**API Specification Doc**

**(*Zalo Fake App*)**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | Feb – 25 – 2024 | Bui Tri Thuc | Authenticate and Register API |
| 1.0.1 | March – 05 – 2024 | Bui Tri Thuc | Change the params (device\_id) of Login API request |
| 2.0 | March – 26 – 2024 | Bui Tri Thuc | Add feature : User API |
| 3.0 | April – 12 – 2024 | Bui Tri Thuc | Add feature : Group API |

[I. Authentication Features 4](#_Toc163845052)

[1. Login 4](#_Toc163845053)

[Request 4](#_Toc163845054)

[Response 4](#_Toc163845055)

[2. Logout 5](#_Toc163845056)

[Request 5](#_Toc163845057)

[Response 5](#_Toc163845058)

[3. Register 6](#_Toc163845059)

[Request 6](#_Toc163845060)

[Response 6](#_Toc163845061)

[4. Refresh Token 7](#_Toc163845062)

[Request 7](#_Toc163845063)

[Response 7](#_Toc163845064)

[II. User Features 8](#_Toc163845065)

[1. Update profile 8](#_Toc163845066)

[Request 8](#_Toc163845067)

[Response 8](#_Toc163845068)

[2. Update avatar 9](#_Toc163845069)

[Request 9](#_Toc163845070)

[Response 9](#_Toc163845071)

[3. Update background 10](#_Toc163845072)

[Request 10](#_Toc163845073)

[Response 10](#_Toc163845074)

[4. Get user information 11](#_Toc163845075)

[Request 11](#_Toc163845076)

[Response 11](#_Toc163845077)

[III. Group Features 12](#_Toc163845078)

[1. Create Group 12](#_Toc163845079)

[Request 12](#_Toc163845080)

[Response 12](#_Toc163845081)

[2. Update Group 13](#_Toc163845082)

[Request 13](#_Toc163845083)

[Response 13](#_Toc163845084)

[3. Get Group By ID 14](#_Toc163845085)

[Request 14](#_Toc163845086)

[Response 14](#_Toc163845087)

[Request 15](#_Toc163845088)

[Response 15](#_Toc163845089)

[5. Add Members 15](#_Toc163845090)

[Request 15](#_Toc163845091)

[Response 16](#_Toc163845092)

[6. Remove Members 17](#_Toc163845093)

[Request 17](#_Toc163845094)

[Response 17](#_Toc163845095)

[7. Delete Group 18](#_Toc163845096)

[Request 18](#_Toc163845097)

[Response 18](#_Toc163845098)

[IV. Conventions 19](#_Toc163845099)

[Status Codes 19](#_Toc163845100)

# Authentication Features

## 1. Login

* Authenticate the user with the system and obtain the token

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/auth/login |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST  POST | phone  password  device\_id [optional] | string  string  string ( get device id ) |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "user": {  "profile": {  "name": "Tester02"  },  "phone": "0909878765",  "createdAt": "2024-02-28T08:18:54.887Z",  "friendList": [],  "groupList": [],  "posts": [],  "moments": [],  "\_\_v": 0  },  "accessToken": < Access Token >  } | Login on Web**(2)** |
| 200 | {  "user": {  "profile": {  "name": "Tester02"  },  "phone": "0909878765",  "createdAt": "2024-02-28T08:18:54.887Z",  "friendList": [],  "groupList": [],  "posts": [],  "moments": [],  "\_\_v": 0  },  "accessToken": < Access Token >,  "refreshToken": <Refresh Token>  } | Login on Mobile |
| 401 | {"message":"Invalid phone or password"} |  |
| 400 | {"message":"error"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

***(1) :*** If login form web applications, device\_id is not required, server will be genarate and manager this base on user IP Address

***(2) :*** If login form web applications, refresh token will be add to cookies without return it in response body.

## 2. Logout

* Logout the user with the system and remove the token

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/auth/logout |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEADER.Cookie | refreshToken | string |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {"message":"Logout successfully"} |  |
| 403 | {"message":"You’re not authenticate !"} | Missing refresh token value |
| 403 | {"message":"Refresh token is not valid !"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

## 3. Register

* Register the user and account of this user with the system

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/auth/register |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST  POST | phone  password  name | string  string  string |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 201 | {  "user": {  "phone": <phone of user sign up>,  "profile": {  "name": <Name of user sign up>  }  }  } |  |
| 400 | {"message":<error while create user>} |  |
| 409 | {"message":"Phone number already exists"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

## 4. Refresh Token

* To ensure security, an access token only remains valid for 10 minutes, after which it needs to be refreshed whenever it expires

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/auth/refreshToken |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| HEADER.Cookie | refreshToken | string |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "newAccessToken": <New access token>  } | Request form Web**(2)** |
| 200 | {  "newAccessToken": <New access token>,  "newRefreshToken": < New Refresh token>  } | Request form Mobile |
| 403 | {"message":"You’re not authenticated !"} | Missing refresh token value |
| 403 | {"message":"Refresh Token í not valid !"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

***(2) :*** If request form web applications, the new refresh token will be add to cookies without return it in response body.

