API document:

1. url: /user
   1. method: POST
   2. function: create user
   3. request body is required,

format:

{

"username": string,

"password": string

}

For example:

{

"username": "test1",

"password": "12345678aA"

}

* 1. validation:

when username or password is not string:

{

**"username": 13,**

"password": "123456A"

}

and there will be an Unprocessable Entity Error

{

"detail": [

{

"type": "string\_type",

"loc": [

"body",

"username"

],

**"msg": "Input should be a valid string",**

"input": 13,

"url": "https://errors.pydantic.dev/2.6/v/string\_type"

}

]

}

* 1. Response:
     1. **when a user created**, response body

{

"success": true

}

status code: 201

* + 1. if the **user is existed**,

{

"success": false,

"reason": "Username already existed"

}

status code: 409 (fail)

* + 1. **input username failed**:

{

"username": "a",

"password": "12345678aA"

}

response:

{

"success": false,

"reason": "Username length minimum is 3, and maximun is 32."

}

status code: 422 (fail)

* + 1. **input password failed**:

{

"username": "test3",

"password": "123456A"

}

{

"success": false,

"reason": "Password length minimum is 8, and maximun is 32, with at least 1 uppercase, 1 lowercase, 1 number."

}

status code: 422 (fail)

1. url: /login
   1. method: POST
   2. function: validate user and its password
   3. request body is required,

format:

{

"username": string,

"password": string

}

For example:

{

"username": "test1",

"password": "12345678aA"

}

* 1. validation:

when username or password is not string:

{

"username": "user3",

"password": 14

}

and there will be an Unprocessable Entity Error:

{

"detail": [

{

"type": "string\_type",

"loc": [

"body",

"password"

],

"msg": "Input should be a valid string",

"input": 14,

"url": "https://errors.pydantic.dev/2.6/v/string\_type"

}

]

}

* 1. Response:
     1. when the **user successfully validated**:

{

"username": "user10",

"password": "12345678aA"

}

response:

{

"success": true

}

* + 1. when the user is **not existed**:

{

"username": "user100",

"password": "12345678aA"

}

response:

{

"success": false,

"reason": "The user not existed."

}

status code: 401 (fail)

* + 1. when the **password is wrong**:

{

"username": "user10",

"password": "12345678a"

}

response:

{

"success": false,

"reason": "The password is not corrected."

}

status code: 401 (fail)

* + 1. if password is wrong for 5 times, **lock the user**:

{

"success": false,

"reason": "The user is being locked"

}

status code: 403 (fail)