User  
  
**1. 회원가입**

**URL 및 메서드**

**POST** /api/users/register

**Description**

새로운 사용자를 등록합니다.

**Request Body**

{

"email": "user@example.com",

"password": "password123",

"name": "홍길동",

"birthDate": "950101", // 생년월일 (선택, 개인 회원일 경우 필수)

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890", // 사업자 번호 (선택, 기업 회원일 경우 필수)

"role": "PERSONAL" // PERSONAL, COMPANY, ADMIN 중 하나

}

**Response Body**

{

"result": true,

"message": "회원가입이 완료되었습니다.",

"data": {

"id": 1,

"email": "user@example.com",

"name": "홍길동",

"birthDate": "950101",

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890",

"image": null,

"role": "PERSONAL"

}

}

**Error Response**

{

"result": false,

"message": "이미 존재하는 이메일입니다."

}

**2. 로그인**

**URL 및 메서드**

**POST** /api/users/login

**Description**

이메일과 비밀번호를 이용해 로그인합니다.

**Request Body**

{

"email": "user@example.com",

"password": "password123",

"role": "PERSONAL"

}

**Response Body**

{

"result": true,

"message": "로그인 성공",

"data": {

"token": "eyJhbGciOiJIUz...",

"exprTime": "3600",

"name": "홍길동"

}

}

**Error Response**

{

"result": false,

"message": "이메일 또는 비밀번호가 일치하지 않습니다."

}

**3. 모든 사용자 목록 조회 (관리자 전용)**

**URL 및 메서드**

**GET** /api/users

**Description**

모든 사용자 정보를 조회합니다. 관리자만 접근 가능합니다.

**Response Body**

{

"result": true,

"message": "회원 목록 조회 성공",

"data": [

{

"id": 1,

"email": "user@example.com",

"name": "홍길동",

"birthDate": "950101",

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890",

"image": null,

"role": "PERSONAL"

}

]

}

**Error Response**

{

"result": false,

"message": "접근 권한이 없습니다."

}

**4. 이메일로 사용자 조회**

**URL 및 메서드**

**GET** /api/users/{email}

**Description**

특정 이메일의 사용자 정보를 조회합니다.

**Response Body**

{

"result": true,

"message": "사용자 조회 성공",

"data": {

"id": 1,

"email": "user@example.com",

"name": "홍길동",

"birthDate": "950101",

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890",

"image": null,

"role": "PERSONAL"

}

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**5. 이메일로 사용자 정보 업데이트**

**URL 및 메서드**

**PUT** /api/users/{email}

**Description**

특정 이메일의 사용자 정보를 수정합니다.

**Request Body**

{

"name": "홍길동",

"birthDate": "950101",

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890"

}

**Response Body**

{

"result": true,

"message": "사용자 정보 업데이트 성공",

"data": {

"id": 1,

"email": "user@example.com",

"name": "홍길동",

"birthDate": "950101",

"gender": "남성",

"phone": "01012345678",

"businessNumber": "1234567890",

"image": null,

"role": "PERSONAL"

}

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**6. 이메일로 사용자 삭제**

**URL 및 메서드**

**DELETE** /api/users/{email}

**Description**

특정 이메일의 사용자를 삭제합니다.

**Response Body**

{

"result": true,

"message": "사용자 삭제 완료",

"data": null

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**7. 이메일 찾기**

**URL 및 메서드**

**GET** /api/users/find-id

**Description**

사용자 이름과 전화번호 등 정보를 이용해 이메일을 조회합니다.

**요청 파라미터**

* name: 사용자 이름
* phone: 전화번호
* role: PERSONAL 또는 COMPANY
* businessNumber (선택): 사업자 번호 (기업 회원일 경우)

**Response Body**

{

"result": true,

"message": "이메일 찾기 성공",

"data": "user@example.com"

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**8. 비밀번호 재설정**

**URL 및 메서드**

**POST** /api/users/reset-password

**Description**

새로운 비밀번호를 설정합니다.

