Music Store – MongoDB NoSQL Project Report

Course: Advanced Databases (NoSQL)
Student Name: Abubakir Elnur
Group: BDA-2407
Instructor: Zhunissova Dinara

1. Introduction

This project is a full-stack web application called Music Store, developed as part of the Advanced Databases course.

The purpose of the project is to demonstrate:

- MongoDB NoSQL data modeling
- CRUD operations
- Advanced update operators
- Aggregation framework
- Indexing and performance optimization
- RESTful API development
- Authentication and authorization
- Frontend integration

The system allows users to browse musical instruments, create orders, and analyze sales data.

2. System Architecture

The application follows a three-tier architecture:

Frontend (HTML / JavaScript / CSS)
→ Backend (Node.js + Express REST API)
→ Database (MongoDB)

JWT tokens are used for authentication between the frontend and backend.

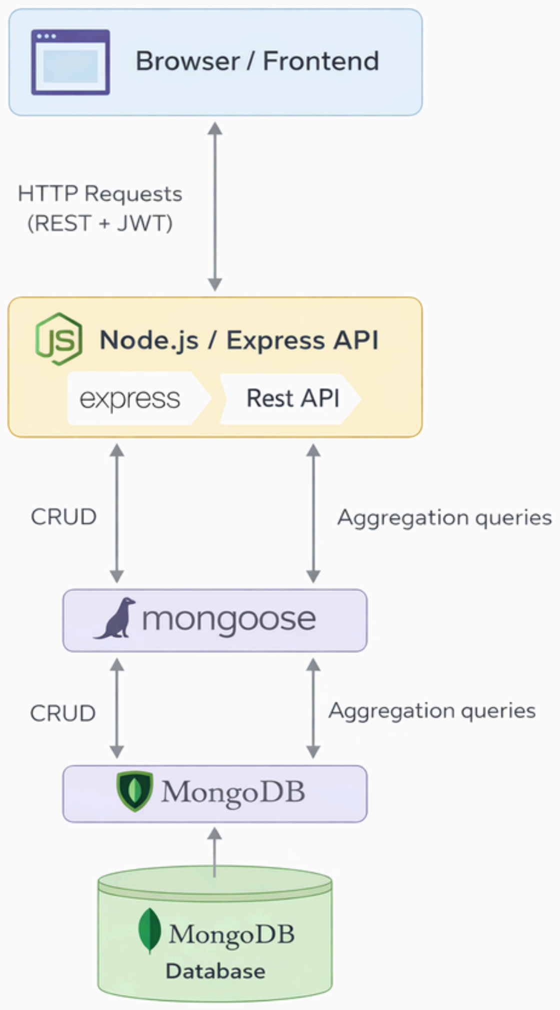


Figure 1. System architecture: the frontend communicates with the Express REST API via HTTP requests secured by JWT tokens. The backend accesses MongoDB through Mongoose ODM.

3. Technology Stack

- Node.js
- Express.js
- MongoDB
- Mongoose ODM
- JWT Authentication
- Vanilla JavaScript frontend
- HTML + CSS
- GitHub for version control

4. Database Design

4.1 Collections

The database contains three main collections:

Users

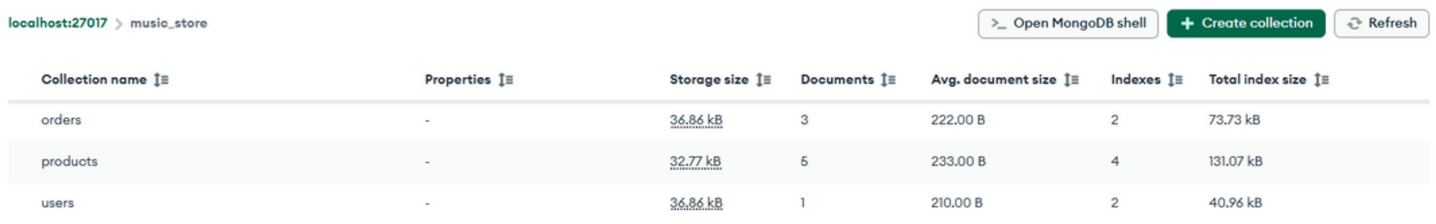
- _id
- name
- email (unique)
- passwordHash
- role

Products

- _id
- name
- category
- brand
- price
- stock
- description
- tags
- ratingAvg
- ratingCount

Orders

- _id
- userId (reference to Users)
- items (embedded array)
  - productId
  - nameSnapshot
  - priceSnapshot
  - qty
- total
- status
- createdAt



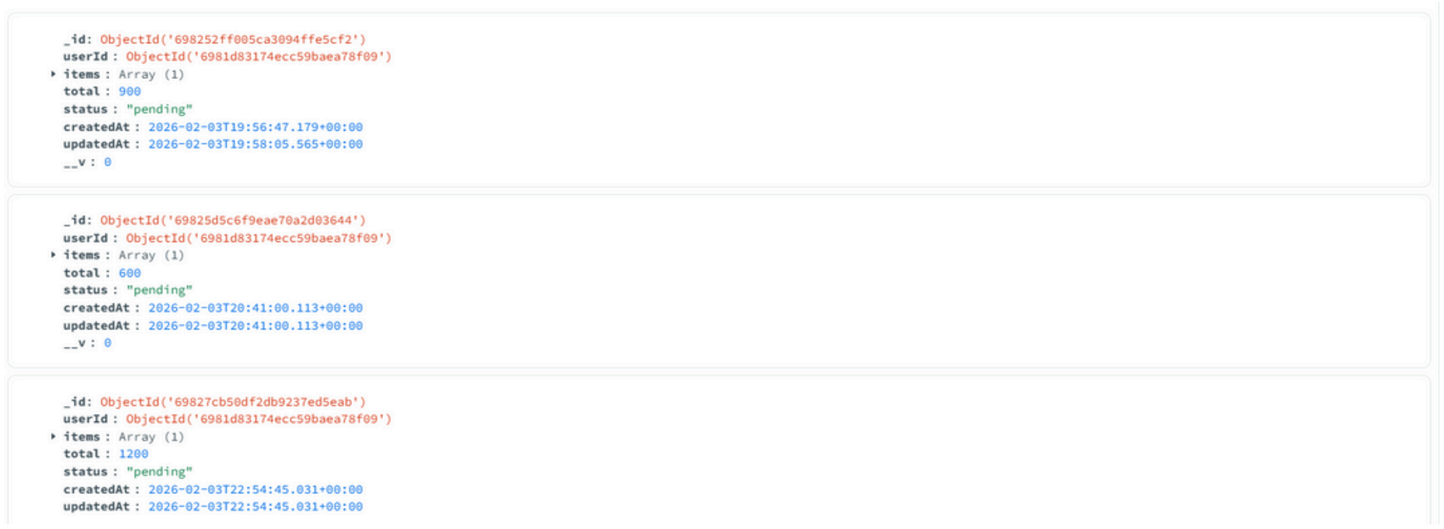
Collection name	Properties	Storage size	Documents	Avg. document size	Indexes	Total index size
orders	-	36.86 kB	3	222.00 B	2	73.73 kB
products	-	32.77 kB	5	233.00 B	4	131.07 kB
users	-	36.86 kB	1	210.00 B	2	40.96 kB

4.2 Embedded and Referenced Documents

Orders embed order items in order to preserve historical product prices.

Products and users are referenced using ObjectId relationships.

