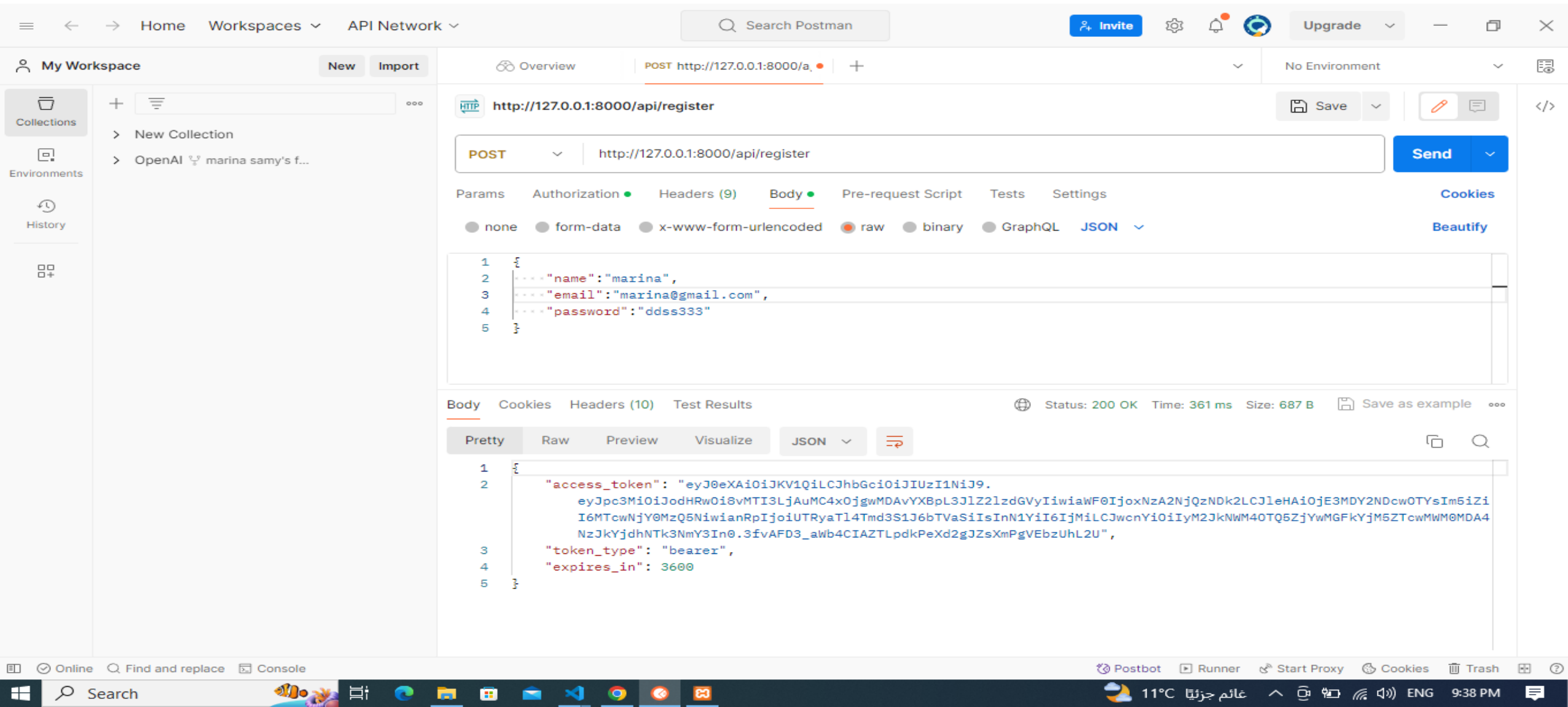


# Instruction

- 1- Createing new Laravel application
- 2- Install the third party package called jwt-auth
- 3- Generate a jwt secret key by running  
php artisan jwt:secret
- 4- Define two methods to return the JWTIdentifier and JWTCustomClaims in user model
- 5- Setting the api guard as the default and telling the api guard to use jwt
- 6- Set up the Database connection in env file which name is noteDB
- 7- Make migration, models and controller
- 8- Test in postman

