



BITS Pilani
Pilani Campus

Assignment – 1

Full Stack Application Development

Submitted by

Monita(2023TM93771)

Technologies Used are

1. PostgreSQL
2. Node.js
3. Express.js
4. REST
5. Sequelize (ORM)

1. PostgreSQL (Postgres)

Why?: PostgreSQL is a powerful, open-source relational database management system (RDBMS) that supports advanced features like complex queries, foreign keys, and transactional integrity. It's ideal for applications requiring structured data storage, scalability, and data consistency.

Benefits:

- **ACID Compliance:** Ensures reliable transactions.
- **Support for Advanced Data Types:** Such as JSON, which can be useful for modern web apps.
- **Scalability:** Great for growing applications.

2. Node.js

Why?: Node.js is a JavaScript runtime that allows you to build server-side applications using JavaScript. It's non-blocking and asynchronous, making it suitable for handling multiple requests efficiently.

Benefits:

- **Asynchronous I/O:** Handles multiple requests concurrently, which improves performance for I/O-heavy apps.
- **Single Language:** You can use JavaScript on both the frontend (React) and backend (Node.js), making it easier to share code between layers.
- **Vibrant Ecosystem:** A large number of npm libraries and modules are available.

3. Express.js

Why?: Express is a minimal, flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It helps in routing, handling requests, and creating APIs quickly.

Benefits:

- **Middleware:** Allows for easy inclusion of third-party or custom middleware for handling requests and responses.
- **Simplified Routing:** Manages routing efficiently for RESTful API development.
- **Lightweight:** Provides a minimal structure, leaving you free to choose additional libraries as needed.

4. REST (Representational State Transfer)

Why?: REST is a widely-used architectural style for designing networked applications. It uses standard HTTP methods (GET, POST, PUT, DELETE) for CRUD operations, making it easy to interact with APIs over the web.

Benefits:

- **Scalability:** Stateless interactions ensure that each request is independent, making it easier to scale the application.
- **Interoperability:** REST APIs can be consumed by different clients (mobile apps, web browsers) using standard protocols.
- **Simplicity:** REST uses simple HTTP, which is easy to understand and integrate.

5. Sequelize (ORM)

Why?: Sequelize is an Object-Relational Mapping (ORM) library for Node.js that simplifies database interaction by allowing you to use JavaScript objects to perform SQL queries. It abstracts away the complexity of raw SQL, allowing developers to interact with the database in a more intuitive way.

Benefits:

- **Simplified CRUD Operations:** Provides simple, high-level methods to handle database queries.
- **Database-agnostic:** Though you're using PostgreSQL, Sequelize supports multiple databases (MySQL, SQLite, etc.), allowing for flexibility in the future.
- **Model Synchronization:** Automatically syncs models (JavaScript objects) with the database, helping maintain structure.
- **Associations:** Manages relationships between different models (e.g., User and Books) seamlessly.

Below is the home page

- We have two button.
- First button to Login.
- Second button to Register.
- After Register it'll take us to Login page and after successfully login it'll take us to profile page.
- Worked on User registration and Authentication, Book listing and Book search.



Welcome

Discover, exchange, and share your favorite books with others. Whether you're looking to borrow, lend, or exchange books, our platform makes it easy and convenient.

"A room without books is like a body without a soul." —
Marcus Tullius Cicero

Login

Register

© 2024 Book Exchange Platform

Book Exchange Platform

Register

new1

new1@gmail.com

.....

