A person and person smoking

Description automatically generated

|  |
| --- |
|  |
| TODO APPLICATION |

API MODULE DOCUMENTATIONS

TABLE OF CONTENTS

1. AUTHENTICATION
2. SIGN IN

URL: POST /api/auth/sign-in

BODY: {

username: <String>(255)[ required ],

password: <String>(255)[ required ]

}

Sign-in request gets authentication infors, validates them and responds to the client if the infors are valid or not.

There are 3 responses may show up when the API is called:

1. Signed in successfully

With correct username and password, the response will bring a session token with it, this can be use as authorization token in future.

Response will have the structure as below:

200 OK : {

success: true,

data: {

success: true,

content: “Successfully signed in”,

token: “[ TOKEN STRING ]”,

code: “SU\_AU103”,

user: {

username: <String>,

email: <String>,

phone: <String>,

avatar: <String>

}

}

}

1. Signed in failed

There are several reasons to cause the server response this way: User does not exist, wrong password. But we won’t show the detail reason due to security purpose.

200 OK : {

success: true,

data: {

success: false,

content: “Signed in fail due to wrong information”,

code: “ER\_AU104”

}

}

1. Bad request

Wrong request body structure.

400 BAD\_REQUEST : {}

1. SIGN UP

URL: POST /api/auth/sign-up

BODY: {

username: <String>(255)[ required ],

password: <String>(255)[ required ],

fullname: <String>(255)[ required ],

phone: <String>(20)[ optional ],

email: <String>(255)[ optional ]

}

This API creates a new user with given infors.

1. Signed up successfully

With all data in correct format, a new user will be created.

The server will automatically generate a default image for newly user.

200 OK : {

success: true,

data: {

success: true,

content: “User created”,

token: “[ TOKEN STRING ]”,

code: “SU\_AU102”,

user: {

username: <String>,

fullname: <String>,

email: <String>,

phone: <String>,

avatar: <String>

}

}

}

After create user, server will generate a token and attach it to the response as well.

1. Sign up failed due to wrong data format.

This may happy when weird data fields exist in the request body or (and) the data itself are in wrong fomat.

Beside max length rule as above, we define min length rule for required fields:

* username: 8
* password: 8
* fullname: 1

400 BAD\_REQUEST : {

success: true,

data: {

success: false,

content: “Wrong data format”,

code: “ER\_UN101”

}

}

1. Sign up failed due to the username already existed

Username in request body may have already been existed in database.

Although our database can instantly detect that error but it could suspend the server. We will take a step ahead and handle that error.

200 OK : {

success: true,

data: {

success: false,

content: “Username already existed”,

code: “ER\_UN102”

}

}

1. Bad request

Wrong request body structure.

400 BAD\_REQUEST : {}

1. UPDATE USER INFORMATION

URL: PUT /api/auth/infors

BODY: {

fullname: <String>(255)[ optional ],

phone: <String>(20)[ optional ],

email: <String>(255)[ optional ]

}

HEADER : {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

Update user infor using provided infor from request body, only those fields which contain data will be updated.

1. Update successfully

With valid username, token and at least one provided infor field, user can be updated:

200 OK {

success: true,

data: {

success: true,

content: “User updated”,

code: “SU\_AU101”

}

}

1. User no longer exists

The user might be deleted during its session, if it were removed by any chance, instantly response error.

404 NOT\_FOUND : {

success: false,

data: {

success: false,

content: “User not found or removed”,

code: “ER\_AU101”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. Bad request

Wrong request body structure.

400 BAD\_REQUEST : {}

1. UPDATE USER AVATAR

URL: PUT /api/auth/avatar

BODY: {

image: <Base64Image>[ required ]

}

HEADER : {

“Authorization”: “[ TOKEN STRING]”

}

Change user avatar.

1. Successfully update.

200 OK {

success: true,

data: {

success: true,

content: “User updated”,

code: “SU\_AU101”

}

}

1. Invalid file format.

Users somehow chose wrong file format, detect that error and response error.