# Register new user



# Validation error when register

The screenshot shows the Postman interface with a REST client request configured for a POST method to the endpoint `http://127.0.0.1:8000/api/register`. The request body is a JSON object:

```
1 {
2   "name": "marina",
3   "email": "note2",
4   "password": "ddss333"
5 }
```

The response status is **400 Bad Request** with a time of 609 ms and a size of 381 B. The response body, shown in JSON format, contains an error message:

```
1 {
2   "error": {
3     "email": [
4       "The email must be a valid email address."
5     ]
6   }
7 }
```

The left sidebar shows the 'My Workspace' section with a collection named 'OpenAI' and a sub-collection 'marina samy's f...'. The bottom status bar includes various utility icons and the system clock showing 9:38 PM.

# Unauthorized error

The screenshot shows the Postman interface with a REST client request configured for a POST method to the endpoint `http://127.0.0.1:8000/api/notes/7/attach-file`. The request body is set to 'form-data' and includes a file named 'dashboard (6).sql'. The response status is '400 Bad Request' with a message indicating that the file must be of a supported type (jpg, jpeg, png, webp, pdf, doc.).

**Request Details:**

- Method: POST
- URL: `http://127.0.0.1:8000/api/notes/7/attach-file`
- Body Type: form-data
- File: dashboard (6).sql