Register

Here data got saved in our database

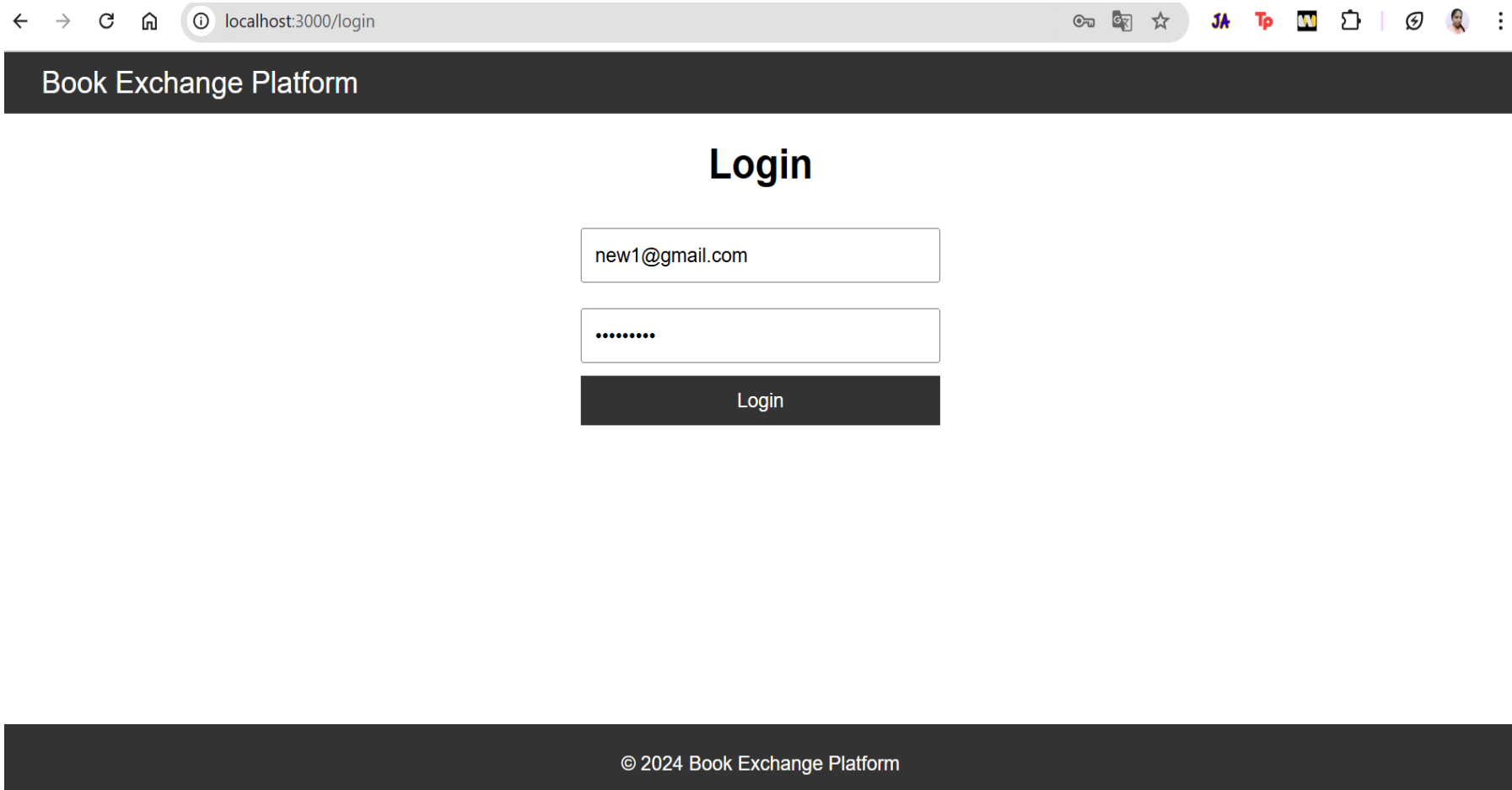
```
Backend > src > models > JS UserProfile.js > [0] UserProfile
1  const { DataTypes } = require('sequelize');
2  const sequelize = require('../database/sequelize');
3  const User = require('../User');
4
5  const UserProfile = sequelize.define('UserProfile', {
6    favorite_genres: {
7      type: DataTypes.TEXT,
8    },
9    reading_preferences: {
10     type: DataTypes.TEXT,
11   },
12 });
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
2024-11-17 01:05:28.258+05:30 | 2024-11-17 01:05:28.258+05:30
7 | test6 | tes6@gmail.com | $2a$10$Kjf2V8Pv05nWjBkBuAMNtOixSeF8ZjuEvCvmPOJH1pivcQn1JNU90 |
2024-11-17 11:49:19.418+05:30 | 2024-11-17 11:49:19.418+05:30
8 | monita123test | mon123@example.com | $2a$10$i12cJQd/ydgkIvNZoJysD.ozj2uciXjvrpUSwprFqLyqGIidErLzW |