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid file format”,

code: “ER\_AU103”

}

}

1. User no longer exits *( same as above )*
2. Invalid token *( same as above )*
3. Bad request *( same as above )*
4. SIGN OUT

URL: POST /api/auth/sign-out

HEADER : {

“Authorization”: “[ TOKEN STRING]”

}

Revoke token

1. Successfully revoked token

With valid token, the server revokes token and remove it from database.

200 OK : {

success: true,

data: {

success: true,

content: “Token roveked”,

code: “SU\_AU104 ”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. REMINDERS
2. Global enums

There is some ENUM sets will be defined before we start

1. Statuses

A Reminder can have one of these status: Active, Disabled, Deleted or Permanently Deleted

* Active: This status allows reminder itself join every activity in software flow.
* Disabled: Reminders with this status still exist in configuration section but won’t be shown even its triggers are activated
* Deleted: Reminders that were marked as deleted will be disabled and only be shown in trash section waiting to be deleted permenantly.
* Permenantly Deleted : Reminders with this status will be hidden permanent

1. Repeat

Repetition decides if a specified reminder will be repeat at the same datetime after a interval of time.

There are several options:

* Every day
* Every week
* Every month
* Every year

1. Categories

Reminder can have one of four following categories:

* Working: Reminders for working purpose
* Reading: Books and documents
* Watching: Film, anime or something else
* Others: Things which are not included in others categories

These categories are const and defined as a list in /global\_enum.py => DEFAULT\_CATEGORIES\_LIST

1. CRUB methods
2. GET ALL

URL: GET /api/reminders/scheduled?from=FROM&to=TO

QUERY: {

FROM: <Int>,

TO: <Int>,

}

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

Get all scheduled reminders belong to current user.

Max offset between FROM & TO is 24

1. Successfully retrieving data

200 OK : {

success: true,

data: {

success: true,

content: “Successfully retrieve remiders”,

code: “SU\_AU201”,

remiders: [

{

remider\_id: <Int>,

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>,

status: <String>,

categories: <String>[],

jobs: [

{

job\_id: <Int>,

title: <String>,

description: <String>,

categories: <String>[]

},

]

},

]

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. GET ONE

URL: GET /api/reminders/scheduled/<:reminder\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

Get one scheduled reminder

1. Successfully retrieved data

200 OK : {

success: true,

data: {

success: true,

content: “Successfully retrieve remider”,

code: “SU\_AU201”,

remider: {

remider\_id: <Int>,

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>,

status: <String>,

categories: <String>[],

jobs: [

{

job\_id: <Int>,

title: <String>,

description: <String>,

categories: <String>[]

},

]

},

}

}

1. Not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Not found”,

code: “ER\_AU201”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. POST

URL: POST /api/reminders/scheduled/

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

BODY: {

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>

categories: <Int>[ ],

jobs: [

{

title: <String>,

description: <String>,

categories: <Int>

},

]

}

1. Successfully created

With all data validated, new reminder will be created then response json as follows:

201 CREATED : {

success: true,

data: {

success: true,

content: “Successfully created”,

code: “SU\_AU202”,

remider: {

remider\_id: <Int>,

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>,

status: <String>,

categories: <String>[],

jobs: [

{

job\_id: <Int>,

title: <String>,

description: <String>,

categories: <String>[]

},

]

},

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. Invalid repetition type

Repeat field’s value not included in ENUM set which defined in /global\_enum.py DEFAULT\_REPETITION\_LIST

400 BAD REQUEST : {

success: false,

data: {

success: false,

content: “Invalid repetition type”,

code: “ ER\_AU202 ”

}

}

1. Invalid category

Category field includes invalid category id.

400 BAD REQUEST : {

success: false,

data: {

success: false,

content: “Invalid category set”,

code: “ER\_AU203”

}

}

1. Children job have invalid category

400 BAD REQUEST : {

success: false,

data: {

success: false,

content: “Invalid children job category set”,

code: “ER\_AU204”

}

}

1. Wrong data format

Any weird fields appear in request body will lead to this response.

400 BAD\_REQUEST : {

success: true,

data: {

success: false,

content: “Wrong data format”,

code: “ER\_AU205”

}

}

1. PUT

URL: PUT /api/reminders/scheduled/<:reminder\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

BODY: {

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>

categories: <Int>[ ]

}

Update reminder general infors, descendent jobs will be updated separately for performance optimizing purpose

1. Successfully updated

Update data and response new object.