This design balances performance and data consistency.

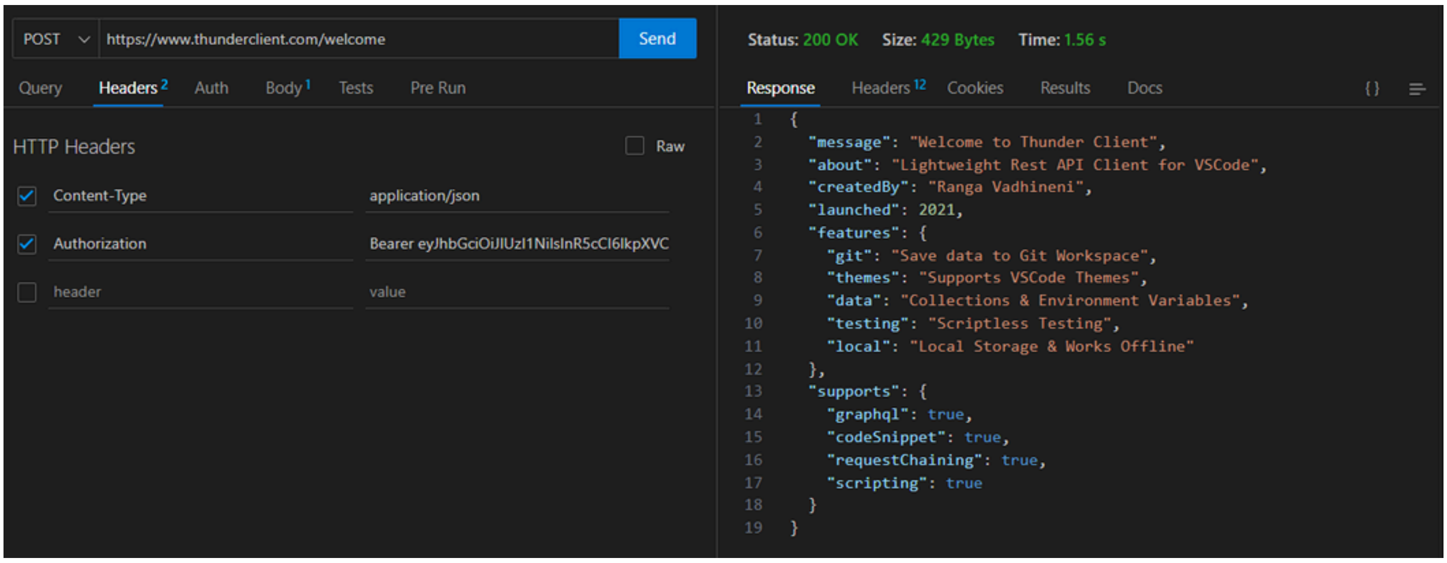


```
_id: ObjectId('698252ff005ca3094ffe5cf2')
userId : ObjectId('6981d83174ecc59baea78f09')
▸ items : Array (1)
total : 900
status : "pending"
createdAt : 2026-02-03T19:56:47.179+00:00
updatedAt : 2026-02-03T19:58:05.565+00:00
__v : 0

_id: ObjectId('69825d5c6f9eae70a2d03644')
userId : ObjectId('6981d83174ecc59baea78f09')
▸ items : Array (1)
total : 600
status : "pending"
createdAt : 2026-02-03T20:41:00.113+00:00
updatedAt : 2026-02-03T20:41:00.113+00:00
__v : 0

_id: ObjectId('69827cb50df2db9237ed5eab')
userId : ObjectId('6981d83174ecc59baea78f09')
▸ items : Array (1)
total : 1200
status : "pending"
createdAt : 2026-02-03T22:54:45.031+00:00
updatedAt : 2026-02-03T22:54:45.031+00:00
```

5. CRUD Operations

CRUD functionality was implemented mainly for the Product and Order entities.

- Create: add products, create orders

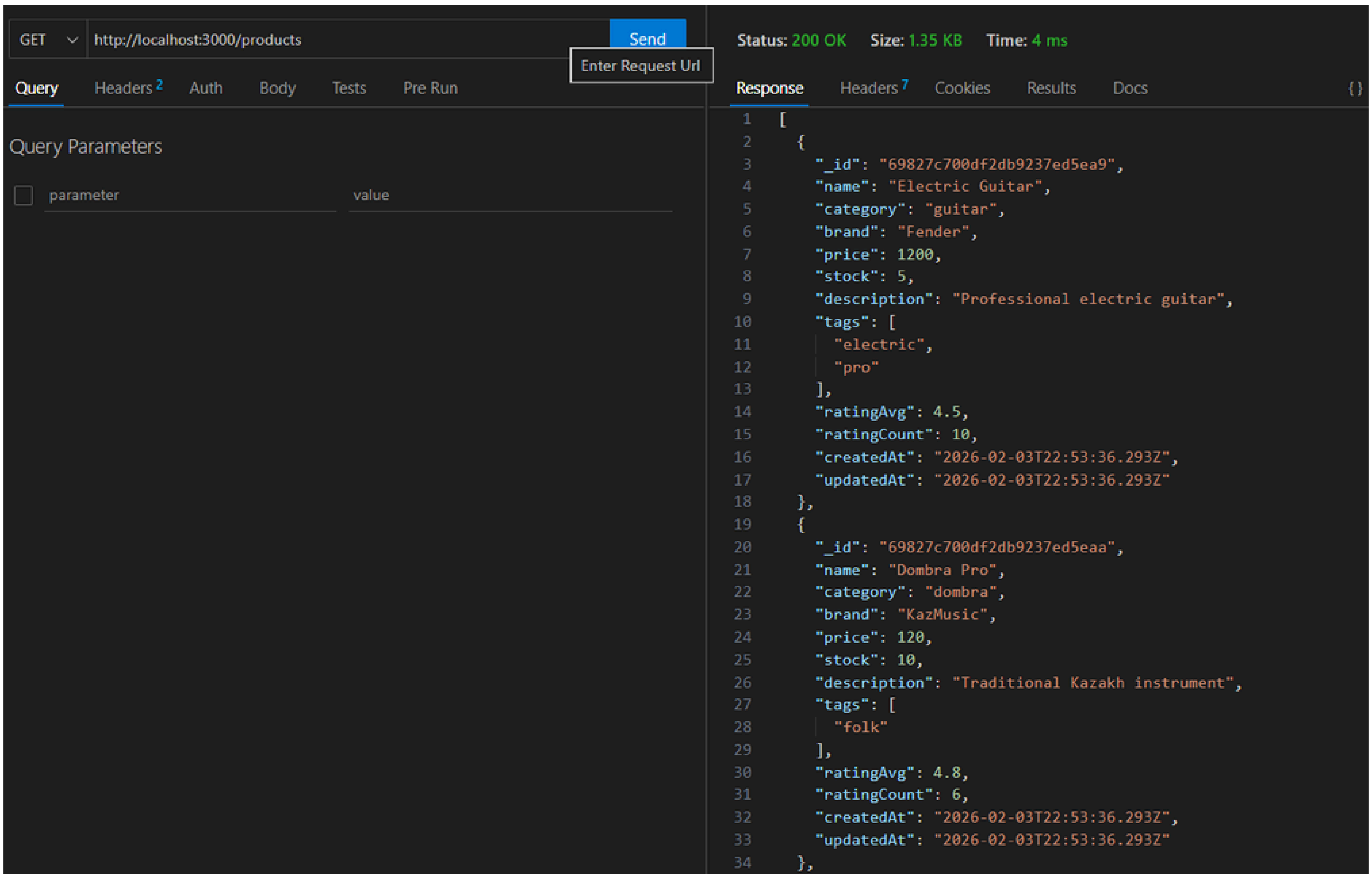


```
POST https://www.thunderclient.com/welcome

HTTP Headers
Content-Type     application/json
Authorization    Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC
header           value
```

```
Status: 200 OK   Size: 429 Bytes   Time: 1.56 s
```

```
{
  "message": "Welcome to Thunder Client",
  "about": "Lightweight Rest API Client for VSCode",
  "createdBy": "Ranga Vadhineni",
  "launched": 2021,
  "features": {
    "git": "Save data to Git Workspace",
    "themes": "Supports VSCode Themes",
    "data": "Collections & Environment Variables",
    "testing": "Scriptless Testing",
    "local": "Local Storage & Works Offline"
  },
  "supports": {
    "graphql": true,
    "codeSnippet": true,
    "requestChaining": true,
    "scripting": true
  }
}
```