2024-11-17 13:22:11.892+05:30 | 2024-11-17 13:22:11.892+05:30
12 | test10 | test10@gmail.com | $2a$10$0teOAVs4OQGCeT.pR1/1UuBPWm.BpQcLjbB7cfjfXpoozJLL9Nge |
2024-11-17 15:00:49.252+05:30 | 2024-11-17 15:00:49.252+05:30
13 | test20 | test20@gmail.com | $2a$10$6tANH7JP5Yp1E9PDXQx5nuOvo1ZMPVbc.4tKIhskWZ3rFLWR7Y1EG |
2024-11-17 15:12:03.529+05:30 | 2024-11-17 15:12:03.529+05:30
14 | test11 | test11@gmail.com | $2a$10$WKIXcIqWN2RFnvwCx1LpX.bWpy1CNqrau6UIBjbvFLe8n4e1.HINK |
2024-11-17 15:21:58.45+05:30 | 2024-11-17 15:21:58.45+05:30
15 | monita123test1 | mon123@example.com | $2a$10$EUK1QbCU_CP2BekD3ut_XeITnggB6EdGDS1_6tSLbol8TuiiG6C8e6 |
2024-11-17 15:42:04.516+05:30 | 2024-11-17 15:42:04.516+05:30
16 | new1 | new1@gmail.com | $2a$10$NYaV9j8zXuhsBN6yoRijPuKxHTD1icerGteGVq01z.d25brVTv1z0 |
2024-11-17 20:04:24.649+05:30 | 2024-11-17 20:04:24.649+05:30
(13 rows)
```