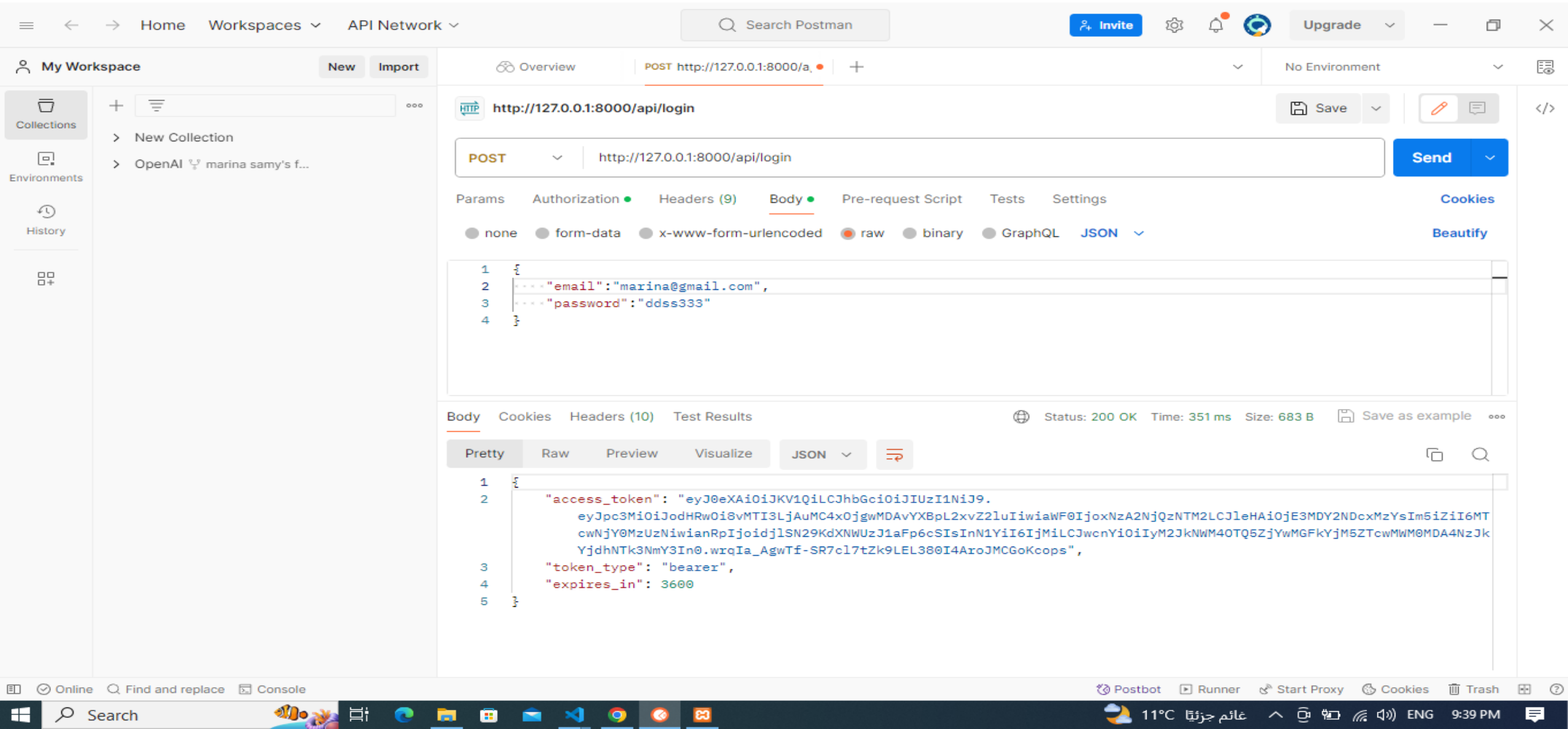
**Response Details:**

- Status: 400 Bad Request
- Time: 490 ms
- Size: 404 B

**Response Body (JSON):**

```
1 {
2   "error": {
3     "file": [
4       "The file must be a file of type: jpg, jpeg, png, webp, pdf, doc."
5     ]
6   }
7 }
```

# User login



# Authorized token

The screenshot displays the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'My Workspace' with options to 'New' or 'Import' collections and environments. The main panel is set to 'Overview' for a POST request to 'http://127.0.0.1:8000/api/login'. The 'Authorization' tab is active, showing a 'Bearer Token' type. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' The 'Token' field contains a long alphanumeric string. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response status is '200 OK' with a time of '351 ms' and a size of '683 B'. The JSON body is as follows:

```
1 {
2   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vMTI3LjAuMC4xOjgwMDAvYXBpL2xvZ2luIiwiaWF0IjoxNzA2NjQzNTM2LCJleHAiOiJlMDY2NDcxMzYsIm5iZiI6MTcwNjY0MzUzNiwiianRpIjojdj1SN29KdXNWUzJ1aFp6cSIsInN1YiI6IjMiLCJwcnYiOiIyM2JkNW40OTQ5ZjYwMGFkYjM5ZTcwMWM0MDA4NzJkYjdhNTk3NmY3In0.wrqIa_AgwTf-SR7cl7tZk9LEL380I4AroJMCGoKcops",
3   "token_type": "bearer",
4   "expires_in": 3600
5 }
```

The bottom status bar shows 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Trash', and system information: '11°C', 'ENG', and '9:39 PM'.

# Get auth user data

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/api/user` using Bearer Token authorization. The response is a JSON object containing user data.

**Request Details:**

- Method: GET
- URL: `http://127.0.0.1:8000/api/user`
- Authorization: Bearer Token
- Token: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...`

**Response Details:**

- Status: 200 OK
- Time: 377 ms
- Size: 473 B

**Response Body (JSON):**

```
1 {
2   "id": 7,
3   "name": "Marina Samy",
4   "email": "test@gmail.com",
5   "email_verified_at": null,
6   "created_at": "2024-01-30T22:22:09.000000Z",
7   "updated_at": "2024-01-30T22:22:09.000000Z"
8 }
```

# Create new note

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'My Workspace' with options to 'New' or 'Import' collections and environments. The main panel is set to 'Overview' for the endpoint `POST http://127.0.0.1:8000/api/notes`. The request body is a JSON object:

```
1 {
2   "title": "title of the note",
3   "content": "note content here"
4 }
```

The response is shown in the 'Body' tab, indicating a status of 201 Created. The response body is a JSON object:

```
1 {
2   "message": "Note created successfully",
3   "note": {
4     "title": "title of the note",
5     "content": "note content here",
6     "user_id": 3,
7     "updated_at": "2024-01-30T19:40:02.000000Z",
8     "created_at": "2024-01-30T19:40:02.000000Z",
9     "id": 3
10  }
11 }
```

The bottom status bar shows 'Online', 'Find and replace', 'Console', and system information like '11°C' and '9:40 PM'.