- Read: list products, view orders



GET http://localhost:3000/products Send

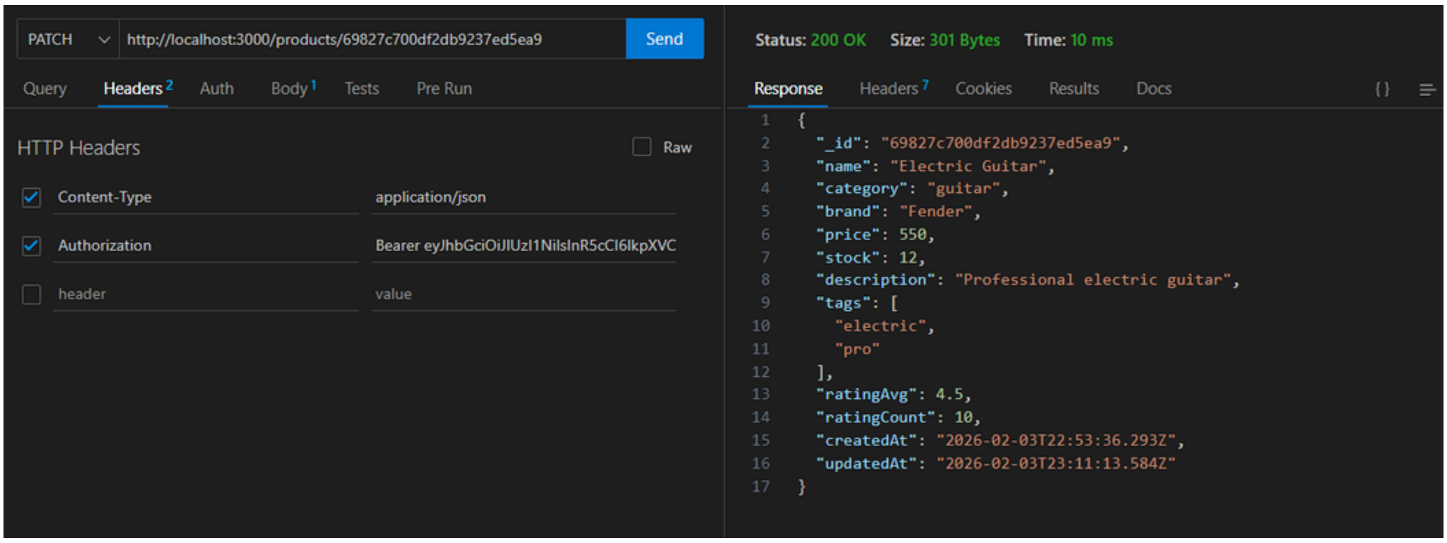
Enter Request Url

Query Parameters

Status: 200 OK Size: 1.35 KB Time: 4 ms

```
[
  {
    "_id": "69827c700df2db9237ed5ea9",
    "name": "Electric Guitar",
    "category": "guitar",
    "brand": "Fender",
    "price": 1200,
    "stock": 5,
    "description": "Professional electric guitar",
    "tags": [
      "electric",
      "pro"
    ],
    "ratingAvg": 4.5,
    "ratingCount": 10,
    "createdAt": "2026-02-03T22:53:36.293Z",
    "updatedAt": "2026-02-03T22:53:36.293Z"
  },
  {
    "_id": "69827c700df2db9237ed5eaa",
    "name": "Dombra Pro",
    "category": "dombra",
    "brand": "KazMusic",
    "price": 120,
    "stock": 10,
    "description": "Traditional Kazakh instrument",
    "tags": [
      "folk"
    ],
    "ratingAvg": 4.8,
    "ratingCount": 6,
    "createdAt": "2026-02-03T22:53:36.293Z",
    "updatedAt": "2026-02-03T22:53:36.293Z"
  },
```

- Update: modify product fields and stock



PATCH http://localhost:3000/products/69827c700df2db9237ed5ea9 Send

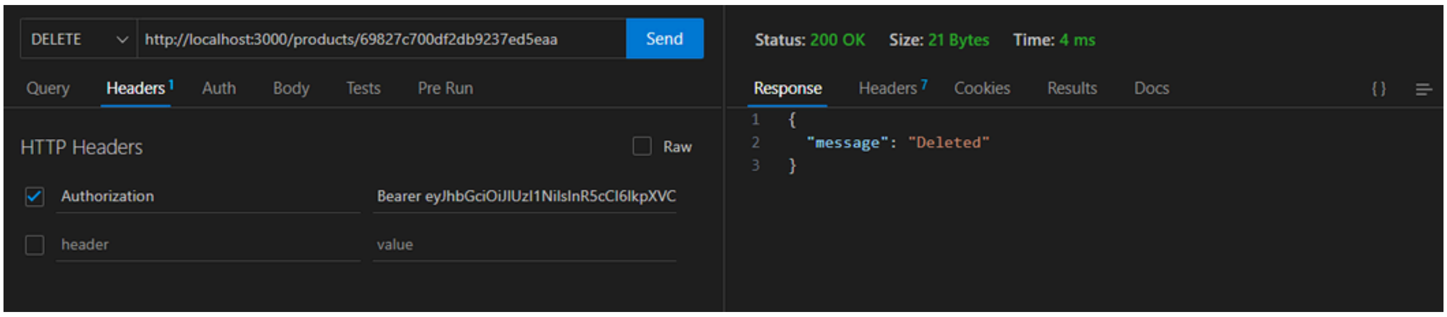
HTTP Headers

✓ Content-Type	application/json
✓ Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC

Status: 200 OK Size: 301 Bytes Time: 10 ms

```
{
  "_id": "69827c700df2db9237ed5ea9",
  "name": "Electric Guitar",
  "category": "guitar",
  "brand": "Fender",
  "price": 550,
  "stock": 12,
  "description": "Professional electric guitar",
  "tags": [
    "electric",
    "pro"
  ],
  "ratingAvg": 4.5,
  "ratingCount": 10,
  "createdAt": "2026-02-03T22:53:36.293Z",
  "updatedAt": "2026-02-03T23:11:13.584Z"
}
```

- Delete: remove products



DELETE http://localhost:3000/products/69827c700df2db9237ed5eaa Send

HTTP Headers

✓ Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC

Status: 200 OK Size: 21 Bytes Time: 4 ms

```
{
  "message": "Deleted"
}
```

6. Advanced MongoDB Operations

The project demonstrates several advanced update operators:

- $set – update product fields
- $inc – update stock values
- $push – add tags
- $pull – remove tags

These operators allow efficient partial document updates.