-- More --

After registering we are re-directed to Login page



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/login'. The browser's toolbar includes navigation icons (back, forward, refresh, home), a search bar, and various extension icons (JA, Tp, W, and others). The page has a dark header bar with the text 'Book Exchange Platform'. Below the header, the word 'Login' is centered in a large, bold font. Underneath, there are two input fields: the first contains the email address 'new1@gmail.com', and the second contains a masked password represented by eight dots. A dark 'Login' button is positioned below the password field. At the bottom of the page, a dark footer bar contains the copyright notice '© 2024 Book Exchange Platform'.

Book Exchange Platform

Login

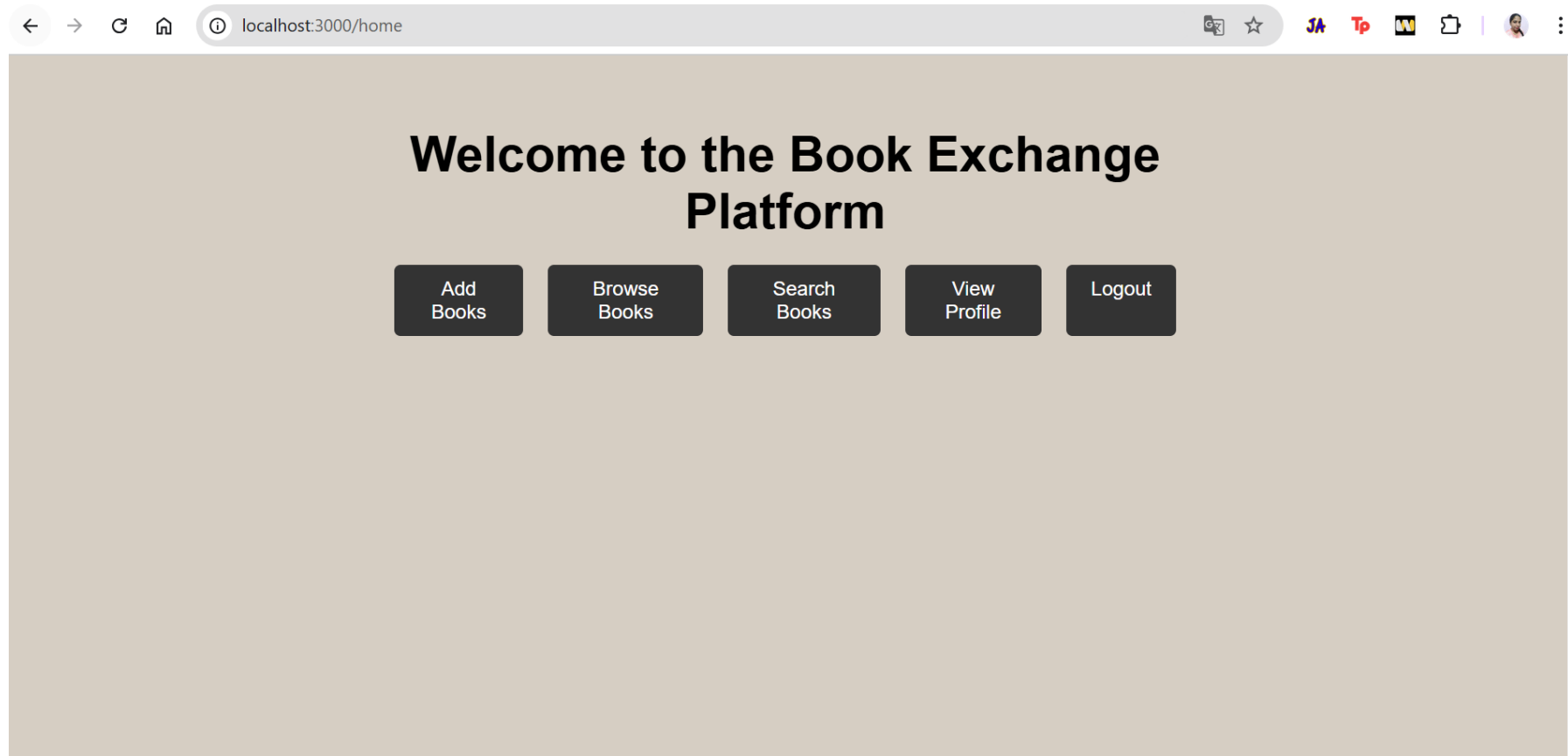
new1@gmail.com

.....