# User Features

## Update profile

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/users/update-profile |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST  POST  POST  POST | userId  name  email  gender  dob | string  string  string  string (enum {male, female, orther})  string (format {yyyy-MM-dd}) |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "message": "User updated successfully",  "user": {  "profile": {  "avatar": {  "url": "<image url>",  "public\_id": "iamge public id"  },  "background": {  "url": "<image url>",  "public\_id": "iamge public id"  },  "name": "<user’s name>",  "dob": "<user’s dob>",  "gender": "male"  },  "phone": "<user’s phone>",  "createdAt": "<Date>",  "\_\_v": 1,  "lastActive": "<Date>",  "email": "user’s email"  }  } |  |
| 404 | {"message":"User not found"} |  |
| 400 | {"message":"Can't update this user"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

## Update avatar

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/users/update-avatar |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST | userId  avatar | string  file |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "message": "Avatar uploaded successfully",  "avatar": {  "url": "<new image url>",  "public\_id": "new image url"  }  } |  |
| 404 | {"message":"User not found"} |  |
| 400 | {"message":"userId is required"} | UserId not found in request body |
| 400 | {"message":"Image is required"} | Image file not found in request body |
| 400 | {"message":"Can't update avatar"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

## Update background

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/users/update-background |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST | userId  background | string  file |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "message": "Background uploaded successfully",  "avatar": {  "url": "<new image url>",  "public\_id": "new image url"  }  } |  |
| 404 | {"message":"User not found"} |  |
| 400 | {"message":"userId is required"} | UserId not found in request body |
| 400 | {"message":"Image is required"} | Image file not found in request body |
| 400 | {"message":"Can't update background"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

## Get user information

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | /api/users/get/phone/{phone number} |
| **GET** | /api/users/get/uid/{user id} |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "user": {  "profile": {  "avatar": {  "url": "<image url>",  "public\_id": "image public id"  },  "background": {  "url": "<image url>",  "public\_id": "image public id"  },  "name": "<user’s name>",  "dob": "<user’s dob>",  "gender": "male"  },  "email": "<user’s email>",  "phone": "<user’s phone>"  }  } |  |
| 404 | {"message":"User not found"} |  |
| 400 | {"message":"Can’t not get this user"} |  |
| 500 | {"message":"Something went wrong. Please try again later."} |  |

# Group Features

## Create Group

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/groups/create |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST  POST | name  members | string  array ([user id])  contrain : members.length >= 2 |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 201 | {  "group": {  "groupName":<group name>,  "avatar": {  "url": "https://res.cloudinary.com/dq3pxd9eq/image/upload/group\_avatar.jpg",  "public\_id": "group\_avatar"  },  "createAt": Date,  "createBy": <UserId>,  "conversation":<conversation id>,  "\_id": <group Id>,  "\_\_v": 0  }  } | avatar is initialize by default  create by is user login, who sent request |
| 404 | {"error":"User not found"} |  |
| 400 | {"error":"name is require"} |  |
| 400 | {"error":"members is require"} |  |
| 500 | {"error":"Something went wrong. Please try again later."} |  |

## Update Group

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **PUT** | /api/groups/update/{groupId} |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| **PUT PUT** | name  avatar | string  file |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "group": {  "avatar": {  "url": <avt url>  "public\_id": <avt id>  },  "\_id": <group Id>,  "groupName":<New Name>,  "createAt": <Date>,  "createBy": <userId>,  "conversation": <conversationId>,  "\_\_v": 0  }  } |  |
| 404 | {"error":"Group not found"} |  |
| 500 | {"error":"Something went wrong. Please try again later."} |  |

### Get Group By ID

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/groups/get/{groupId} |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "group": {  "avatar": {  "url": <url>  "public\_id": <avt id>  },  "\_id": < groupId >,  "groupName": <group name>,  "createAt":<Date>,  "createBy": <user id>,  "conversation": {  "\_id": <conversationId>,  "participants": [  <Array Of members Id>  ],  "messages": [  < Array Of Message Id>  ],  "tag": "group",  "createdAt": <Date>,  "\_\_v": 0  },  "\_\_v": 0  }  } |  |
| 404 | {"error":"Group not found"} |  |
| 500 | {"error":"Something went wrong. Please try again later."} |  |