7. Aggregation Framework

MongoDB aggregation pipelines were implemented to analyze sales data.

Examples include:

- Total revenue calculation
- Number of orders
- Average order value

The pipeline uses multiple stages:

$match → $group → $unwind → $sort

This provides real business insights for the store.

{Screenshot of aggregation endpoint response}

8. Indexes and Performance

Indexes were created to improve query performance:

- Compound index on category + price
- Text index for searching products
- Compound index on userId + createdAt in orders

These indexes speed up filtering, sorting, and analytics queries.

{Screenshot from MongoDB Compass showing indexes}

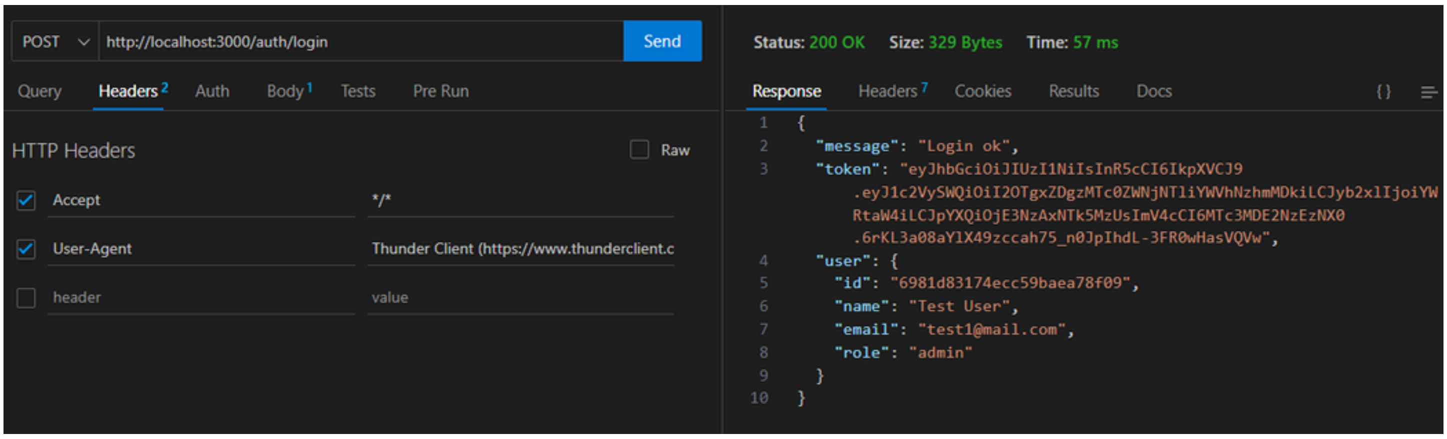
9. REST API Design

The backend follows REST principles:

- Resource-based endpoints
- HTTP verbs for operations
- JSON request and response bodies

Main endpoints include:

- /auth/login



POST http://localhost:3000/auth/login

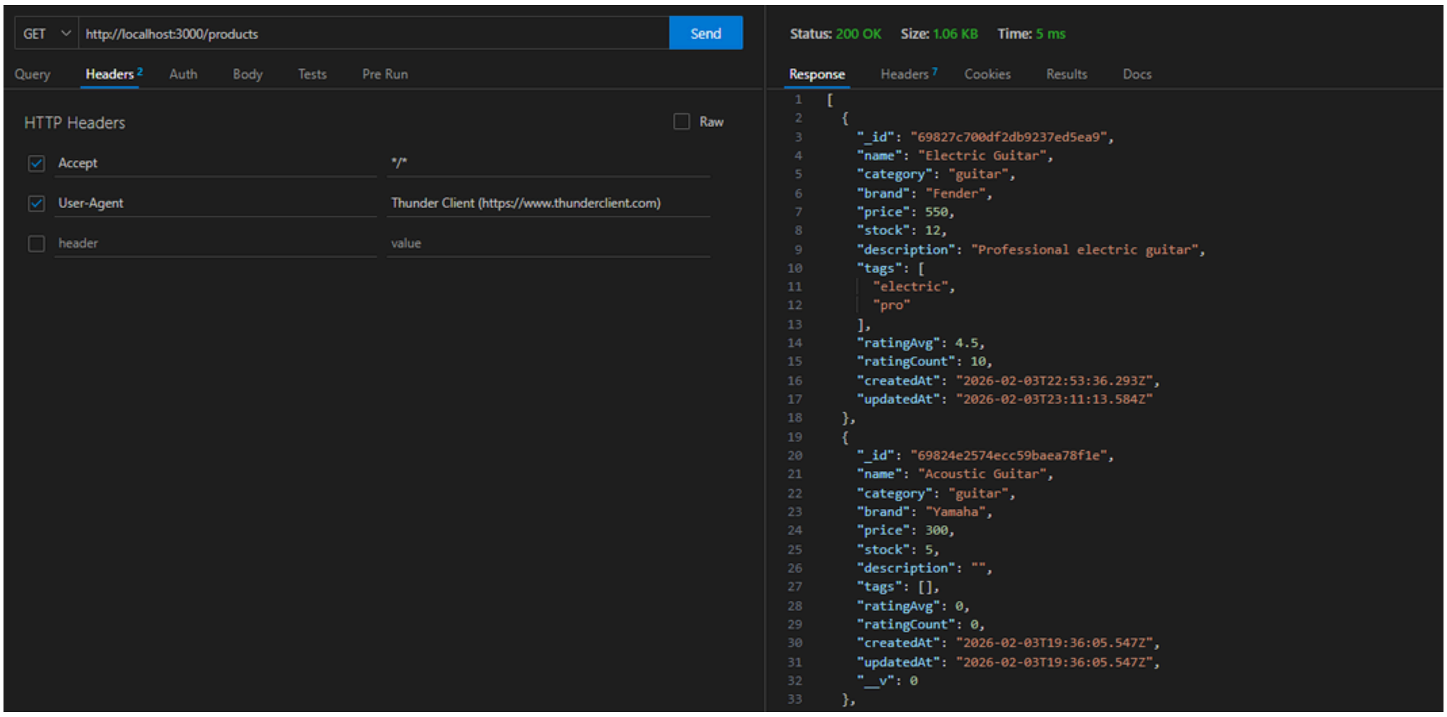
HTTP Headers

```
Accept        */*
User-Agent    Thunder Client (https://www.thunderclient.c
header        value
```

Status: 200 OK   Size: 329 Bytes   Time: 57 ms

```
{
  "message": "Login ok",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
     .eyJ1c2VySWQiOiI2OTgxZDgzMTc0ZWNjNTliYWVhNzhmMDkiLCJyb2xlIjoiYW
     RtaW4iLCJpYXQiOjE3NzAxNTk5MzUsImV4cCI6MTc3MDE2NzEzNX0
     .6rKL3a08aY1X49zccah75_n0JpIhdL-3FR0wHasVQVw",
  "user": {
    "id": "6981d83174ecc59baea78f09",
    "name": "Test User",
    "email": "test1@mail.com",
    "role": "admin"
  }
}
```

