How to Install and Run a social media API server

Installation Steps

1. Create a new repository on local machine or on a remote repository.
2. Open terminal and navigate to your project directory.
3. Run `npm init` to initialize a new npm project.
4. Install the required dependencies for server by running the command `npm install express mongoose`
5. Install the development dependencies for server by running the command `npm install --save-dev mocha chai dotenv nodemon` . This will install Mocha and Chai for testing, dotenv for loading environment variables, and nodemon for automatic server restarts during development.
6. If needed, please change the package.json's script part to `"scripts": { "test": "mocha", "devStart": "nodemon server.js" },`

Running the Server

To run the server, follow these steps:

1. Open your terminal and navigate to project directory.
2. Run the command `npm run devStart` or `npm start server.js` . `npm run devStart` will start the server and automatically restart it whenever there are changes to the code.
3. Visit http://localhost:3000 in the web browser to view the server homepage.

Running the Test

To run the test, follow these steps:

1. Run the command `npm test` for testing all the tests.

API Documentation

Status Code

Status Code	Description
200	Request succeed
201	Creation succeed
400	Invalid request
404	Invalid URL
500	Server error

/user

Method	Endpoint	Description
GET	/	Get all users from the database

```
GET /api/user/
```

Response:

- **Status code: 200 OK**

Body:

- **An array of user objects, each containing the user's information, such as username, id, createdAt.**

Method	Endpoint	Description
GET	/{id}	Get single user by its userId

```
GET /api/user/{id}
```

Parameters:

- id (required): The id of the user to get.

Response:

- Status code: 200 OK

Body:

- User object, each containing the user's information, such as username, id, createdAt.

Method	Endpoint	Description
POST	/create	Creates a new user

```
POST /api/user/create
```

Request Body:

- username (required): The username of the new user.

Response:

- Status code: 201 Created

Body:

- Created user object, containing the user's information, such as username, id, createdAt.

Method	Endpoint	Description
PATCH	/{id}	Updates an existing user by its id

```
PATCH /api/user/{id}
```

Request Body:

- id (required): The id of the user to update.

Response:

- Status code: 200 OK

Body:

- Updated user object, containing the user's information, such as username, id, createdAt.

Method	Endpoint	Description
DELETE	/{id}	Deletes an existing user by its id

Method	Endpoint	Description

```
DELETE /api/user/{id}
```

Request Body:

- id (required): The id of the user to delete.

Response:

- Status code: 200 OK

Body:

- JSON object with a message property indicating that the user has been deleted. If the user is not found, an error message is returned with an appropriate status code. If there is a server error, a 500 status code is returned with an error message

/post

Method	Endpoint	Description
GET	/	Get all posts from the database

```
GET /api/post/
```

Response:

- Status code: 200 OK

Body:

- An array of post objects, each containing the post's information, such as post content, post id, post's userId, targetPost which is used for reply, and createdAt.

Method	Endpoint	Description
GET	/{id}	Get single post by its postId

```
GET /api/post/{id}
```

Parameters:

- id (required): The id of the post to get.

Response:

- Status code: 200 OK

Body:

- Post object, containing the post's information, such as post content, post id, post's userId, targetPost which is used for reply, and createdAt.

Method	Endpoint	Description
GET	/{id}/replies	Get post and its related replies.

```
GET /api/post/{id}/replies
```

Parameters:

- id (required): The id of the post to get the replies.

Response:

- Status code: 200 OK

Body:

- Array of post objects, serparated to two properties. One is post property that will contain the post's information, such as post content, post id, post's userId, targetPost which is used for reply, and createdAt. Additionaly, there is replies properties of the retrived post which will include basic post information. This replies property will contain multiple objects.

Method	Endpoint	Description
POST	/create	Creates a new post

```
POST /api/post/create
```

Request Body:

- content (required): The content of the post.
- userId (required): The userId of the post.
- targetPost: This property is used when post is created as a reply

Response:

- Status code: 201 Created

Body:

- Created post object, containing post's post id, post's userId, targetPost which is used for reply, and createdAt.

Method	Endpoint	Description
PATCH	/{id}	Updates an existing post by its id

```
PATCH /api/post/{id}
```

Request Body:

- id (required): The id of the post to update.

Response:

- Status code: 200 OK

Body:

- Updated post object, containing the post's post id, post's userId, targetPost which is used for reply, and createdAt.

Method	Endpoint	Description
DELETE	/{id}	Deletes an existing post by its id

```
DELETE /api/post/{id}
```

Request Body:

- id (required): The id of the post to delete.

Response:

- Status code: 200 OK

Body:

- JSON object with a message property indicating that the post has been deleted. If the post is not found, an error message is returned with an appropriate status code. If there is a server error, a 500 status code is returned with an error message

/like

Method	Endpoint	Description
GET	/	Get all likes from the database

```
GET /api/like/
```

Response:

- Status code: 200 OK

Body:

- An array of like objects, each containing the userId and targetPost.

Method	Endpoint	Description
GET	/{id}/likes	Get all likes of a specific post by ID

```
POST /api/like/{id}/likes
```

Request Body:

- id (required): The id of the post to retrive all likes.

Response:

- Status code: 200 OK

Body:

- Array of like objects, each containing the userId and targetPost.

Method	Endpoint	Description
POST	/{id}	Like a post with a given post's ID

```
POST /api/like/{id}
```

Request Body:

- id: The id of the post to like.

Response:

- Status code: 201 Created

Body:

- Created like object, containing the userId and targetPost.

Method	Endpoint	Description
DELETE	/{id}	Unlike a post with a given post's ID

```
DELETE /api/like/{id}
```

Request Body:

- id (required): The id of the post to unlike.

Response:

- Status code: 200 OK

Body:

- JSON object with a message property indicating that the like has been deleted(unlike). If the post is not found, an error message is returned with an appropriate status code. If there is a server error, a 500 status code is returned with an error message