**Request Body**

{

"email": "user@example.com",

"newPassword": "newPassword123"

}

**Response Body**

{

"result": true,

"message": "비밀번호 재설정 성공",

"data": null

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**9. 사용자 이미지 업로드**

**URL 및 메서드**

**PUT** /api/users/{email}/image

**Description**

사용자의 프로필 이미지를 업로드합니다.

**요청 데이터**

Form-Data:

* image (File)

**Response Body**

{

"result": true,

"message": "이미지 업로드 성공",

"data": "https://example.com/uploads/profile.jpg"

}

**Error Response**

{

"result": false,

"message": "이미지를 업로드할 수 없습니다."

}

Time table  
  
**1. 시간표 등록**

**URL 및 메서드**  
POST /api/timetable/{email}

**Description**  
특정 이메일의 사용자가 시간표를 등록합니다.

**Request Body**

{

"email": "user@example.com",

"registered": true,

"monday": "8, 9, 10",

"tuesday": "11, 12, 20, 21, 22",

"wednesday": null,

"thursday": "8, 9, 10, 11",

"friday": "12, 13",

"saturday": "14, 15",

"sunday": null

}

**Response Body**

{

"result": true,

"message": "시간표가 등록되었습니다.",

"data": {

"email": "user@example.com",

"registered": true,

"monday": "8, 9, 10",

"tuesday": "11, 12, 20, 21, 22",

"wednesday": null,

"thursday": "8, 9, 10, 11",

"friday": "12, 13",

"saturday": "14, 15",

"sunday": null

}

}

**Error Response**

{

"result": false,

"message": "시간표 등록 중 문제가 발생했습니다."

}

**2. 시간표 조회**

**URL 및 메서드**  
GET /api/timetable/{email}

**Description**  
특정 이메일의 사용자 시간표를 조회합니다.

**Response Body**

{

"result": true,

"message": "시간표 조회 성공.",

"data": {

"email": "user@example.com",

"registered": true,

"monday": "8, 9, 10",

"tuesday": "11, 12, 20, 21, 22",

"wednesday": null,

"thursday": "8, 9, 10, 11",

"friday": "12, 13",

"saturday": "14, 15",

"sunday": null

}

}

**Error Response**

{

"result": false,

"message": "시간표를 찾을 수 없습니다."

}

**3. 시간표 수정**

**URL 및 메서드**  
PUT /api/timetable/{email}

**Description**  
특정 이메일의 사용자가 기존의 시간표를 수정합니다.

**Request Body**

{

"email": "user@example.com",

"registered": true,

"monday": "10, 11, 12",

"tuesday": "13, 14",

"wednesday": null,

"thursday": "9, 10",

"friday": "12, 13, 14",

"saturday": null,

"sunday": "15, 16"

}

**Response Body**

{

"result": true,

"message": "시간표 수정 성공.",

"data": {

"email": "user@example.com",

"registered": true,

"monday": "10, 11, 12",

"tuesday": "13, 14",

"wednesday": null,

"thursday": "9, 10",

"friday": "12, 13, 14",

"saturday": null,

"sunday": "15, 16"

}

}

**Error Response**

{

"result": false,

"message": "시간표 수정 중 문제가 발생했습니다."

}

Resume

**1. 이력서 등록**

**URL 및 메서드**  
POST /api/resumes

**Description**  
새로운 이력서를 등록합니다.

**Request Body**

{

"resumeTitle": "Software Developer Resume",

"selfIntroduction": "Passionate about technology and innovation.",

"educationLevel": "Bachelor's Degree in Computer Science",

"career": "5 years at XYZ Tech, developing backend solutions",

"preferredWorkLocation": ["Seoul", "Incheon"],

"preferredJobTypes": ["IT", "Software Development"],

"employmentTypes": ["Full-time", "Remote"],

"workPeriod": "6 months",

"workDays": "Monday to Friday"

}

**Query Parameter**

* email (string): 등록하려는 사용자의 이메일

**Response Body**

{

"result": true,

"message": "이력서가 등록되었습니다.",

"data": {

"id": 1,

"resumeTitle": "Software Developer Resume",

"selfIntroduction": "Passionate about technology and innovation.",

"educationLevel": "Bachelor's Degree in Computer Science",

"career": "5 years at XYZ Tech, developing backend solutions",

"preferredWorkLocation": ["Seoul", "Incheon"],

"preferredJobTypes": ["IT", "Software Development"],

"employmentTypes": ["Full-time", "Remote"],

"workPeriod": "6 months",

"workDays": "Monday to Friday"

}

}

**Error Response**

{

"result": false,

"message": "사용자를 찾을 수 없습니다."