- /products



GET http://localhost:3000/products

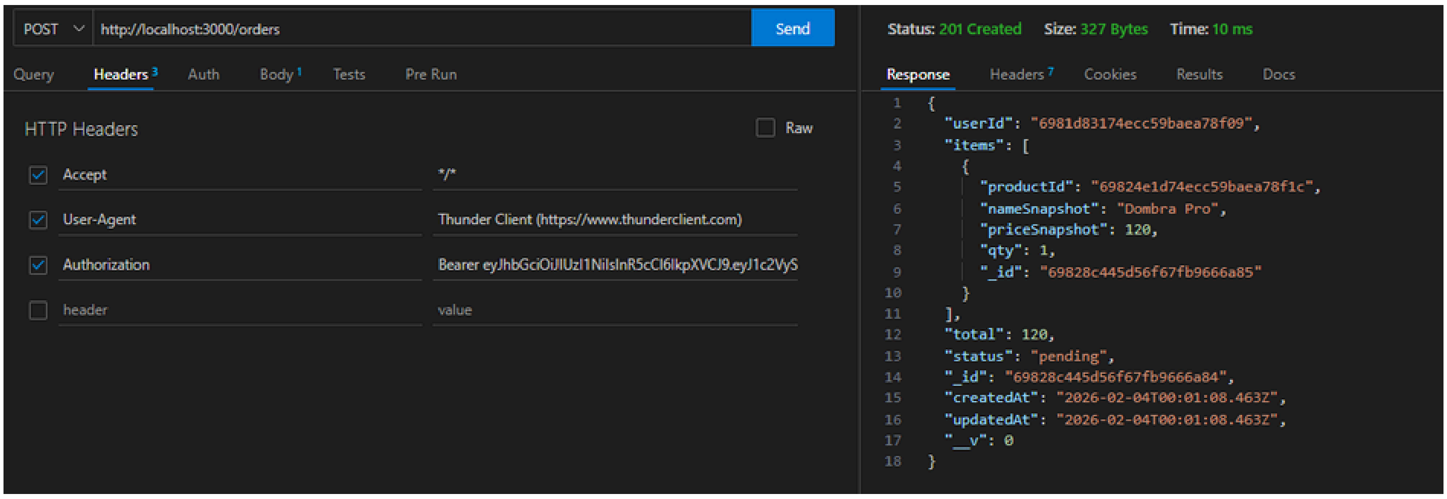
HTTP Headers

```
Accept        */*
User-Agent    Thunder Client (https://www.thunderclient.com)
header        value
```

Status: 200 OK   Size: 1.06 KB   Time: 5 ms

```
[
  {
    "_id": "69827c700df2db9237ed5ea9",
    "name": "Electric Guitar",
    "category": "guitar",
    "brand": "Fender",
    "price": 550,
    "stock": 12,
    "description": "Professional electric guitar",
    "tags": [
      "electric",
      "pro"
    ],
    "ratingAvg": 4.5,
    "ratingCount": 10,
    "createdAt": "2026-02-03T22:53:36.293Z",
    "updatedAt": "2026-02-03T23:11:13.584Z"
  },
  {
    "_id": "69824e2574ecc59baea78f1e",
    "name": "Acoustic Guitar",
    "category": "guitar",
    "brand": "Yamaha",
    "price": 300,
    "stock": 5,
    "description": "",
    "tags": [],
    "ratingAvg": 0,
    "ratingCount": 0,
    "createdAt": "2026-02-03T19:36:05.547Z",
    "updatedAt": "2026-02-03T19:36:05.547Z",
    "__v": 0
  },
```

- /orders



POST http://localhost:3000/orders

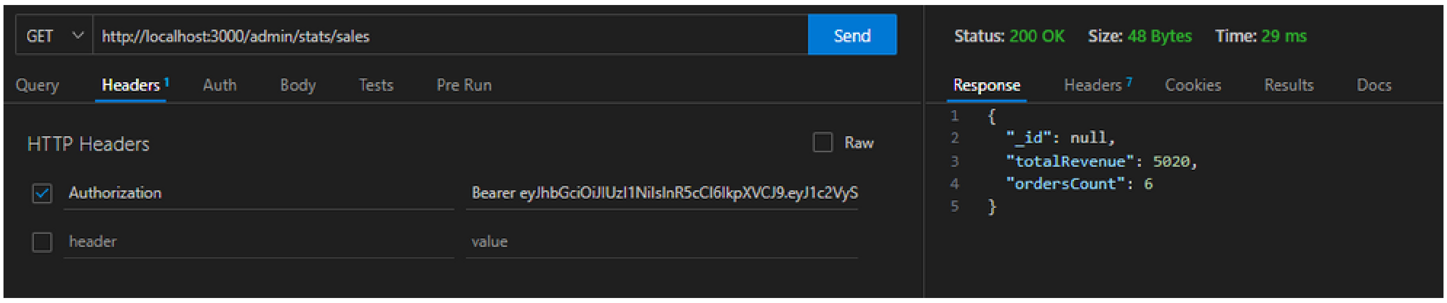
HTTP Headers

```
Accept          */*
User-Agent      Thunder Client (https://www.thunderclient.com)
Authorization   Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyS
header          value
```

Status: 201 Created   Size: 327 Bytes   Time: 10 ms

```
{
  "userId": "6981d83174ecc59baea78f09",
  "items": [
    {
      "productId": "69824e1d74ecc59baea78f1c",
      "nameSnapshot": "Dombra Pro",
      "priceSnapshot": 120,
      "qty": 1,
      "_id": "69828c445d56f67fb9666a85"
    }
  ],
  "total": 120,
  "status": "pending",
  "_id": "69828c445d56f67fb9666a84",
  "createdAt": "2026-02-04T00:01:08.463Z",
  "updatedAt": "2026-02-04T00:01:08.463Z",
  "__v": 0
}
```

- /orders/stats/sales



```
GET http://localhost:3000/admin/stats/sales
Headers
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyS
```

```
Status: 200 OK   Size: 48 Bytes   Time: 29 ms
{
  "_id": null,
  "totalRevenue": 5020,
  "ordersCount": 6
}
```

10. Security Implementation

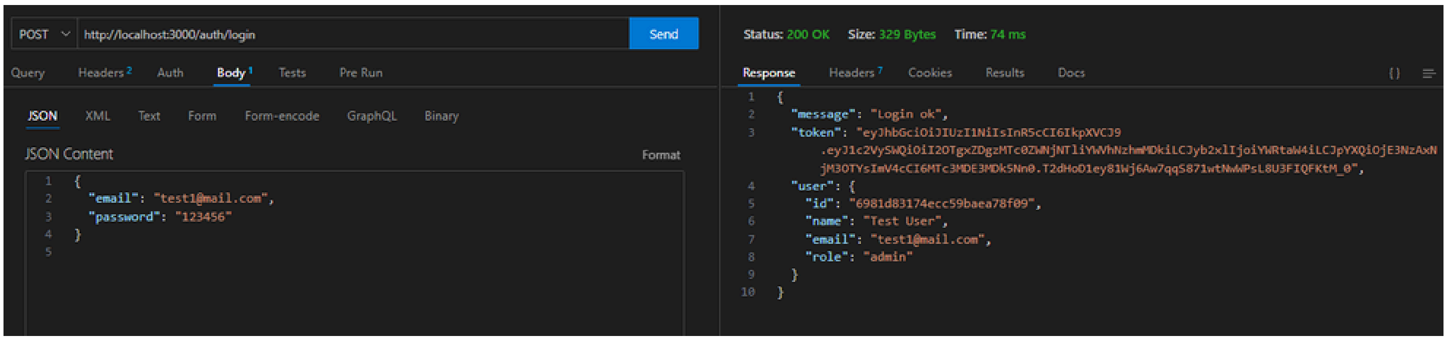
Security is handled using JWT authentication.

