User Management API Documentation

This document provides detailed information about the various endpoints used in the user management system, including required payloads, parameter relationships, and example curl commands.

# 1. User Registration

Endpoint: POST /api/users/register

## Registers a new user in the system. The email domain validation depends on the user's role.

## Payload:

{  
 "firstName": "John",  
 "lastName": "Doe",  
 "email": "johndoe@example.com",  
 "phone": "58798598",  
 "password": "password123",  
 "userRole": "STUDENT"  
}

Keys Explanation:

firstName *(string)*: User's first name. Required.

lastName *(string)*: User's last name. Required.

email *(string)*: User's email. Required.

* Must be unique in the system.
* For COMPANY role, the email domain must be a professional domain (.com, .org, .net, .co, .io, .tech, .in).
* For STUDENT and UNIVERSITY roles, any valid email address is allowed.

phone *(string)*: User's contact number. Required.

password *(string)*: User's password. Required.

* The password will be hashed before saving in the database.

userRole *(enum: "STUDENT", "COMPANY", "UNIVERSITY")*: The role assigned to the user.

* STUDENT: Allows any valid email.
* COMPANY: Requires a professional email address.
* UNIVERSITY: Allows any valid email.

.

## curl Example:

curl --location 'http://localhost:3000/api/users/register' --header 'Content-Type: application/json' --data-raw '{  
 "firstName": "John",  
 "lastName": "Doe",  
 "email": "johndoe@example.com",  
 "phone": "58798598",  
 "password": "password123",  
 "userRole": "STUDENT"  
}'

# User Login

Endpoint: POST /api/users/login

Logs a user into the system and returns a JWT token.

## Payload:

{  
 "email": "johndoe@example.com",  
 "password": "password123"  
}

## curl Example:

curl --location 'http://localhost:3000/api/users/login' --header 'Content-Type: application/json' --data-raw '{  
 "email": "johndoe@example.com",  
 "password": "password123"  
}'

# Get User Basic Info

Endpoint: POST /api/users/getUserData

Fetches user's basic information using their email.

## Payload:

{  
 "email": "johndoe@example.com"  
}

## curl Example:

curl --location 'http://localhost:3000/api/users/getUserData' --header 'Content-Type: application/json' --data-raw '{  
 "email": "johndoe@example.com"  
}'

# Send OTP (Email)

Endpoint: POST /api/otp/send-otp

Sends a 4-digit OTP to the user's email.

## Payload:

{  
 "email": "dikshavashist2007@gmail.com"  
}

## curl Example:

curl --location 'http://localhost:3000/api/otp/send-otp' --header 'Content-Type: application/json' --data-raw '{  
 "email": "dikshavashist2007@gmail.com"  
}'

# Verify OTP (Email)

Endpoint: POST /api/otp/verify-otp

Verifies the OTP sent to the user's email.

## Payload:

{  
 "email": "dikshavashist2007@gmail.com",  
 "otp": "8384"  
}

## curl Example:

curl --location 'http://localhost:3000/api/otp/verify-otp' --header 'Content-Type: application/json' --data-raw '{  
 "email": "dikshavashist2007@gmail.com",  
 "otp": "8384"  
}'

# Send OTP (Mobile)

Endpoint: POST /api/mobileotp/sendotp

Sends a 4-digit OTP to the user's mobile number.

## Payload:

{  
 "phoneNumber": "6353004557"  
}

## curl Example:

curl --location 'http://localhost:3000/api/mobileotp/sendotp' --header 'Content-Type: application/json' --data '{  
 "phoneNumber": "6353004557"  
}'

# Verify OTP (Mobile)

Endpoint: POST /api/mobileotp/verifyotp

Verifies the OTP sent to the user's mobile.

## Payload:

{  
 "phoneNumber": "6353004557",  
 "otp": "4999"  
}

## curl Example:

curl --location 'http://localhost:3000/api/mobileotp/verifyotp' --header 'Content-Type: application/json' --data '{  
 "phoneNumber": "6353004557",  
 "otp": "4999"  
}'

# Submit Detailed User Profile

Endpoint: POST /api/user-details/detail

Submits full profile data for a user.