Login

© 2024 Book Exchange Platform

After login we are re-directed to main home page



From here we go to user profile, add book, book search, book list and logout

localhost:3000/profile

Book Exchange Platform

User Profile

Favorite Genres

Reading Preferences

Books Owned

Books Wanted

Update Profile

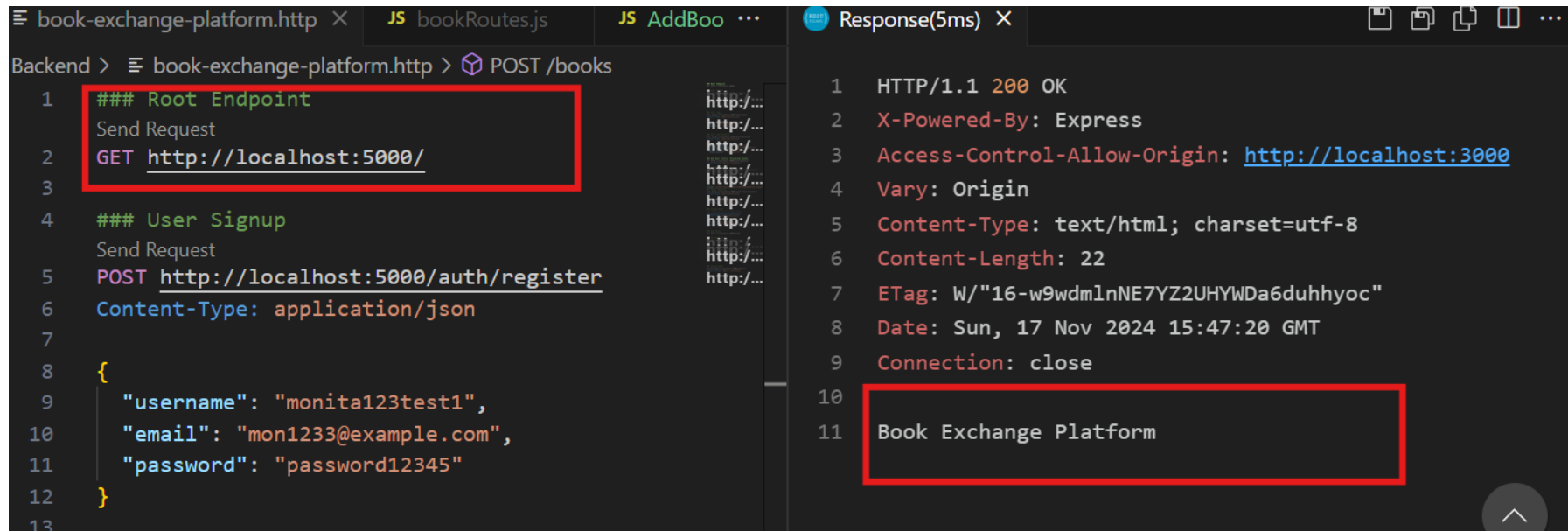
© 2024 Book Exchange Platform

Book Exchange Platform

Add Book

Add

Logout will re-direct to starting home page. Below are the API response



The screenshot displays a web browser's developer tools interface. The left pane shows a REST client with two requests. The first request, 'GET http://localhost:5000/', is highlighted with a red box. The second request is a 'POST' to 'http://localhost:5000/auth/register' with a JSON body containing user details. The right pane shows the response for the first request, which is a 200 OK status with various headers and a body containing the text 'Book Exchange Platform', also highlighted with a red box.

```
book-exchange-platform.http X JS bookRoutes.js JS AddBoo ... Response(5ms) X
Backend > book-exchange-platform.http > POST /books
1 ### Root Endpoint
  Send Request
2 GET http://localhost:5000/
3
4 ### User Signup
  Send Request
5 POST http://localhost:5000/auth/register
6 Content-Type: application/json
7
8 {
9   "username": "monita123test1",
10  "email": "mon1233@example.com",
11  "password": "password12345"
12 }
13

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: http://localhost:3000
4 Vary: Origin
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 22
7 ETag: W/"16-w9wdm1nNE7YZ2UHYWDa6duhhyoc"
8 Date: Sun, 17 Nov 2024 15:47:20 GMT
9 Connection: close
10
11 Book Exchange Platform
```


Register the user api

```
### User Signup
Send Request
POST http://localhost:5000/auth/register
Content-Type: application/json

{
  "username": "monita123test1",
  "email": "mon1233@example.com",
  "password": "password12345"
}

http://...
http://...
:::
http://...
http://...

5 Content-Type: application/json; charset=utf-8
6 Content-Length: 33
7 ETag: W/"21-iGErLku/9taeeOQN+9qwFgLXnuo"
8 Date: Sun, 17 Nov 2024 15:48:10 GMT
9 Connection: close
10
11 {
12   "message": "User already exists"
13 }
```

User login api

```
### User Login
Send Request
POST http://localhost:5000/auth/login
Content-Type: application/json

{
  "email": "mon1@example.com",
  "password": "password1234"
}

### post user profile
Send Request
POST http://localhost:5000/users/profile

Etag: W/"e0-00VREVNKaEjZK5XdLB3M43WIM+U"
8 Date: Sun, 17 Nov 2024 15:49:11 GMT
9 Connection: close
10
11 {
12   "message": "User login successfully",
13   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJ0b24xQGV4YW1wbGUuY29tIiwiaWF0IjoxNzU0ODU4NTUxYkYk-eTDv2pkNoowLyhPh-DiPGONgmzvpQ8t5oS5Qx_E",
14   "redirectUrl": "/profile"
15 }
```