}

**2. 이메일 기준 이력서 조회**

**URL 및 메서드**  
GET /api/resumes

**Description**  
특정 이메일로 등록된 이력서를 조회합니다. 이메일이 없는 경우 전체 이력서를 조회할 수 있습니다.

**Query Parameter**

* email (string, optional): 조회하려는 사용자의 이메일

**Response Body**

{

"result": true,

"message": "이력서 목록 조회 성공",

"data": [

{

"id": 1,

"resumeTitle": "Software Developer Resume",

"selfIntroduction": "Passionate about technology and innovation.",

"educationLevel": "Bachelor's Degree in Computer Science",

"career": "5 years at XYZ Tech, developing backend solutions",

"preferredWorkLocation": ["Seoul", "Incheon"],

"preferredJobTypes": ["IT", "Software Development"],

"employmentTypes": ["Full-time", "Remote"],

"workPeriod": "6 months",

"workDays": "Monday to Friday"

},

{

"id": 2,

"resumeTitle": "Data Analyst Resume",

"selfIntroduction": "Specialized in big data and analytics.",

"educationLevel": "Master's Degree in Statistics",

"career": "3 years at ABC Analytics",

"preferredWorkLocation": ["Busan"],

"preferredJobTypes": ["Data Analysis"],

"employmentTypes": ["Full-time"],

"workPeriod": "1 year",

"workDays": "Monday to Friday"

}

]

}

**Error Response**

{

"result": false,

"message": "등록된 이력서가 없습니다."

}

**3. 이력서 ID로 조회**

**URL 및 메서드**  
GET /api/resumes/{id}

**Description**  
특정 ID로 이력서를 조회합니다.

**Response Body**

{

"result": true,

"message": "이력서 조회 성공",

"data": {

"id": 1,

"resumeTitle": "Software Developer Resume",

"selfIntroduction": "Passionate about technology and innovation.",

"educationLevel": "Bachelor's Degree in Computer Science",

"career": "5 years at XYZ Tech, developing backend solutions",

"preferredWorkLocation": ["Seoul", "Incheon"],

"preferredJobTypes": ["IT", "Software Development"],

"employmentTypes": ["Full-time", "Remote"],

"workPeriod": "6 months",

"workDays": "Monday to Friday"

}

}

**Error Response**

{

"result": false,

"message": "이력서를 찾을 수 없습니다."

}

**4. 이력서 수정**

**URL 및 메서드**  
PUT /api/resumes/{id}

**Description**  
특정 ID의 이력서를 수정합니다.

**Request Body**

{

"resumeTitle": "Updated Resume Title",

"selfIntroduction": "Updated introduction.",

"educationLevel": "Updated education level",

"career": "Updated career details",

"preferredWorkLocation": ["Updated Location"],

"preferredJobTypes": ["Updated Job Type"],

"employmentTypes": ["Updated Employment Type"],

"workPeriod": "Updated Period",

"workDays": "Updated Days"

}

**Query Parameter**

* email (string): 수정하려는 사용자의 이메일

**Response Body**

{

"result": true,

"message": "이력서 수정 완료",

"data": {

"id": 1,

"resumeTitle": "Updated Resume Title",

"selfIntroduction": "Updated introduction.",

"educationLevel": "Updated education level",

"career": "Updated career details",

"preferredWorkLocation": ["Updated Location"],

"preferredJobTypes": ["Updated Job Type"],

"employmentTypes": ["Updated Employment Type"],

"workPeriod": "Updated Period",

"workDays": "Updated Days"

}

}

**Error Response**

{

"result": false,

"message": "이력서를 찾을 수 없습니다."

}

**5. 이력서 삭제**

**URL 및 메서드**  
DELETE /api/resumes/{id}

**Description**  
특정 ID의 이력서를 삭제합니다.

**Query Parameter**

* email (string): 삭제하려는 사용자의 이메일

**Response Body**

{

"result": true,

"message": "이력서 삭제 완료",

"data": null

}

**Error Response**

{

"result": false,

"message": "이력서를 찾을 수 없습니다."

}

Job Post

**1. Create Job Post**

**URL 및 메서드:**  
POST /api/job-posts

**Description:**  
새로운 구인 공고를 생성합니다.

**Request Body (Multipart):**

{

"jobPost": {

"title": "Software Developer",

"companyName": "Tech Corp",

"companyContent": "We are hiring developers for exciting projects.",

"place": "Seoul",

"placeDetail": "Tech Tower, 5th Floor",

"workCategory": ["IT", "Development"],

"workType": ["Full-time", "Contract"],

"peopleNum": 5,

"career": "2+ years",

"workTerm": "6 months",

"workDays": ["Monday", "Tuesday"],

"workTime": "09:00~18:00",

"pay": "5000 USD",

"gender": "Any",

"age": "20-30",

"deadline": "2024-12-31",

"submitMethod": ["Email", "Website"]

},

"companyImage": "File",

"email": "hr@techcorp.com"

}

**Response Body:**

{

"result": true,

"message": "구인 공고가 등록되었습니다.",

"data": {

"id": 1,

"title": "Software Developer",

"companyName": "Tech Corp",

"place": "Seoul",

"workTime": "09:00~18:00",

"pay": "5000 USD",

"createdAt": "2024-11-26T12:00:00Z"

}

}

**Error Response:**

{

"result": false,

"message": "구인 공고 등록 실패"

}

**2. Get Job Posts (Filtered by Email)**

**URL 및 메서드:**  
GET /api/job-posts

**Description:**  
특정 이메일에 해당하는 사용자의 구인 공고를 조회합니다.

**Query Parameters:**

* email (string, optional): 조회할 사용자의 이메일

**Response Body:**

{

"result": true,

"message": "구인 공고 목록 조회 성공",

"data": [

{

"id": 1,

"title": "Software Developer",

"companyName": "Tech Corp",

"place": "Seoul",

"workTime": "09:00~18:00",

"pay": "5000 USD",

"createdAt": "2024-11-26T12:00:00Z"

}

]

}

**Error Response:**

{

"result": false,

"message": "구인 공고를 찾을 수 없습니다."

}

**3. Get Single Job Post by ID**

**URL 및 메서드:**  
GET /api/job-posts/{id}

**Description:**  
특정 ID에 해당하는 단일 구인 공고를 조회합니다.

**Path Parameters:**

* id (integer, required): 구인 공고의 ID

**Response Body:**

{

"result": true,

"message": "구인 공고 조회 성공",

"data": {

"id": 1,

"title": "Software Developer",

"companyName": "Tech Corp",

"place": "Seoul",

"workTime": "09:00~18:00",

"pay": "5000 USD",

"createdAt": "2024-11-26T12:00:00Z"

}

}

**Error Response:**

{

"result": false,

"message": "구인 공고를 찾을 수 없습니다."

}

**4. Update Job Post**

**URL 및 메서드:**  
PUT /api/job-posts/{id}

**Description:**  
특정 ID의 구인 공고 정보를 업데이트합니다.

**Request Body (Multipart):**

{

"updatedJobPost": {

"title": "Updated Developer Position",

"workTime": "10:00~19:00"

},

"updatedCompanyImage": "File",

"email": "hr@techcorp.com"

}

**Response Body:**

{

"result": true,

"message": "구인 공고가 수정되었습니다.",

"data": {

"id": 1,

"title": "Updated Developer Position",

"workTime": "10:00~19:00"

}

}

**Error Response:**

{

"result": false,

"message": "구인 공고를 수정할 수 없습니다."