## Payload:

 userId: References User.id (foreign key).

 email: Should match the email address stored in the User table.

{  
 "userId": "123",  
 "firstName": "john",  
 "lastName": "doe",  
 "email": "johndoe@example.com",  
 "phone": "1234567890",  
 "dob": "1990-01-01",  
 "city": "CityName",  
 "gender": "Male",  
 "userType": "Working Professional",  
 "jobLocation": "San Francisco",  
 "totalExperience": "10 year",  
 "salaryDetails": "100000",  
 "currentlyLookingFor": "job",  
 "workMode": "Remote"  
}

## curl Example:

curl --location 'http://localhost:3000/api/user-details/detail' --header 'Content-Type: application/json' --data-raw '{  
 "userId":"123",  
 "firstName":"john",  
 "lastName":"doe",  
 "email": "johndoe@example.com",  
 "phone":"1234567890",  
 "dob":"1990-01-01",  
 "city":"CityName",  
 "gender":"Male",  
 "userType":"Working Professional",  
 "jobLocation":"San Francisco",  
 "totalExperience":"10 year",  
 "salaryDetails":"100000",  
 "currentlyLookingFor":"job",  
 "workMode":"Remote"  
}'

# Upload Skill Certificate

**Endpoint: POST /upload-skill**

Uploads **multiple skill certificates** for a single user.

**Payload (multipart/form-data):**

* user\_id (Integer): References User.id
* skills (JSON string): An array of skill objects, each containing:
  + skill\_id (Integer): References Skill.id
  + skill (String): Name of the skill
  + authority (String): Authority/issuer of the certificate
* certificate\_image\_<index> (File): Certificate image for each skill, where <index> corresponds to the index in the skills array.

**cURL Example:**

bash

CopyEdit

curl --location 'http://localhost:3000/upload-skill' \

--form 'user\_id="1"' \

--form 'skills="[{

\"skill\_id\": 101,

\"skill\": \"JavaScript\",

\"authority\": \"Udemy\"

}, {

\"skill\_id\": 102,

\"skill\": \"Node.js\",

\"authority\": \"Coursera\"

}]"' \

--form 'certificate\_image\_0=@"/C:/Users/DIKSHA/Desktop/WhatsApp Image 2024-12-30 at 16.18.03\_25b8d73c.jpg"' \

--form 'certificate\_image\_1=@"/C:/Users/DIKSHA/Desktop/WhatsApp Image 2025-05-07 at 11.20.55\_19246a3f.jpg"'

**API: Create Job Post**

**URL**

bash

CopyEdit

POST /api/jobpost/create

**Headers**

| **Key** | **Value** |
| --- | --- |
| Content-Type | application/json |
| Authorization | Bearer <JWT token> |

**Authorization**

JWT is required. The token should contain a valid userId, typically assigned during login. This links the job post to the user.

**Relationships (DB)**

* JobPost.userId → references User.id from the JWT token (decoded server-side).
* You **do not** send userId manually — it’s inferred from the token.

**Example cURL**

bash

CopyEdit

curl --location 'http://localhost:3000/api/jobpost/create' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer <your-token-here>' \

--data '{

"opportunityType": "Internship",

"jobProfile": "Software Developer",

"skillsRequired": "JavaScript, Node.js",

...

}'

# 📘 Complete API Flow Based on UI + Features

## 🔹 Common Flow (All Users)

* • Register: `POST /api/users/register`
* • Login: `POST /api/users/login`
* • Verify Email OTP: `POST /api/otp/verify-otp`
* • Verify Mobile OTP: `POST /api/mobileotp/verifyotp`
* • Submit Full User Profile: `POST /api/user-details/detail`
* • Upload Skill Certificates: `POST /upload-skill`

## 👨‍🎓 Student / Fresher / Working Professional Flow

* • Browse Opportunities: `GET /api/opportunities`
* • Apply to a Job/Internship: `POST /api/apply`
* • Check Aadhar Status: `GET /api/user/aadhar-status`
* • Verify Aadhar: `POST /api/user/verify-aadhar`
* • View Dashboard: `GET /api/user/dashboard`
* • View Profile: `GET /api/users/getUserData`