Protected routes require an Authorization header:

Authorization: Bearer <token>

Role-based authorization is used for admin-only operations.

Passwords are stored in hashed format using bcrypt.



```
POST http://localhost:3000/auth/login
Body (JSON)
{
  "email": "test1@mail.com",
  "password": "123456"
}
```

```
Status: 200 OK   Size: 329 Bytes   Time: 74 ms
{
  "message": "Login ok",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJ1c2VySWQiOiI2OTgxZDgzMTc0ZWNjNTliYWVhNzhmMDkiLCJyb2xlIjoiYWRtaW4iLCJpYXQiOjE3NzAxN
      jM3OTYsImV4cCI6MTc3MDE3MDk5Nn0.T2dHoD1ey81Wj6Aw7qqS871wtNwWPsL8U3FIQFKtM_0",
  "user": {
    "id": "6981d83174ecc59baea78f09",
    "name": "Test User",
    "email": "test1@mail.com",
    "role": "admin"
  }
}
```

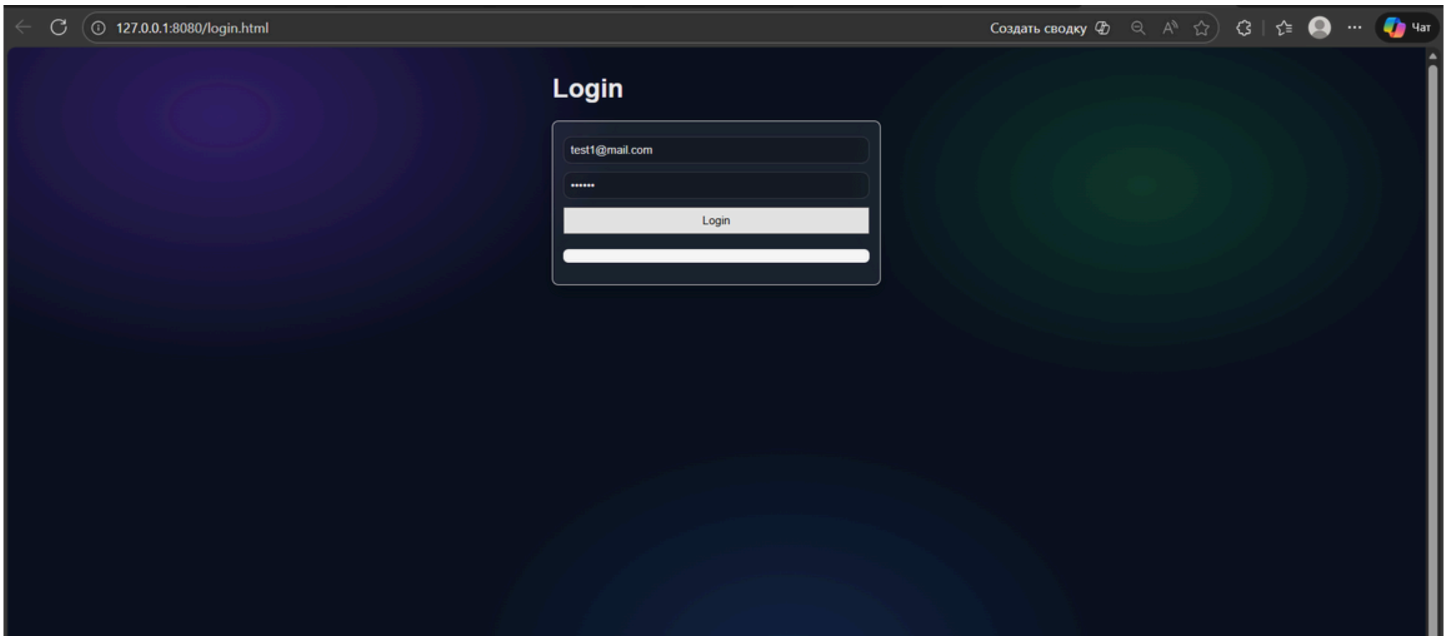
11. Frontend Implementation

The frontend consists of four pages:

- Login page
- Products list
- Create order
- My orders

Each page communicates with the backend using real HTTP requests.

CSS styling was added to create a clean and usable interface.