Password reset mail

```
### Forgot Password
Send Request
POST http://localhost:5000/auth/forgot-password
Content-Type: application/json

{
  "email": "mon1233@example.com"
}

###
5 references

http://.../ 4 Vary: Origin
http://.../ 5 Content-Type: application/json; charset=utf-8
http://.../ 6 Content-Length: 39
7 ETag: W/"27-pe5pEgQALXk2w2XfiSECwu6etKQ"
8 Date: Sun, 17 Nov 2024 15:51:38 GMT
9 Connection: close
10
11 ✓ {
12   "message": "Password reset email sent"
13 }
```

Password Reset to: <mon1233@example.com> a minute ago
Password Reset to: <john@example.com> a day ago
Password Reset to: <john@example.com> a day ago
Password Reset to: <john@example.com> 16 days ago
Password Reset to: <john@example.com> 16 days ago
Password Reset to: <john@example.com> 16 days ago
Password Reset to: <john@example.com> 16 days ago

From: <your_email@example.com> 2024-11-17 15:51, 561 Bytes
To: <mon1233@example.com>
[Show Headers](#)

HTML HTML Source Text **Raw** Spam Analysis Tech Info

From: your_email@example.com
To: mon1233@example.com
Subject: Password Reset
Message-ID: <61e1c8bc-82ec-f977-59d6-f8b62f9d0741@example.com>
Content-Transfer-Encoding: quoted-printable
Date: Sun, 17 Nov 2024 15:51:35 +0000
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8

Click the following link to reset your password: http://localhost:5000/reset-password?token=3DeyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTUzImVtYWlsIjoibW9uMTIzM0BlcGFtcGx1LmNvbSI6Im1hdCI6MTczMTg1ODY5NSw=iZXhwIjoxNzMyODYyMjk1fQ.hIhXoxtrV8YQ-7pRvmHnViqHCsOFbg4_A2SPY2VwpAw

Reset password for current session

The image shows a VS Code editor with two REST Client requests and their corresponding JSON responses.

Request 1: Reset Password

```
### Reset Password
Send Request
POST http://localhost:5000/auth/reset-password
Content-Type: application/json

{
  "token": "{{token}}",
  "newPassword": "newpassword123"
}
```

Response 1:

```
{
  "message": "Password reset successfully",
  "user": {
    "id": 1,
    "username": "monita",
    "email": "mon1@example.com",
    "password": "$2a$10$tv5Jg4YMPCyRInf97qAb4009RW05glMuIzigQ1DN9yLZUTuJE/MOO",
    "createdAt": "2024-11-16T13:31:58.135Z",
    "updatedAt": "2024-11-17T15:54:10.367Z"
  }
}
```

Request 2: Create Book

```
### Create Book
Send Request
POST http://localhost:5000/books
```

Response 2:

```
{
  "message": "Book created successfully",
  "book": {
    "id": 1,
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "year": 1925,
    "status": "available"
  }
}
```

Added book to database

```
### Create Book
Send Request
POST http://localhost:5000/books
Authorization: Bearer {{token}}
Content-Type: application/json

{
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald"
}

11 {
12   "message": "Book created successfully",
13   "book": {
14     "id": 5,
15     "title": "The Great Gatsby",
16     "author": "F. Scott Fitzgerald",
17     "user_id": 1,
18     "updatedAt": "2024-11-17T15:54:53.881Z",
19     "createdAt": "2024-11-17T15:54:53.881Z"
20   }
```

```
book_exchange_db=# SELECT * FROM "Books";
 id | title | author | createdAt | updatedAt | u
ser_id
-----+-----+-----+-----+-----+--
 2 | The Great Gatsby | F. Scott Fitzgerald | 2024-11-17 20:35:36.432+05:30 | 2024-11-17 20:35:36.432+05:30 | 1
 3 | The Great Gatsby | F. Scott Fitzgerald | 2024-11-17 21:06:10.625+05:30 | 2024-11-17 21:06:10.625+05:30 | 1
 4 | The Great Gatsby | F. Scott Fitzgerald | 2024-11-17 21:11:17.927+05:30 | 2024-11-17 21:11:17.927+05:30 | 1
 5 | The Great Gatsby | F. Scott Fitzgerald | 2024-11-17 21:24:53.881+05:30 | 2024-11-17 21:24:53.881+05:30 | 1
(4 rows)

-- More --
```