## 🏢 Company Flow

* • Submit Company Details: `PUT /api/company/details`
* • Create Job Post: `POST /api/jobpost/create`
* • View My Job Posts: `GET /api/opportunities/my`
* • View Company Dashboard: `GET /api/company/dashboard`
* • View Applications per Job: `GET /api/applications?jobId=...`
* • View Application Details: `GET /api/applications/:id`
* • Update Application Status: `PUT /api/applications/:id/status`
* • Schedule Interview: `POST /api/interviews/schedule`
* • List Upcoming Interviews: `GET /api/interviews/upcoming`
* • View Candidate Pipeline: `GET /api/pipeline?jobId=...`
* • Task Management: `GET /api/tasks, POST /api/tasks/create, PUT /api/tasks/:id/update`
* • Approval Management: `GET /api/approvals/pending, POST /api/approvals/:id/approve`

# 🧩 UI-Based Stepwise API Flow Breakdown (from Full Screenshot)

## 1. User Registration & Login

* • Endpoint: `POST /api/users/register`

Payload Example:

* - firstName: string
* - lastName: string
* - email: string
* - phone: string
* - password: string
* - userRole: STUDENT | COMPANY | UNIVERSITY
* • Endpoint: `POST /api/users/login`

Payload Example:

* - email: string
* - password: string
* • Endpoint: `POST /api/otp/verify-otp`

Payload Example:

* - email: string
* - otp: string

## 2. Profile Completion (Students/Professionals)

* • Endpoint: `POST /api/user-details/detail`

Payload Example:

* - userId: string
* - email: string
* - dob: YYYY-MM-DD
* - gender: Male/Female/Other
* - city: string
* - userType: Working Professional | Fresher | Student
* - jobLocation: string
* - totalExperience: string
* - salaryDetails: string
* - currentlyLookingFor: job/internship
* - workMode: Remote/On-site/Hybrid

## 3. Upload Skill Certificates

* • Endpoint: `POST /upload-skill`

Payload Example:

* - user\_id: int (form field)
* - skills: [{ skill\_id, skill, authority }]
* - certificate\_image\_X: File (multipart/form-data)

## 4. Aadhar Authentication

* • Endpoint: `POST /api/user/verify-aadhar`

Payload Example:

* - userId: string
* - aadharNumber: string
* - dob: YYYY-MM-DD
* - name: string

## 5. Company Job Posting Flow

* • Endpoint: `POST /api/jobpost/create`

Payload Example:

* - opportunityType: Internship/Job
* - jobProfile: string
* - skillsRequired: comma-separated string
* - type: Remote/On-site/Hybrid
* - duration: e.g. 3 months
* - stipend: string
* - perks: string
* - screeningQuestions: [string]

## 6. Job Listing & Application

* • Endpoint: `GET /api/opportunities`

Payload Example:

* • Endpoint: `POST /api/apply`

Payload Example:

* - jobId: string
* - userId: string
* - screeningAnswers: [{ question, answer }]

## 7. Messaging System

* • Endpoint: `GET /api/chats`

Payload Example:

* • Endpoint: `GET /api/chats/:id/messages`

Payload Example:

* • Endpoint: `POST /api/chats/:id/messages`

Payload Example:

* - message: string

## 8. Dashboards (Company & Student)

* • Endpoint: `GET /api/company/dashboard`

Payload Example:

* • Endpoint: `GET /api/user/dashboard`

Payload Example:

## 9. Application Management (Company)

* • Endpoint: `GET /api/applications?jobId=...`

Payload Example:

* • Endpoint: `PUT /api/applications/:id/status`

Payload Example:

* - status: shortlisted/rejected/selected

## 10. Interview Scheduling & Approvals

* • Endpoint: `POST /api/interviews/schedule`

Payload Example:

* - jobId: string
* - applicantId: string
* - dateTime: ISO8601
* - mode: online/offline
* • Endpoint: `GET /api/interviews/upcoming`

Payload Example:

* • Endpoint: `GET /api/approvals/pending`

Payload Example:

* • Endpoint: `POST /api/approvals/:id/approve`

Payload Example: