Wallet Management Backend Nestjs

Emurgo Labs

Version: 1.0

Content

1. Database Design.. 3
2. APIs Usage..4
2.1. /user/signup.. 4
2.2. /user/confirm-otp..5
2.3. /user/check/username.. 6
2.4. /user/check/email..7
2.5. /user/signin.. 8
2.6. /user/signin/email..9
2.7. /user/profile.. 10
2.8. /user/all.. 11
2.9. /user/change-password.. 12
2.10. /user/forgot-password.. 13
2.11. /user/reset-password.. 14
3. Deployment..15

1. Database Design

Table User:

Field Name	Type	Description
id	number	A unique identifier (primary key) for each user record in the table, which auto-increments
username	string	The user’s chosen display or login name
password	string	The user’s login password
firstName	string	User’s given name
lastName	string	User’s family name
email	string	User’s email address
mnemonic	string	The 24-word recovery phrase for the user’s crypto wallet, generated randomly during sign-up and is unique to each user. This phrase is encrypted before being stored in the database and decrypted only when the user wants to transfer funds from their wallet
walletAddress	string	The Cardano wallet address, which is derived from the mnemonic
emailVerified	boolean	Indicates whether the user has verified their email after sign-up. To sign in, the user must verify their email by entering the OTP sent to their registered email address
role	string	Defines the user’s permission level (role) in the system. There are two roles: ‘admin’ and ‘user’. Only admins can query information about all users in the system
otp	string	One-Time Password code for email verification (after sign-up), or password reset

2. APIs Usage

2.1. /user/signup

Method: POST

Purpose: This API is used to sign up a new user. During the process, the system generates a random 24-word recovery phrase (mnemonic) and creates a new Cardano wallet address for the user. After signing up, the user must verify their email by entering the OTP sent to them to sign in. If the email is not verified, the user cannot access the system.

Payload example:

```
{
  "username": "congle",
  "password": "123456",
  "firstName": "Cong",
  "lastName": "Le",
  "email": "lecongbk93@gmail.com"
}
```

Success response example:

```
{
  "id": 2,
  "username": "congle",
  "firstName": "Cong",
  "lastName": "Le",
  "email": "lecongbk93@gmail.com",
  "walletAddress":
"addr_test1qz8p8rp4cvmnepjhh53j3ewqzsfmu4r3lcpyqpen4cpt4jcdetjl6fr0y9jdudksm22hx8x22
q3p3q3zyxy84mlefqwslzevet"
}
```

If the username already exists, you will receive this error message:

```
{
  "statusCode": 400,
  "message": "Username existed!"
}
```

If the email already exists, you will receive this error message:

```
{
  "statusCode": 400,
  "message": "Email existed!"
}
```

2.2. /user/confirm-otp

Method: POST

Purpose: This API is used to confirm a user's email with an OTP after sign-up. Once the email is verified, the user can sign in with their account.

Payload example:

```
{
  "email": "lecongbk93@gmail.com",
  "otp": "536455"
}
```

Success response example:

```
{
  "result": true
}
```

If you enter a wrong email, you will receive this error message:

```
{
  "statusCode": 400,
  "message": "Email not found"
}
```

If you enter a wrong otp, you will receive this error message:

```
{
  "statusCode": 400,
  "message": "OTP is invalid"
}
```

2.3. /user/check/username

Method: POST

Purpose: This API checks whether a username already exists when a user fills out the sign-up form. It enables real-time validation and alerts the user before they submit the form.

Payload example:

```
{
  "username": "congle"
}
```

Here is the response if the username already exists:

```
{
  "result": true
}
```

Here is the response if the username is still available to sign up:

```
{
  "result": false
}
```

2.4. /user/check/email

Method: POST

Purpose: This API checks whether an email already exists when a user fills out the sign-up form. It enables real-time validation and alerts the user before they submit the form.

Payload example:

```
{
  "email": "lecongbk93@gmail.com"
}
```

Here is the response if the email already exists:

```
{
  "result": true
}
```

Here is the response if the email is still available to sign up:

```
{
  "result": false
}
```

2.5. /user/signin

Method: POST

Purpose: This API is used to log in to the system by username and password

Payload example:

```
{
  "username": "congle",
  "password": "123456"
}
```

Success response example:

```
{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…_wS0TOBkhpLR34ujvSyxiyYE4"
}
```

If the username or password is incorrect, you will receive this error message:

```
{
  "message": "Incorrect username or password!",
  "error": "Unauthorized",
  "statusCode": 401
}
```

If you try to sign in before verifying your email, you will receive this error message:

```
{
  "message": "Email has not been verified",
  "error": "Unauthorized",
  "statusCode": 401
}
```

2.6. /user/signin/email

Method: POST

Purpose: This API is used to log in to the system by email and password

Payload example:

```
{
  "email": "lecongbk93@gmail.com",
  "password": "123456"
}
```

Success response example:

```
{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…_wS0TOBkhpLR34ujvSyxiyYE4"
}
```

If the email or password is incorrect, you will receive this error message:

```
{
  "message": "Incorrect email or password!",
  "error": "Unauthorized",
  "statusCode": 401
}
```

If you try to sign in before verifying your email, you will receive this error message:

```
{
  "message": "Email has not been verified",
  "error": "Unauthorized",
  "statusCode": 401
}
```

2.7. /user/profile

Method: GET

Purpose: After a user signs in successfully, this API returns all information related to that user. An access token is required to call this API; otherwise, no data will be returned. It means each user can only see their own data

Success response example:

```
{
  "id": 2,
  "username": "congle",
  "email": "lecongbk93@gmail.com",
  "wallet_address":
"addr_test1qz8p8rp4cvmnepjhh53j3ewqzsfmu4r3lcpyqpen4cpt4jcdetjl6fr0y9jdudksm
22hx8x22q3p3q3zyxy84mlefqwslzevet",
  "mnemonic": "over muscle alone cotton chunk nature crash box noodle supply
truly twin silent night eager town quiz sweet violin system idle soup useful
canvas"
}
```

If you call this API without an access token, you will receive this error message:

```
{
  "message": "Unauthorized",
  "statusCode": 401
}
```

2.8. /user/all

Method: GET

Purpose: This API allows an admin to retrieve information about all users in the system. Only admins have this permission. An access token is required; otherwise, no data will be returned.

Success response example:

```
[
  {
    "id": 1,
    "username": "admin",
    "firstName": "admin",
    "lastName": "admin",
    "email": "congle@emurgo.africa",
    "walletAddress": null
  },
  {
    "id": 2,
    "username": "congle",
    "firstName": "Cong",
    "lastName": "Le",
    "email": "lecongbk93@gmail.com",
    "walletAddress":
"addr_test1qz8p8rp4cvmnepjhh53j3ewqzsfmu4r3lcpyqpen4cpt4jcdetjl6fr0y9jdudksm
22hx8x22q3p3q3zyxy84mlefqwslzevet"
  }
]
```

If you call this API without an access token, you will receive this error message:

```
{
  "message": "Unauthorized",
  "statusCode": 401
}
```

If you are not the admin but are trying to call this API, you will receive this error message:

```
{
  "message": "Forbidden resource",
  "error": "Forbidden",
  "statusCode": 403
}
```

2.9. /user/change-password

Method: POST

Purpose: This API is used to change a user’s password. The user must be logged in (have an access token) and provide their current password to set a new one.

Payload example:

```
{
  "currentPassword": "123456",
  "newPassword": "abcdef"
}
```

Success response example:

```
{
  "result": true
}
```

If the current password is incorrect, you will receive this error message:

```
{
  "message": "Wrong current password!",
  "error": "Not Found",
  "statusCode": 404
}
```

2.10. /user/forgot-password

Method: POST

Purpose: This API is used to request a password reset. A new OTP is generated and sent to the user’s email, which the user must provide to set a new password with another API.

Payload example:

```
{
  "email": "lecongbk93@gmail.com"
}
```

Success response example:

```
{
  "result": true
}
```

If your email does not exist, you will receive this error message:

```
{
  "message": "Email not found",
  "error": "Not Found",
  "statusCode": 404
}
```

2.11. /user/reset-password

Method: POST

Purpose: After receiving the OTP for password reset (above API), the user can use this API to set a new password

Payload example:

```
{
  "username": "congle",
  "password": "abcdef",
  "otp": "593639"
}
```

Success response example:

```
{
  "result": true
}
```

If your username does not exist, you will receive this error message:

```
{
  "message": "User not found",
  "error": "Not Found",
  "statusCode": 404
}
```

If your OTP is invalid, you will receive this error message:

```
{
  "message": "OTP is invalid",
  "error": "Not Found",
  "statusCode": 404
}
```

3. Deployment

All environment variables:

Variable	Example value	Description
NODE_ENV	dev	Defines the runtime environment (dev, staging, production)
APP_DOMAIN	http://127.0.0.1:3000	The base domain/URL where the application is hosted
APP_PORT	3000	The port on which the application server runs
TTL	60000	Time-to-live (in milliseconds) for cache or session expiration
LIMIT	1000	Maximum number of requests allowed (rate limiting)
CORS_ORIGIN	*	Allowed origin(s) for cross-origin requests (* = all origins)
DATABASE_HOST	127.0.0.1	Host address of the database server
DATABASE_PORT	5432	Port used to connect to the database (default for PostgreSQL: 5432)
DATABASE_USERNAME	postgres	Username for database authentication
DATABASE_PASSWORD	postgres	Password for database authentication
DATABASE_NAME	creative_operations	Name of the application's database
DATABASE_SYNC	1	Controls schema synchronization (1 = enabled, 0 = disabled)
LOG_FOLDER	logs	Directory path where log files are stored
ADMIN_PASSWORD	Abc123@@	Default admin password for system initialization
JWT_SECRET	Def123@@	Secret key used to sign and verify JWT tokens
JWT_EXPIRE	1h	Expiration time for JWT tokens (e.g., 1h, 30m)

ENCRYPT_KEY	Xyz123@@	Encryption key used for securing sensitive data (e.g., mnemonics)
MAIL_HOST	smtp.gmail.com	SMTP server host for sending emails
MAIL_PORT	587	Port for SMTP communication (587 for TLS, 465 for SSL).
MAIL_USER	congle@emurgo.africa	Email used for SMTP authentication. This is also the admin's email in the system.
MAIL_PASSWORD	"<YOUR_APP_MAIL_PASSWORD>"	Password or app-specific key for SMTP authentication
MAIL_FROM	noreply@gmail.com	Default sender email address used in outgoing emails
NETWORK	Preprod	Cardano network environment (e.g., Preprod, Mainnet)