Get all books

```
"author": "F. Scott Fitzgerald"
```

Get Books

Send Request

```
GET http://localhost:5000/books
```

Get Book by ID

Send Request

```
GET http://localhost:5000/books/1
```

Update Book

Send Request

```
PUT http://localhost:5000/books/1
```

```
Authorization: Bearer {{token}}
```

```
Content-Type: application/json
```

```
{
  "title": "The Great Gatsby (Updated)",
```

[illegible]

```
11  ∨ {
12  ∨   "books": [
13  ∨     {
14  ∨       "id": 2,
15  ∨       "title": "The Great Gatsby",
16  ∨       "author": "F. Scott Fitzgerald",
17  ∨       "createdAt": "2024-11-17T15:05:36.432Z",
18  ∨       "updatedAt": "2024-11-17T15:05:36.432Z",
19  ∨       "user_id": 1,
20  ∨       "User": {
21  ∨         "id": 1,
22  ∨         "username": "monita",
23  ∨         "email": "mon1@example.com",
24  ∨         "password": "$2a$10$tv5Jg4YMPCyRInf97q
Ab4009RW05glMuIzigQ1DN9ylZUTuJE/M00",
25  ∨         "createdAt": "2024-11-16T13:31:58.123
Z",
```

```
69 GET http://localhost:5000/books
```

70

```
71     ### Get Book by ID
```

Send Request

```
72 GET http://localhost:5000/books/1
```

73

```
74     ### Update Book
```

Send Request

```
75 PUT http://localhost:5000/books/1
```

```
76 Authorization: Bearer {{token}}
```

```
77 Content-Type: application/json
```

78

79 {

```
80     "title": "The Great Gatsby (Updated)",
```

68

```
70     "username": "monita",
```

```
71 "email": "mon1@example.com",
```

```
72 "password": "$2a$10$tv5Jg4YMPCyRInf97q
```

Ab4009RW05g1MuIzigQ1DN9y1ZUTuJE/M00",

```
73      "createdAt": "2024-11-16T13:31:58.135"
```

```
74      "updatedAt": "2024-11-17T15:54:10.367"
```

 z''

75

Book by Id

```
id 7 - Book exchange platform.http GET /books/5
Send Request
GET http://localhost:5000/books

### Get Book by ID
Send Request
GET http://localhost:5000/books/5

### Update Book
Send Request
PUT http://localhost:5000/books/1
Authorization: Bearer {{token}}
Content-Type: application/json

{
  "title": "The Great Gatsby (Updated)",
  "author": "F. Scott Fitzgerald"
}

### Delete Book
```

```
11 {
12   "book": {
13     "id": 5,
14     "title": "The Great Gatsby",
15     "author": "F. Scott Fitzgerald",
16     "createdAt": "2024-11-17T15:54:53.881Z",
17     "updatedAt": "2024-11-17T15:54:53.881Z",
18     "user_id": 1,
19   "User": {
20     "id": 1,
21     "username": "monita",
22     "email": "mon1@example.com",
23     "password": "$2a$10$tv5Jg4YMPCyRInf97qAb
4009RW05glMuIzigQ1DN9ylZUTuJE/M00",
24     "createdAt": "2024-11-16T13:31:58.135Z",
25     "updatedAt": "2024-11-17T15:54:10.3
26   }
```

Updated the book by id

```
### Update Book
Send Request
PUT http://localhost:5000/books/5
Authorization: Bearer {{token}}
Content-Type: application/json

{
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald new"
}

### Delete Book
Send Request
DELETE http://localhost:5000/books/1
Authorization: Bearer {{token}}
```

```
9 Connection: close
10
11 ✓ {
12   "message": "Book updated successfully",
13   "book": {
14     "id": 5,
15     "title": "The Great Gatsby",
16     "author": "F. Scott Fitzgerald new",
17     "createdAt": "2024-11-17T15:54:53.881Z",
18     "updatedAt": "2024-11-17T16:01:07.476Z",
19     "user_id": 1
20   }
21 }
```


Thank You