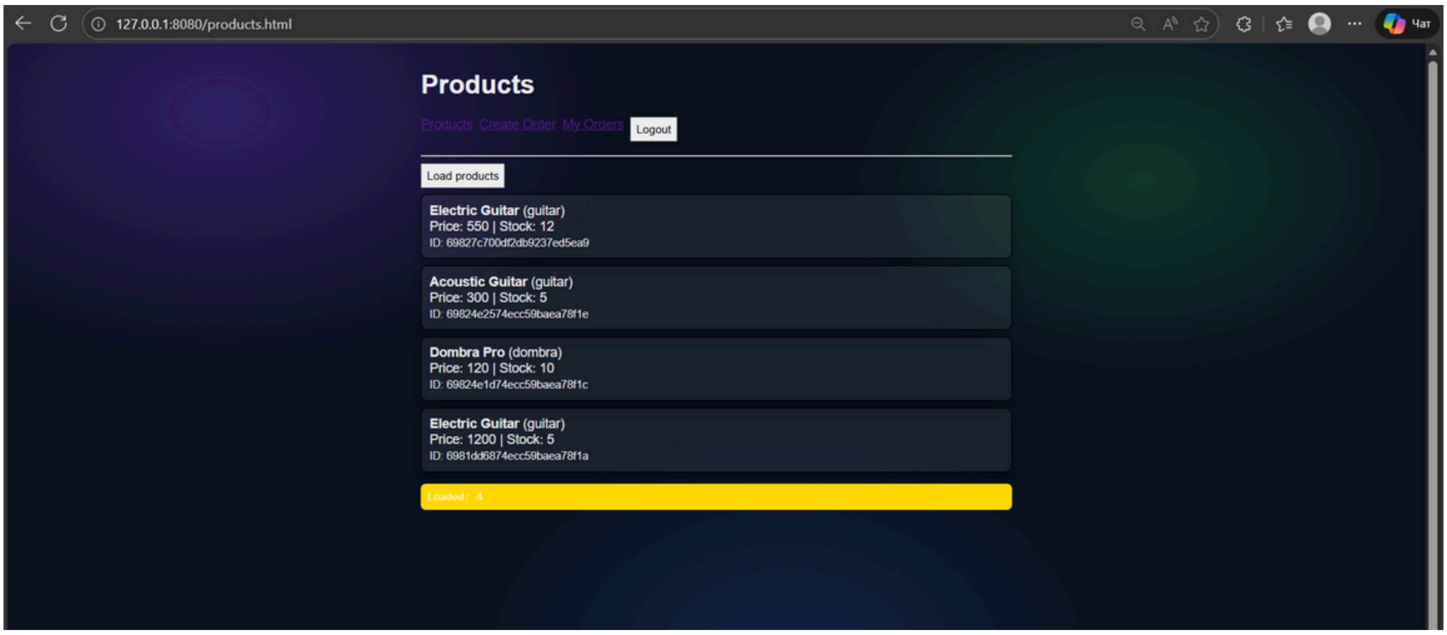


Login

test1@mail.com

••••••

Login



Products

Products Create Order My Orders Logout

Load products

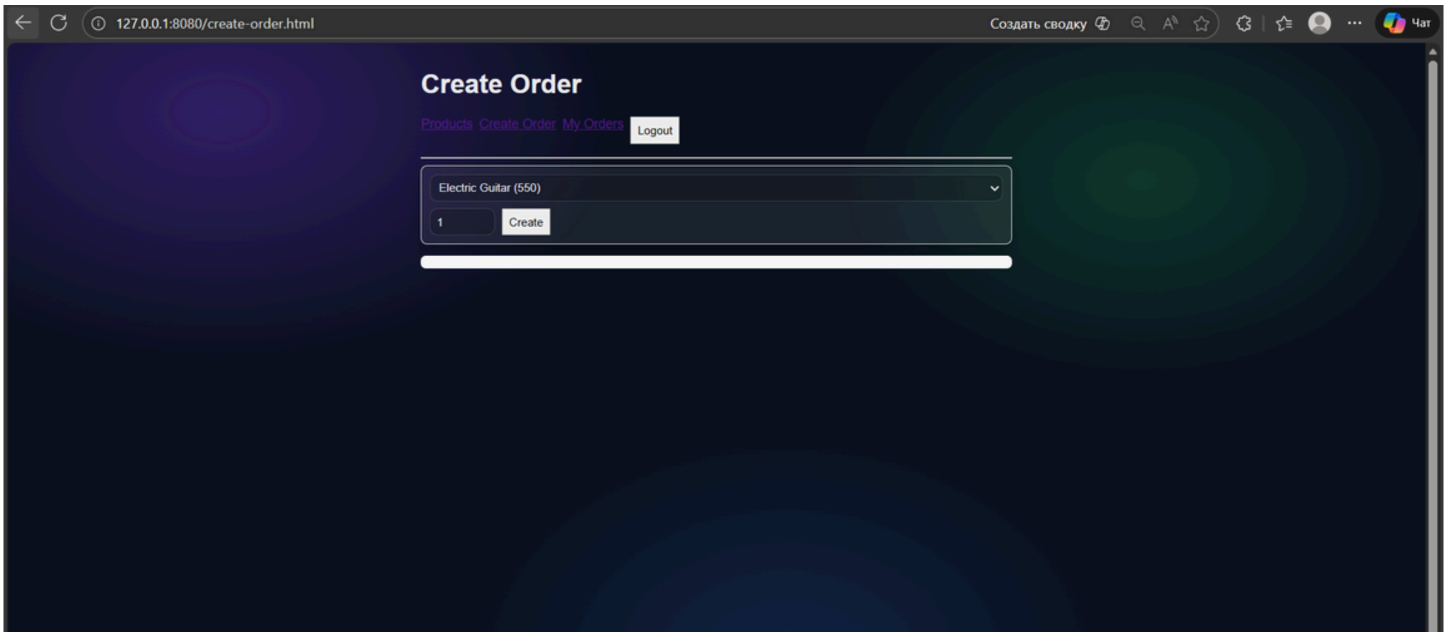
Electric Guitar (guitar)
Price: 550 | Stock: 12
ID: 69827c700df2db9237ed5ea9

Acoustic Guitar (guitar)
Price: 300 | Stock: 5
ID: 69824e2574ecc59baea78f1e

Dombra Pro (dombra)
Price: 120 | Stock: 10
ID: 69824e1d74ecc59baea78f1c

Electric Guitar (guitar)
Price: 1200 | Stock: 5
ID: 6981dd6874ecc59baea78f1a

Loaded: 4

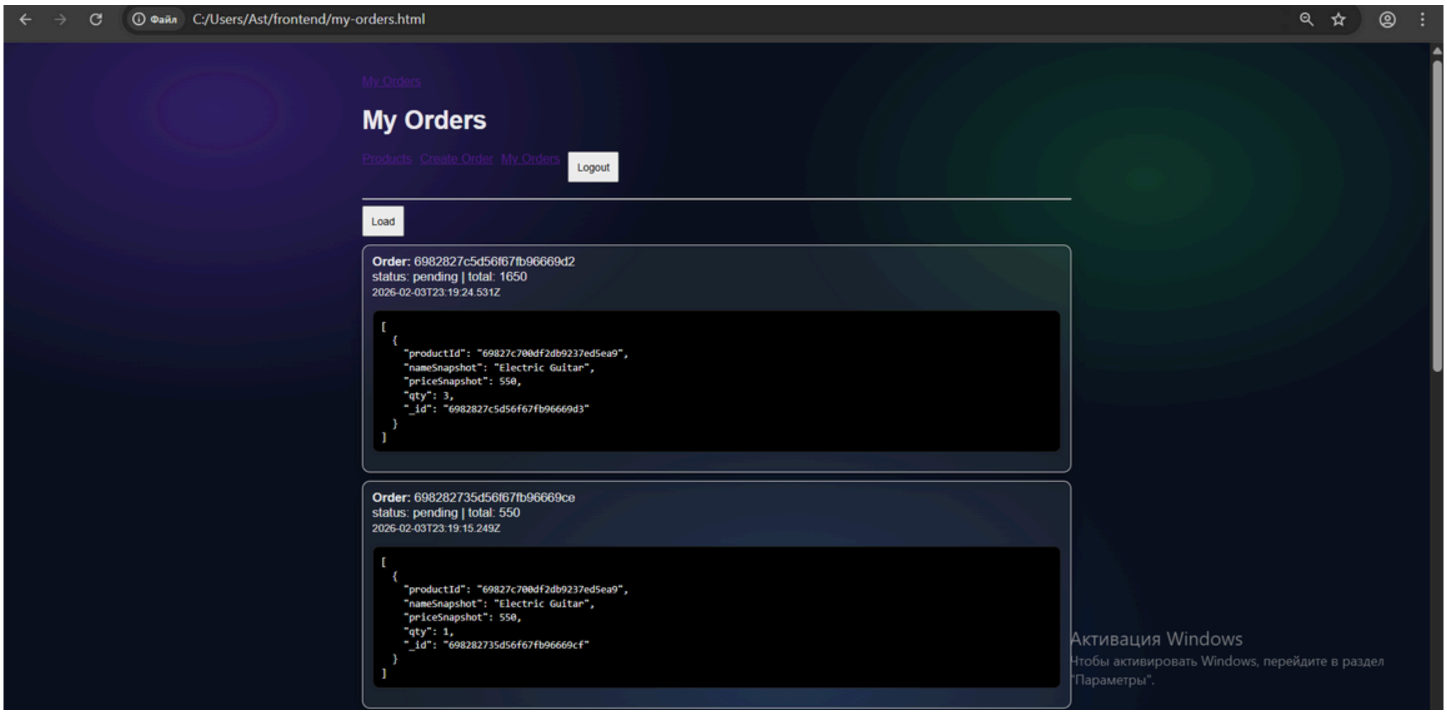


Create Order

Products Create Order My Orders Logout

Electric Guitar (550)

1 Create


Файл C:/Users/Ast/frontend/my-orders.html
My Orders
Products Create Order My Orders Logout
Load
Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

