

API 명세서 - 2. 회원 API

전체 API 목록 [↗](#)

API	분류	Method	API 명	설명
회원	회원가입	POST	/api/user/sign-up	계정 생성
	이메일, 닉네임 중복 확인	GET	/api/user/check	계정, 닉네임 중복 확인
	이메일 인증	POST	/api/user/check/auth	인증 코드 메일로 전송
		GET	/api/user/check/auth	메일 인증
	회원탈퇴	POST	/api/user/withdrawal	계정 삭제
	로그인	POST	/api/user/login	로그인 후 토큰 획득
	로그아웃	GET	/api/user/logout	로그아웃, 토큰 해제
	계정 조회	GET	/api/user/all	모든 계정 정보 조회
		GET	/api/user/{id}	본인 계정 정보 조회
	계정 정보 수정	PUT	/api/user/{id}	계정 정보 수정
	계정 권한 수정	PUT	/api/user/role	계정 권한 수정

회원 가입 API [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X POST 'http://localhost/api/user/sign-up'
```

API 호출 방식 [↗](#)

메서드	요청 url
POST	/api/user/sign-up

Request Body [↗](#)

```
1 {
2   "email": "zhyun@gmail.com",
3   "password": "secret!!",
```

```
4   "nickname": "얼거스"
5 }
```

파라미터	타입	필수 여부	설명
email	String	필수	아이디로 사용 될 이메일주소
password	String	필수	비밀번호
nickname	String	필수	닉네임

Response [↗](#)

회원가입 성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "얼거스님 가입을 축하합니다! 🎉"
4 }
```

실패 응답 예시 [↗](#)

- CASE 1 : 이메일 필드에 값이 없음 [↗](#)

```
1 {
2   "status": false,
3   "message": "이메일을 입력해 주세요."
4 }
```

- CASE 2 : 비밀번호 필드에 값이 없음 [↗](#)

```
1 {
2   "status": false,
3   "message": "비밀번호를 입력해 주세요."
4 }
```

- CASE 3 : 닉네임 필드에 값이 없음 [↗](#)

```
1 {
2   "status": false,
3   "message": "닉네임을 입력해 주세요."
4 }
```

이메일 중복확인 API [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/check?email=zhyun@gmail.com'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/check

Query String [↗](#)

```
1 ?email=zhyun@gmail.com
```

Key	타입	필수 여부	설명
email	String	필수	아이디로 사용 될 이메일 주소

Response [↗](#)

성공 응답 - 중복되지 않음 [↗](#)

```
1 {
2   "status": true,
3   "message": "사용 가능한 이메일입니다. 이메일 인증을 진행해주세요!"
4 }
```

실패 응답 [↗](#)

CASE 1 : 중복됨 [↗](#)

```
1 {
2   "status": false,
3   "message": "이미 사용중인 이메일입니다."
4 }
```

CASE 2 : 이메일 형식에 어긋남 [↗](#)

```
1 {
2   "status": false,
3   "message": "올바른 이메일 주소를 입력해 주세요."
4 }
```

CASE 3 : 이메일 값이 비어있음 [↗](#)

```
1 {
2   "status": false,
3   "message": "이메일 주소를 입력해 주세요."
4 }
```

닉네임 중복확인 API [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/check?nickname=얼거스'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/check

Query String [↗](#)

```
1 ?nickname=얼거스
```

Key	타입	필수 여부	설명
nickname	String	필수	닉네임

Response [↗](#)

성공 응답 - 중복되지 않음 [↗](#)

```
1 {
2   "status": true,
3   "message": "사용 가능한 닉네임 입니다."
4 }
```

실패 응답 [↗](#)

CASE 1 : 중복됨 [↗](#)

```
1 {
2   "status": false,
3   "message": "이미 사용중인 닉네임 입니다."
4 }
```

CASE 2 : 닉네임 형식에 어긋남 [🔗](#)

```
1 {
2   "status": false,
3   "message": "닉네임은 6글자 이하로 작성해야 합니다."
4 }
```

CASE 3 : 닉네임 값이 비어있음 [🔗](#)

```
1 {
2   "status": false,
3   "message": "닉네임을 입력해 주세요."
4 }
```

이메일 인증 API - 이메일 전송 [🔗](#)

- api 진입시 인증 코드를 메일로 전송

Request [🔗](#)

Request Syntax [🔗](#)

```
1 curl -X POST 'http://localhost/api/user/check/auth'
```

API 호출 방식 [🔗](#)

메서드	요청 url
POST	/api/user/check/auth

Request Body [🔗](#)

```
1 {
2   "email": "zhyun@gmail.com"
3 }
```

파라미터	타입	필수 여부	설명
email	String	필수	아이디로 사용 될 이메일주소

Response [🔗](#)

이메일 인증 메일 발송 성공 응답 [🔗](#)

```
1 {
2   "status": true,
3   "message": "전송 되었습니다."
4 }
```

이메일 인증 메일 발송 실패 응답 예시 - Case 1 : 중복확인 안 된 이메일 [↗](#)

```
1 {
2   "status": false,
3   "message": "이메일 중복확인을 먼저 진행해주세요."
4 }
```

이메일 인증 메일 발송 실패 응답 예시 - Case 2 : 잘못된 이메일 형식 [↗](#)

```
1 {
2   "status": false,
3   "message": "이메일 형식이 올바르지 않습니다."
4 }
```

이메일 인증 API - 메일 인증 [↗](#)

- 이메일 주소로 전송된 코드를 Query String으로 입력

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/check/auth?code=SDFG2345'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/check/auth

Query String [↗](#)

```
1 ?code=ASDG234
```

Key	타입	필수 여부	설명
code	String	필수	서버에서 생성되는 이메일 인증 코드

Response [↗](#)

이메일 인증 성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
```

```
3   "message": "인증 되었습니다."
4 }
```

실패 응답 예시 - Case 1 : 인증 코드가 일치하지 않음 [↗](#)

```
1 {
2   "status": false,
3   "message": "인증 번호가 일치하지 않습니다."
4 }
```

실패 응답 예시 - Case 2 : 만료된 인증 코드 [↗](#)

```
1 {
2   "status": false,
3   "message": "인증 번호가 만료되었습니다. 인증을 다시 진행해주세요!"
4 }
```

회원 탈퇴 API [↗](#)

- 논리 삭제 후 30일 후에 물리 삭제

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X DELETE 'http://localhost/api/user/withdrawal'
```

API 호출 방식 [↗](#)

메서드	요청 url
DELETE	/api/user/withdrawal

Response [↗](#)

회원 탈퇴 성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "얼거스(zhyun@gmail.com)님 탈퇴되었습니다."
4 }
```

로그인 API [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/login'
```

API 호출 방식 [↗](#)

메서드	요청 url
POST	/api/user/login

Request Body [↗](#)

```
1 {
2   "email": "zhyun@gmail.com",
3   "password": "secret!!"
4 }
```

파라미터	타입	필수 여부	설명
email	String	필수	아이디로 사용 될 이메일주소
password	String	필수	비밀번호

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "얼거스님 로그인 성공"
4 }
```

실패 응답 예시 - Case 1 : 잘못된 email [↗](#)

```
1 {
2   "status": false,
3   "message": "없는 사용자입니다."
4 }
```

실패 응답 예시 - Case 2 : 잘못된 비밀번호 [↗](#)

```
1 {
2   "status": false,
3   "message": "계정 정보가 일치하지 않습니다."
```



```
4 }
```

로그아웃 API [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/logout'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/logout

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "얼거스(zhyun@gmail.com)님 로그아웃 성공"
4 }
```

계정 조회 API - 본인 계정 상세 조회 [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/{id}'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/{id}

- path parameter : user id

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "상세 조회",
4   "result": {
5     "id": 1,
6     "email": "zhyun@gmail.com",
7     "nickname": "얼거스",
8     "role": "ADMIN",
9     "created_at": "2023-12-10T21:54:32.321",
10    "modified_at": "2023-12-10T21:54:32.321"
11  }
12 }
```

실패 응답 예시 - Case 1 : 다른 사람 id 조회 (권한 없음) [↗](#)

```
1 {
2   "status": false,
3   "message": "권한이 없습니다."
4 }
```

계정 조회 API - 전체 계정 상세 조회 [↗](#)

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/all'
```

API 호출 방식 [↗](#)

메서드	요청 url
GET	/api/user/all

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "전체 계정 상세 조회",
4   "result": [
5     {
```

```

6      "id": 1,
7      "email": "zhyun@gmail.com",
8      "nickname": "얼거스",
9      "role": "ADMIN",
10     "created_at": "2023-12-10T21:54:32.321",
11     "modified_at": "2023-12-10T21:54:32.321"
12   },
13   {
14     "id": 2,
15     "email": "gimwlgus@kakao.com",
16     "nickname": "김얼거스",
17     "role": "MEMBER",
18     "created_at": "2023-12-10T21:54:32.321",
19     "modified_at": "2023-12-10T21:54:32.321"
20   },
21   {
22     "id": 3,
23     "email": "gimwlgus@gmail.com",
24     "nickname": "닉넴",
25     "role": "WITHDRAWAL",
26     "created_at": "2023-12-10T21:54:32.321",
27     "modified_at": "2023-12-10T21:54:32.321"
28   },
29 }
30 }

```

실패 응답 예시 - Case 1 : ADMIN 권한이 아닌 계정의 API 사용 [🔗](#)

```

1 {
2   "status": false,
3   "message": "권한이 없습니다."
4 }

```

계정 수정 API [🔗](#)

- 이메일 변경시 인증 process 진행 후 변경

Request [🔗](#)

Request Syntax [🔗](#)

```
1 curl -X GET 'http://localhost/api/user/{id}'
```

API 호출 방식 [🔗](#)

메서드	요청 url
PUT	/api/user/{id}

- path parameter : user id

Request Body [↗](#)

```
1 {
2   "id": 1,
3   "email": "zhyun@naver.com",
4   "nickname": "오예"
5 }
```

파라미터	타입	필수 여부	설명
email	String	필수	아이디로 사용 될 이메일주소
nickname	String	필수	닉네임

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {
2   "status": true,
3   "message": "계정 정보가 수정되었습니다."
4 }
```

실패 응답 예시 - Case 1 : 닉네임 수정시 닉네임 형식에 어긋남 [↗](#)

```
1 {
2   "status": false,
3   "message": "닉네임은 6글자 이하로 작성해야 합니다."
4 }
```

실패 응답 예시 - Case 2 : 닉네임 수정시 중복 검사 안함 [↗](#)

```
1 {
2   "status": false,
3   "message": "닉네임 중복 확인을 진행해주세요."
4 }
```

계정 권한 수정 API [↗](#)

- ADMIN 만 사용 가능

Request [↗](#)

Request Syntax [↗](#)

```
1 curl -X GET 'http://localhost/api/user/role'
```

API 호출 방식 [↗](#)

메서드	요청 url
PUT	/api/user/role

- path parameter : user id

Request Body [↗](#)

```
1 {  
2   "role": "MEMBER"  
3 }
```

파라미터	타입	필수 여부	설명
role	String	필수	사용자 권한

Response [↗](#)

성공 응답 예시 [↗](#)

```
1 {  
2   "status": true,  
3   "message": "닉네임님 권한이 MEMBER(으)로 수정되었습니다."  
4 }
```

실패 응답 예시 - Case 1 : 없는 사용자 id [↗](#)

```
1 {  
2   "status": false,  
3   "message": "잘못된 요청입니다."  
4 }
```

실패 응답 예시 - Case 2 : 권한이 없는 사용자의 진입 [↗](#)

```
1 {  
2   "status": false,  
3   "message": "권한이 없습니다."  
4 }
```