}

**5. Delete Job Post**

**URL 및 메서드:**  
DELETE /api/job-posts/{id}

**Description:**  
특정 ID에 해당하는 구인 공고를 삭제합니다.

**Query Parameters:**

* email (string, required): 공고 소유자의 이메일

**Response Body:**

{

"result": true,

"message": "구인 공고가 삭제되었습니다.",

"data": null

}

**Error Response:**

{

"result": false,

"message": "구인 공고를 삭제할 수 없습니다."

}

**6. Get Sorted Job Posts**

**URL 및 메서드:**  
GET /api/job-posts/sorted

**Description:**  
정렬된 구인 공고를 조회합니다.

**Query Parameters:**

* sort (string, required): latest 또는 popular

**Response Body:**

{

"result": true,

"message": "정렬된 구인 공고 조회 성공",

"data": [

{

"id": 1,

"title": "Software Developer",

"pay": "5000 USD",

"createdAt": "2024-11-26T12:00:00Z"

}

]

}

**Error Response:**

{

"result": false,

"message": "잘못된 정렬 요청입니다."

}

**7. Search Job Posts**

**URL 및 메서드:**  
GET /api/job-posts/search

**Description:**  
키워드를 기반으로 구인 공고를 검색합니다.

**Query Parameters:**

* keyword (string, required): 검색어

**Response Body:**

{

"result": true,

"message": "검색 결과 조회 성공",

"data": []

}

**Error Response:**

{

"result": false,

"message": "검색어를 입력해야 합니다."

}

**8. Filter Job Posts**

**URL 및 메서드:**  
POST /api/job-posts/filter

**Description:**  
필터를 적용하여 구인 공고를 조회합니다.

**Request Body:**

{

"place": "Seoul",

"workCategory": ["IT"],

"payRange": "3000~6000 USD"

}

**Response Body:**

{

"result": true,

"message": "필터링된 구인 공고 조회 성공",

"data": []

}

**Error Response:**

{

"result": false,

"message": "필터링에 실패했습니다."

}

JobApplication

**1. 알바 지원하기**

**URL 및 메서드**  
**POST** /api/job-applications/apply/{jobPostId}

**Description**  
특정 구인 공고에 알바를 지원합니다. 사용자는 자신의 이력서 중 하나를 선택하고, 간단한 자기소개를 작성합니다.

**Request Body**

{

"resumeId": 123, // 지원자의 이력서 ID

"description": "열심히 하겠습니다!" // 자기소개

}

**Response Body**

{

"result": true,

"message": "알바 지원이 완료되었습니다.",

"data": null

}

**Error Response**

{

"result": false,

"message": "구인 공고를 찾을 수 없습니다."

}

**2. 특정 공고의 지원 내역 조회**

**URL 및 메서드**  
**GET** /api/job-applications/applications/{jobPostId}

**Description**  
특정 공고에 지원한 모든 지원자 정보를 조회합니다.

**Response Body**

{

"result": true,

"message": "신청 목록 조회 성공",

"data": [

{

"id": 8,

"resume": 123,

"description": "열심히 하겠습니다!"

},

{

"id": 9,

"resume": 124,

"description": "성실히 임하겠습니다."

}

],

"applicantCount": 2 // 총 지원자 수

}

**Error Response**

{

"result": false,

"message": "해당 공고에 대한 신청이 없습니다."

}

**3. 내가 지원한 공고 목록 조회**

**URL 및 메서드**  
**GET** /api/job-applications/applied-jobs

**Description**  
로그인한 사용자가 지원한 공고 목록을 조회합니다.

**Query Parameters**

* **email** (string): 조회할 사용자의 이메일

**Response Body**

{

"result": true,

"message": "지원한 공고 목록 조회 성공",

"data": [

{

"id": 1,

"title": "카페 알바 모집",

"place": "서울 강남구",

"companyName": "스타벅스",

"deadline": "2024-12-31",

"createdAt": "2024-11-01"

},

{

"id": 2,

"title": "편의점 야간 근무",

"place": "서울 마포구",

"companyName": "CU 편의점",

"deadline": "2024-12-20",

"createdAt": "2024-11-10"

}

]

}

**Error Response**

{

"result": false,

"message": "해당 이메일로 지원한 공고가 없습니다."

}