# Get all notes of user

The screenshot displays the Postman interface with a GET request to `http://127.0.0.1:8000/api/notes` successfully executed. The response is a JSON array of notes for the user 'marina'.

**Request Details:**

- Method: GET
- URL: `http://127.0.0.1:8000/api/notes`
- Body: Empty (selected)
- Headers (9): none, form-data, x-www-form-urlencoded, raw, binary, GraphQL, JSON (selected)

**Response Details:**

- Status: 200 OK
- Time: 300 ms
- Size: 966 B
- Save as example

**Response Body (JSON):**

```
1 {
2   "notes": [
3     {
4       "id": 3,
5       "user_id": 3,
6       "title": "title of the note",
7       "content": "note content here",
8       "created_at": "2024-01-30T19:40:02.000000Z",
9       "updated_at": "2024-01-30T19:40:02.000000Z",
10      "user": {
11        "id": 3,
12        "name": "marina",
13        "email": "marina@gmail.com",
14        "email_verified_at": null,
15        "created_at": "2024-01-30T19:38:16.000000Z",
16        "updated_at": "2024-01-30T19:38:16.000000Z"
17      }
18    },
19  ]
20 }
```

# Get remain notes of user

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/api/notes`. The response is a JSON array of notes. The first note is selected, showing its details in the 'Body' tab. The response status is 200 OK, with a time of 300 ms and a size of 966 B.

```
1 [
2   {
3     "id": 4,
4     "user_id": 3,
5     "title": "title3",
6     "content": "note3",
7     "created_at": "2024-01-30T19:42:57.000000Z",
8     "updated_at": "2024-01-30T19:42:57.000000Z",
9     "user": {
10      "id": 3,
11      "name": "marina",
12      "email": "marina@gmail.com",
13      "email_verified_at": null,
14      "created_at": "2024-01-30T19:38:16.000000Z",
15      "updated_at": "2024-01-30T19:38:16.000000Z"
16    }
17  }
18 ]
```

# Get specific note by id

The screenshot displays the Postman interface for an API request. The top bar shows navigation links (Home, Workspaces, API Network), a search bar (Search Postman), and utility buttons (Invite, Upgrade). The left sidebar contains 'My Workspace' with 'New' and 'Import' buttons, and a list of collections including 'New Collection' and 'OpenAI marina samy's f...'. The main panel shows a GET request to 'http://127.0.0.1:8000/api/notes/3'. The 'Body' tab is selected, showing a JSON response. The response status is '200 OK' with a time of '265 ms' and size of '651 B'. The JSON body is formatted as follows:

```
1 {
2   "note": {
3     "id": 3,
4     "user_id": 3,
5     "title": "title of the note",
6     "content": "note content here",
7     "created_at": "2024-01-30T19:40:02.000000Z",
8     "updated_at": "2024-01-30T19:40:02.000000Z",
9     "user": {
10      "id": 3,
11      "name": "marina",
12      "email": "marina@gmail.com",
13      "email_verified_at": null,
14      "created_at": "2024-01-30T19:38:16.000000Z",
15      "updated_at": "2024-01-30T19:38:16.000000Z"
16    }
17  }
18 }
```

The bottom status bar includes 'Online', 'Find and replace', 'Console', and a system tray with 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Trash', and a temperature display of '11°C'.

# Error handling

The screenshot displays the Postman interface with a REST client request configured for a GET method to the URL `http://127.0.0.1:8000/api/notes/3`. The request is part of a collection named "Note-App / Note CRUD" and is specifically labeled "handel error". The "Authorization" tab is active, showing a Bearer Token authorization type with a token value of `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...`. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)."

The response is displayed in the "Body" tab, showing a JSON object with an error message: `{ "error": "Note not found" }`. The status bar indicates a 404 Not Found status, a response time of 332 ms, and a size of 341 B. The response is formatted as JSON.

The left sidebar shows the "My Workspace" section with a list of collections and environments. The "Note CRUD" collection is expanded, showing various API endpoints. The "History" section is also visible.

The bottom status bar includes the Postman logo, "Online" status, "Find and replace" and "Console" buttons, and a system tray with the temperature (11°C), time (2:02 AM), and language (ENG).