200 OK : {

success: true,

data: {

success: true,

content: “Successfully updated”,

code: “SU\_AU203”,

remider: {

remider\_id: <Int>,

title: <String>,

description: <String>,

time: <Time>,

place: <String>,

tag: <String>,

repeat: <String>,

status: <String>,

categories: <String>[],

jobs: [

{

job\_id: <Int>,

title: <String>,

description: <String>,

categories: <String>[]

},

]

},

}

}

1. Reminder not found

No reminder matches given reminder\_id

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Reminder not found”,

code: “ER\_AU201”

}

}

1. Wrong data format

Any weird fields appear in request body will lead to this response.

400 BAD\_REQUEST : {

success: true,

data: {

success: false,

content: “Wrong data format”,

code: “ER\_AU205”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. DELETE

URL: DELETE /api/reminders/scheduled/<:reminder\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

Update scheduled reminder status to permanently deleted

1. Successfully removed

Remove and response

200 OK : {

success: true,

data: {

success: true,

content: “Successfully removed scheduled reminder”,

code: “SU\_AU204”

}

}

1. Not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Reminder not found”,

code: “ER\_AU201”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. PATCH

URL: PATCH /api/reminders/scheduled/<:reminder\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

BODY: {

status: <String>

}

Update scheduled reminder status

1. Successfully update

200 OK : {

success: true,

data: {

success: true,

content: “Successfully update scheduled reminder”,

code: “SU\_AU203”

}

}

1. Not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Reminder not found”,

code: “ER\_AU201”

}

}

1. Invalid status value

400 BAD REQUEST: {

success: true,

data: {

success: false,

content: “Invalid status value”,

code: “ER\_AU207”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. Jobs

We won’t implement GET method (for now) because jobs are always called within scheduled reminders.

1. POST

URL: POST /api/reminders/jobs/

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

BODY: {

reminder\_id: <Int>,

title: <String>,

description: <String>,,

categories: <Int>[]

}

Create a descendent job with given reminder\_id

1. Successfully added

200 OK : {

success: true,

data: {

success: true,

content: “Successfully job added”,

code: “SU\_AU205”,

job: {

job\_id: <Int>,

title: <String>,

description: <String>,

reminder\_id: <Int>

}

}

}

1. Reminder not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Reminder not found”,

code: “ER\_AU201”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. Wrong data format

400 BAD\_REQUEST : {

success: true,

data: {

success: false,

content: “Wrong data format”,

code: “ER\_AU205”

}

}

1. PUT

URL: PUT /api/reminders/jobs/<:job\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

BODY: {

title: <String>,

description: <String>,,

categories: <Int>[]

}

Update job information, for now, we don’t implement the change reminder module because it seems no use at all.

1. Successfully updated

200 OK : {

success: true,

data: {

success: true,

content: “Successfully updated job”,

code: “SU\_AU206”,

job: {

job\_id: <Int>,

title: <String>,

description: <String>,

reminder\_id: <Int>

}

}

}

1. Not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Reminder not found”,

code: “ER\_AU201”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. Wrong data format

400 BAD\_REQUEST : {

success: true,

data: {

success: false,

content: “Wrong data format”,

code: “ER\_AU205”

}

}

1. DELETE

URL: DELETE /api/reminder-jobs/<:job\_id>

HEADERS: {

“Authorization”: “TOKEN [ TOKEN STRING]”

}

Delete job from reminder.

1. Successfully delete

200 OK : {

success: true,

data: {

success: true,

content: “Successfully delete job”,

code: “SU\_AU207”

}

}

1. Not found

404 NOT FOUND : {

success: true,

data: {

success: false,

content: “Job not found”,

code: “ER\_AU206”

}

}

1. Invalid token

403 FOBIDDEN : {

success: false,

data: {

success: false,

content: “Invalid token”,

code: “ ER\_AU102”

}

}

1. HABITS
2. CRUB methods
3. Jobs
4. SECTION - HOME
5. SECTION - CATEGORY
6. SECTION - COMPLETED
7. SECTION - TRASH