: Authorization: Bearer <token>

Description: Job seekers browse available mentors.

Query Params (optional):

?expertise=Data Science&location=India

Response:

[

{

"mentor\_id": "string",

"name": "string",

"expertise": ["string"],

"experience\_years": "number",

"bio": "string",

"available\_slots": "number"

}

]

1. Request Mentorship

URL: community/api/mentorship/request

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: A job seeker requests mentorship from a mentor.

Request Body:

{

"mentor\_id": "string",

"mentee\_message": "string" // short intro or request note

}

Response:

{

"request\_id": "string",

"message": "Mentorship request sent"

}

1. Mentor Accept/Reject Request

URL: community/api/mentorship/respond

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Mentor accepts or rejects a mentorship request.

Request Body:

{

"request\_id": "string",

"action": "accept | reject"

}

Response:

{

"message": "Request accepted/rejected successfully"

}

1. Active Mentorship Sessions

URL: community/api/mentorship/sessions

Method: GET

Headers: Authorization: Bearer <token>

Description: Fetch active mentorship sessions (mentor ↔ mentee connections).

Response:

[

{

"session\_id": "string",

"mentor": {

"mentor\_id": "string",

"name": "string"

},

"mentee": {

"mentee\_id": "string",

"name": "string"

},

"status": "active | completed",

"started\_at": "datetime"

}

]

1. End Mentorship Session

URL: community/api/mentorship/end/<session\_id>

Method: POST

Headers: Authorization: Bearer <token>

Description: Mentor or mentee can end the mentorship session.

Response:

{

"message": "Mentorship session ended successfully"

}

1. Rate & Review Mentor

URL: community/api/mentorship/review

Method: POST

Headers: Authorization: Bearer <token>, Content-type: application/json

Description: Mentee rates and reviews mentor after session.

Request Body:

{

"mentor\_id": "string",

"rating": "number (1-5)",

"review": "string"

}

Response:

{

"message": "Review submitted successfully"

}

### Submitted by **Abin Santhosh**