# Update note by id

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'My Workspace' with options for 'New' and 'Import'. The main area is divided into two tabs: 'Overview' and 'PUT http://127.0.0.1:8000/ap'. The 'Overview' tab is active, showing the request details for the endpoint `http://127.0.0.1:8000/api/notes/3`. The request method is 'PUT'. The request body is in JSON format, containing the following data:

```
1 {
2   "title": "updated title",
3   "content": "updated content"
4 }
```

The response is displayed in the 'Body' tab, showing a JSON object with the following data:

```
1 {
2   "message": "Note updated successfully",
3   "note": {
4     "id": 3,
5     "user_id": 3,
6     "title": "updated title",
7     "content": "updated content",
8     "created_at": "2024-01-30T19:40:02.000000Z",
9     "updated_at": "2024-01-30T19:48:48.000000Z",
10    "user": {
11      "id": 3,
12      "name": "marina",
13      "email": "marina@gmail.com",
14      "email_verified_at": null,
15      "created_at": "2024-01-30T19:38:16.000000Z",
```

The status bar at the bottom indicates the request was successful with a status of 200 OK, a time of 427 ms, and a size of 683 B. The bottom of the screen shows the Windows taskbar with various application icons and system information.

# Delete note by id

The screenshot shows the Postman application interface. The top bar includes navigation icons, a search bar labeled "Search Postman", and buttons for "Invite", "Upgrade", and window controls. The left sidebar shows "My Workspace" with "New" and "Import" buttons, and a list of collections and environments. The main panel displays a DELETE request to the URL `http://127.0.0.1:8000/api/notes/4`. The request body is in JSON format, containing `"title": "updated-title"` and `"content": "updated-content"`. The response is shown in the "Body" tab, displaying a JSON object with `"message": "Note deleted successfully"`. The status bar at the bottom indicates the request was successful with a 200 OK status, 315 ms response time, and 347 B response size.

Home Workspaces API Network

Search Postman

Invite Upgrade

My Workspace New Import

Collections

Environments

History

Overview DEL `http://127.0.0.1:8000/api/notes/4`

`http://127.0.0.1:8000/api/notes/4` Save Send

DELETE `http://127.0.0.1:8000/api/notes/4`

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "title": "updated-title",
3   "content": "updated-content"
4 }
```

Body Cookies Headers (10) Test Results Status: 200 OK Time: 315 ms Size: 347 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Note deleted successfully"
3 }
```

Online Find and replace Console

Postbot Runner Start Proxy Cookies Trash

11°C غائم جزئيًا ENG 9:49 PM

# Upload attach for note

The screenshot displays the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'My Workspace' with options to 'New' or 'Import' collections and environments. The main panel is set to 'Overview' for a POST request to `http://127.0.0.1:8000/api/notes/3/attach-file`. The request body is configured as 'form-data' with a single entry: 'file' (checked) with a file selection icon and 'Type' (unchecked) with a text selection icon. The response is shown in the 'Body' tab, indicating a successful status of 200 OK. The response body is formatted as JSON, showing a successful upload message and the attachment path.

Home Workspaces API Network Search Postman Invite Upgrade

My Workspace New Import Overview POST http://127.0.0.1:8000/a. + No Environment

HTTP http://127.0.0.1:8000/api/notes/3/attach-file Save Send

POST http://127.0.0.1:8000/api/notes/3/attach-file

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> file	File 360_F_617132669_YptvM7flucz...		
<input type="checkbox"/> Type	Text File		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 317 ms Size: 429 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "attachments/wTBt4s1UG3PwA8H8LZC7e1LKg6DzW0Q3I9DKqhwc.jpg",
3   "message": "File attached successfully to the note"
4 }
```

Postbot Runner Start Proxy Cookies Trash ENG 9:51 PM

# Validation error attachment

The screenshot shows the Postman interface with a REST client request configured for a POST method to the endpoint `http://127.0.0.1:8000/api/notes/7/attach-file`. The request is part of a collection named "Note-App" under the "Note CRUD" folder. The response tab is selected, showing a "400 Bad Request" status code. The response body is displayed in JSON format, indicating a validation error for the file upload.

**Request Details:**

- Method: POST
- URL: `http://127.0.0.1:8000/api/notes/7/attach-file`
- Params: None
- Headers: None
- Body: None

**Response Details:**

- Status Code: 400 Bad Request
- Body (JSON):