My Orders

Order: 6982827c5d56f67fb96669d2
status: pending | total: 1650
2026-02-03T23:19:24.531Z

```
[
  {
    "productId": "69827c700df2db9237ed5ea9",
    "nameSnapshot": "Electric Guitar",
    "priceSnapshot": 550,
    "qty": 3,
    "_id": "6982827c5d56f67fb96669d3"
  }
]
```

Order: 698282735d56f67fb96669ce
status: pending | total: 550
2026-02-03T23:19:15.249Z

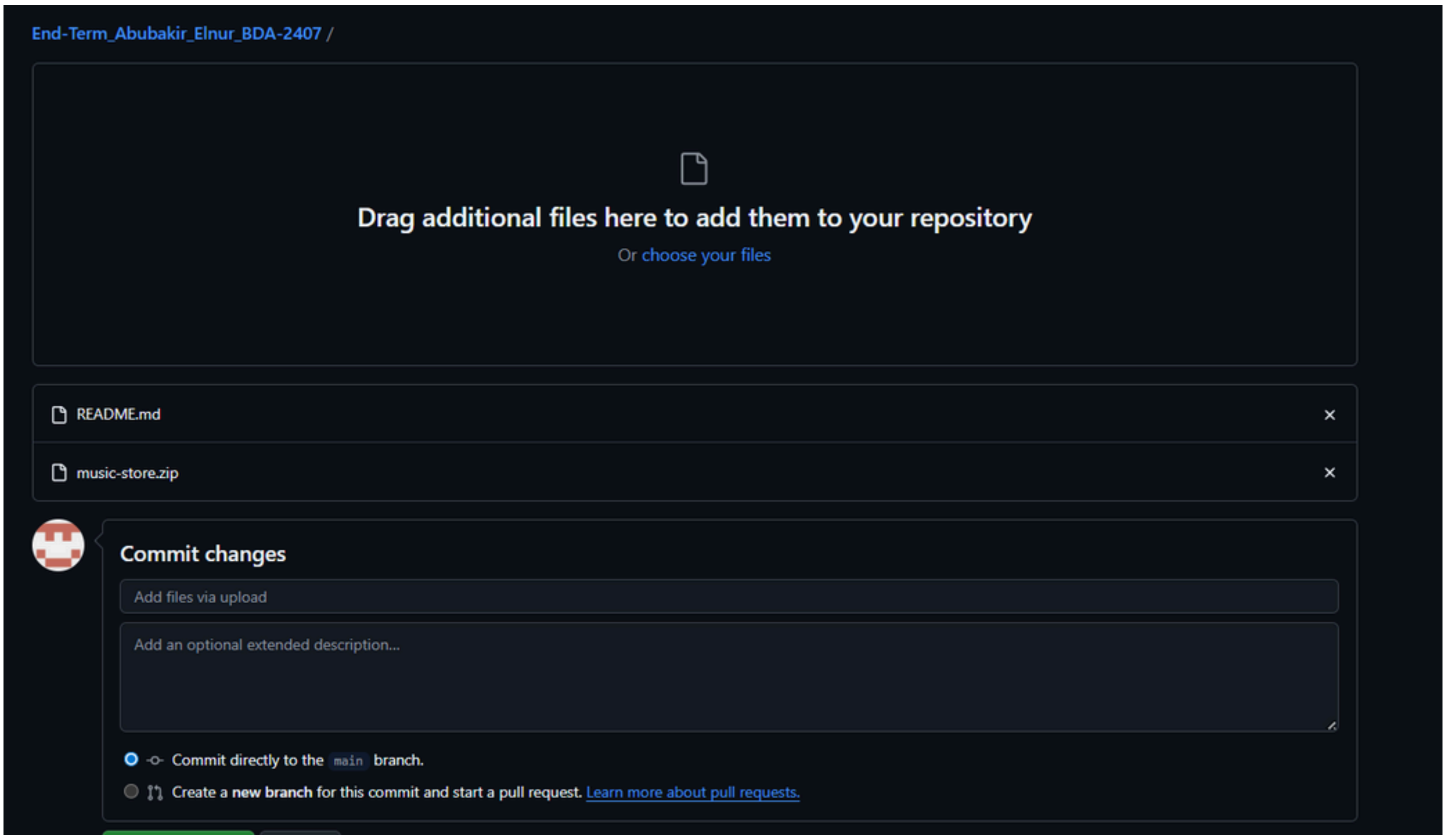
```
[
  {
    "productId": "69827c700df2db9237ed5ea9",
    "nameSnapshot": "Electric Guitar",
    "priceSnapshot": 550,
    "qty": 1,
    "_id": "698282735d56f67fb96669cf"
  }
]
```

12. GitHub Repository

All project files are uploaded to GitHub, including:

- Backend source code
- Frontend files
- README documentation

Version control was used during development.


End-Term_Abubakir_Elnur_BDA-2407 /
Drag additional files here to add them to your repository
Or choose your files
Commit changes
Add files via upload
Add an optional extended description...
Commit directly to the main branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.

README.md
music-store.zip

13. Challenges and Solutions

During development, several technical issues occurred, such as:

- Route configuration errors
- MongoDB connection issues
- Index duplication warnings

These problems were solved by restructuring routes, fixing schema definitions, and cleaning up indexes.

14. Conclusion

This project successfully demonstrates advanced MongoDB usage together with a REST API and frontend interface.

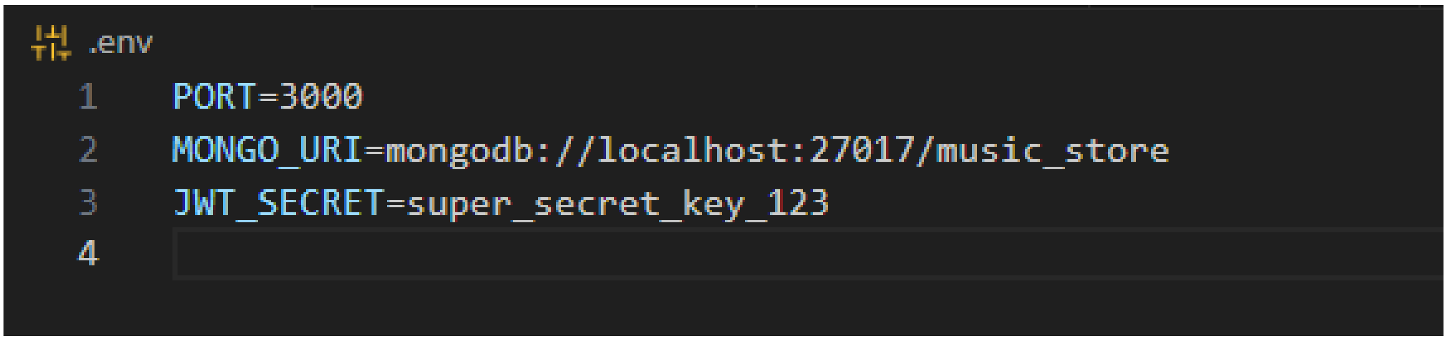
All major grading requirements were addressed, including aggregation, indexing, advanced updates, and documentation.

The Music Store system could be extended in the future with:

- Payment processing
- Admin dashboard
- Review system
- Deployment to cloud platforms

15. Appendix (Optional)

- Example API requests
- Environment variables
- Sample database records



```
PORT=3000
MONGO_URI=mongodb://localhost:27017/music_store
JWT_SECRET=super_secret_key_123

```