1. Get All Group Of User Login

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | /api/groups/all |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | [  {  "avatar": {  "url": <url>  "public\_id": <avt id>  },  "\_id":<groupId>,  "groupName": <Group’s name>,  "createAt": Date ,  "createBy": <userId>,  "conversation": {  "\_id": <conversationId>,  "participants": [  <Array Of members Id>  ],  "messages": [  < Array Of Message Id>  ],  "tag": "group",  "createdAt": <Date>,  "\_\_v": 0  },  "\_\_v": 0  }  ] | Result As Array |
| 500 | {"error":"Something went wrong. Please try again later."} |  |

## Add Members

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/groups/addMembers/{groupId} |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| **POST** | members | array ([user id])  contrain : members.length >= 1 |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "message": [  "Member <member name> already in group. ",  "Member <member name> added to group. "  ],  "group": {  "avatar": {  "url": <url>  "public\_id": <avt>  },  "\_id": <groupId>,  "groupName": <group’s name>,  "createAt": <Date>,  "createBy": <userId>,  "conversation": {  "\_id": <conversationId>,  "participants": [  <Array Of members Id>  ],  "messages": [  < Array Of Message Id>  ],  "tag": "group",  "createdAt": <Date>,  "\_\_v": 1  },  "\_\_v": 0  }  } | "The 'message' array will contain an array of notifications indicating which members have been added and which members have not been added because they already exist in group." |
| 500 | {"error":"Something went wrong. Please try again later."} |  |

## Remove Members

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /api/groups/removeMembers/{groupId} |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| **POST** | members | array ([user id])  contrain : members.length >= 1 |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "group": {  "avatar": {  "url": <url>  "public\_id": <avt>  },  "\_id": <groupId>,  "groupName": <group’s name>,  "createAt": <Date>,  "createBy": <userId>,  "conversation": {  "\_id": <conversationId>,  "participants": [  <That will be new Array Of members Id>  ],  "messages": [  < Array Of Message Id>  ],  "tag": "group",  "createdAt": <Date>,  "\_\_v": 1  },  "\_\_v": 0  }  } |  |
| 404 | {"error":"Group not found."} |  |
| 403 | {"error":"You are not authorized"} | Only group administrators have the authority to remove members. |
| 400 | {"error":"members is require "} |  |
| 400 | {"error":"Group must have at least 2 members"} | The number of members to be deleted exceeds the minimum required number for a group. |
| 500 | {"error":"Something went wrong"} |  |

## Delete Group

### Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **DELETE** | /api/groups/delete/{groupId} |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| **DELETE** | members | array ([user id])  contrain : members.length >= 1 |

### Response

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Note** |
| 200 | {  "group": {  "avatar": {  "url": <url>  "public\_id": <avt>  },  "\_id": <groupId>,  "groupName": <group’s name>,  "createAt": <Date>,  "createBy": <userId>,  "conversation": {  "\_id": <conversationId>,  "participants": [  <That will be new Array Of members Id>  ],  "messages": [  < Array Of Message Id>  ],  "tag": "group",  "createdAt": <Date>,  "\_\_v": 1  },  "\_\_v": 0  }  } |  |
| 404 | {"error":"Group not found."} |  |
| 403 | {"error":"You are not authorized"} | Only group administrators have the authority to remove members. |
| 500 | {"error":"Something went wrong"} |  |

# Conventions

* **Client** - Client application.
* **Status** - HTTP status code of response.
* **Type –** HTTP Request Method (GET, POST, PUT, PATH, DELETE…)
* All response are in JSON format.
* All request parameters are mandatory unless explicitly marked as [optional]

## Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

2XX - Success of some kind

4XX - Error occurred in client’s part

5XX - Error occurred in server’s part

|  |  |
| --- | --- |
| **Status Code** | **Description** |
| 200 | OK |
| 201 | Created |
| 202 | Accepted (Request accepted, and queued for execution) |
| 400 | Bad request |
| 401 | Authentication failure |
| 403 | Forbidden |
| 404 | Resource not found |
| 405 | Method Not Allowed |
| 409 | Conflict |
| 412 | Precondition Failed |
| 413 | Request Entity Too Large |
| 500 | Internal Server Error |
| 501 | Not Implemented |
| 503 | Service Unavailable |