```
1 {
2   "error": {
3     "file": [
4       "The file must be a file of type: jpg, jpeg, png, webp, pdf, doc."
5     ]
6   }
7 }
```

**Left Sidebar (Collections):**

- My Workspace
  - GET logout user
    - successful logout response
  - Note CRUD
    - GET create new note
      - successful create response
    - GET get all notes
      - successful get response
    - GET get note by id
      - successful response
    - GET handel error
      - handel error response
    - GET update note
      - successful update response
    - GET upload attachment
      - successful upload response
    - GET validation error
      - error response
    - GET delete attachment
      - successful deletion response
    - GET delete note
      - successful deletion



# Delete specific file by id

The screenshot shows the Postman interface with a DELETE request configured. The URL is `http://127.0.0.1:8000/api/attach-file/3`. The request is set to `DELETE` and the body is configured with `form-data`. A file named `360_F_617132669_YptvM7flucz...` is attached to the request. The response is a JSON object: `{\"message\": \"file deleted successfully\"}`.

**Request Details:**

- Method: `DELETE`
- URL: `http://127.0.0.1:8000/api/attach-file/3`
- Body Type: `form-data`
- File Name: `360_F_617132669_YptvM7flucz...`

**Response Details:**

- Status: `200 OK`
- Time: `285 ms`
- Size: `347 B`
- Body: `{\"message\": \"file deleted successfully\"}`

# User logout

The screenshot displays the Postman application interface. At the top, the navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar labeled 'Search Postman' is present. On the left sidebar, 'My Workspace' is selected, showing a 'Collections' list with 'New Collection' and 'OpenAI'. The main workspace shows a 'POST' request to 'http://127.0.0.1:8000/api/logout'. The 'Body' tab is active, showing a JSON response: 

```
{  "message": "Successfully logged out"}
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 282 ms', and 'Size: 345 B'. The Windows taskbar at the very bottom shows the system clock at 9:57 PM and temperature at 11°C.

Home Workspaces API Network Search Postman Invite Upgrade

My Workspace New Import Overview POST http://127.0.0.1:8000/a. + No Environment

HTTP http://127.0.0.1:8000/api/logout Save Send

POST http://127.0.0.1:8000/api/logout

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 5

Body Cookies Headers (10) Test Results Status: 200 OK Time: 282 ms Size: 345 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Successfully logged out"
3 }
```

# Database Schema

localhost/phpmyadmin/index.php?route=/database/designer&db=notedb

phpMyAdmin

Recent Favorites

- New
- dashboard
- information\_schema
- laravel
- mysql
- notedb
  - New
  - attachments
  - failed\_jobs
  - migrations
  - notes
  - password\_resets
  - personal\_access\_tokens
  - users
- performance\_schema
- phpmyadmin
- test

Server: 127.0.0.1 Database: notedb

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers

**notedb users**

- id : bigint(20) unsigned
- name : varchar(255)
- email : varchar(255)
- email\_verified\_at : timestamp
- password : varchar(255)
- remember\_token : varchar(100)
- created\_at : timestamp
- updated\_at : timestamp

**notedb notes**

- id : bigint(20) unsigned
- user\_id : bigint(20) unsigned
- title : varchar(255)
- content : text
- created\_at : timestamp
- updated\_at : timestamp

**notedb attachments**

- id : int(10) unsigned
- note\_id : bigint(20) unsigned
- file\_path : varchar(255)
- created\_at : timestamp
- updated\_at : timestamp

notedb.pdf 156 KB • Done

11°C غائم جزئيًا ENG 1:30 AM

# Database Relations

Creating 3 tables

1- user table

2- notes table

3- attachments table

User table has relation one to many with notes table

Notes table has relation one to many with attachments table

Making new table for attachments to allow user to upload more than one file for note not only one and replace it in column notes table