FPT UNIVERSITY

Assignment 4 - Data Rendering & Relations with EJS

I. Instructions

In this second assignment, you will build upon your Multiple-Choice Exam Question Bank backend by creating a set of endpoints that retrieve and present data using **EJS-based HTML views**. This assignment emphasizes:

- Displaying MongoDB data on web pages rendered using the **EJS template engine**
- Using **Mongoose's `populate()`** to pull related data from multiple collections (such as `subjects`, `topics`, and `users`)
- Structuring clean and readable views for real-time browsing of the question bank

You will gain practical experience in:

- Fetching related documents from multiple collections using `.populate()`
- Rendering data as HTML using EJS view templates
- Designing API routes that grow in complexity and data composition

II. Assignment Overview

- **Main Technologies**: Node.js, Express.js, MongoDB, Mongoose, JWT, bcrypt, dotenv, cors
- **Tools**: Postman, MongoDB Compass, Visual Studio Code, Curl

Collections:
You will create 4 collections:

1. `Users` – to store user credentials (teacher accounts)
2. `Subjects` – high-level domains (e.g., Math, Physics)
3. `Topics` – sub-categories under each subject (e.g., Algebra)
4. `Questions` – multiple-choice questions with 4 options and 1 correct answer

You will implement the following **three GET endpoints**, each building on the previous one in complexity:

Task	Endpoint	Description
1	`/subjects/view`	Show all subjects as a simple HTML list
2	`/topics/view`	Show all topics with their associated subject names
3	`/questions/view`	Show all questions with subject, topic, and author populated and displayed in a table

By the end of this assignment, you will be able to:

- Use **Mongoose's `.populate()`** to retrieve related data across collections, such as:
  - The subject name linked to each topic or question
    - The topic name and the author's username associated with a question

  - Render dynamic HTML views using the **EJS template engine** based on data retrieved from the database
- Build public-facing routes that display:
  - A list of all subjects
  - A list of topics with their corresponding subject names
  - A table of questions including their subject, topic, and author
- Organize your views into reusable `.ejs` files and separate the route logic from the presentation layer
- Return clean and semantic HTML from your Express routes using `res.render()`
- Test endpoints using **Postman** and view results via a **web browser**

Authentication Note: This assignment does not require login or JWT authentication — all EJS-rendered endpoints are publicly accessible and intended for read-only viewing of the question bank contents.

III. Assignment Requirements

1. Task 1: Project Initialization and Configuration

◈ **Description:**

- Create a new Node.js project.
- Install required packages: `express`, `mongoose`, `dotenv`, `cors`, `bcryptjs`, `jsonwebtoken`
- Configure environment variables using `.env` file:
- Create `.env`:

```
PORT=6000
MONGO_URI=mongodb://localhost:27017/assignment4
JWT_SECRET=your_jwt_abc
```

✅ **Test**: Start server → http://localhost:6000

Expected Outcome:

- The server starts successfully on `http://localhost:6000`
- MongoDB connects without error

2. Task 2: Get All Questions (Public)

Endpoint: `/ questions/view`

◈ **Description:**

Render all questions using `.populate()` to include:

- `questionText`, `options`, `correctAnswer`
- Subject name (via `subjectId`)
- Topic name (via `topicId`)
- Author username (via `createdBy`)
- Display results in a formatted HTML `<table>`
- No authentication required

◆ Expected Response:


```
[
  {
    "questionText": "What is the solution to 2x + 3 = 7?",
    "options": {
      "A": "x = 1",
      "B": "x = 2",
      "C": "x = 3",
      "D": "x = 4"
    },
    "correctAnswer": "B",
    "subjectId": { "_id": "64aee200c438b927a9cfa222", "name": "Mathematics" },
    "topicId": { "_id": "64aee210c438b927a9cfa333", "name": "Algebra" },
    "createdBy": { "_id": "64aee111c438b927a9cfa111", "username": "teacher01" }
  }
]
```


Postman Test:

```
GET http://localhost:8386/questions/view
```

◆ Expected HTML Output

```
<h2>All Questions</h2>
<table border="1">
  <thead>
    <tr>
      <th>Question</th>
      <th>Subject</th>
      <th>Topic</th>
      <th>Author</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>What is 2 + 2?</td>
      <td>Mathematics</td>
      <td>Arithmetic</td>
      <td>teacher01</td>
    </tr>
  </tbody>
</table>
```

◆ EJS Logic (example)

```
<table>
  <% questions.forEach(q => { %>
    <tr>
      <td><%= q.questionText %></td>
      <td><%= q.subjectId.name %></td>
      <td><%= q.topicId.name %></td>
      <td><%= q.createdBy.username %></td>
    </tr>
  <% }) %>
</table>
```

3. Task 3: View All Subjects

Endpoint: /subjects/view

◈ Description:

- Render a list of all subjects from the database.
- Use `Subject.find()` to get all subjects.
- No `.populate()` is needed here.
- No authentication required.

✅ Expected Response

```
[
  { "_id": "64aee200c438b927a9cfa222", "name": "Mathematics" },
  { "_id": "64aee201c438b927a9cfa223", "name": "Physics" }
]
```

✅ Postman Test:

GET http://localhost:8386/subjects/view

🔷 Expected HTML Output:

```
<h2>All Subjects</h2>
<ul>
  <li>Mathematics</li>
  <li>Physics</li>
</ul>
```

🔷 EJS Logic (example):

```
<ul>
  <% subjects.forEach(s => { %>
    <li><%= s.name %></li>
  <% }) %>
</ul>
```

4. Task 4: View Topics with Subjects

◈ Endpoint: **/topics/view**

◈ Description:

- Render all topics with the names of their associated subjects.
- Use `.populate('subjectId', 'name')` to get the subject name for each topic.
- No authentication required.

✅ Example Response:

```
[
  {
    "_id": "64aee210c438b927a9cfa333",
    "name": "Algebra",
    "subjectId": { "_id": "64aee200c438b927a9cfa222", "name": "Mathematics" }
  }
]
```

◈ Postman Test (Preview)

```
GET http://localhost:8386/topics/view
```

◆ Expected HTML Output

```
<h2>All Topics</h2>
<ul>
  <li>Algebra - Mathematics</li>
  <li>Mechanics - Physics</li>
</ul>
```

IV. 🔐 Data structure

1. Collection: users

Stores user information (e.g., teachers who create questions).

Field	Data Type	Description
_id	`ObjectId`	Automatically generated by MongoDB
username	`String`	Unique account name
password	`String`	Password hashed using `bcrypt`
createdAt	`Date`	Timestamp when the account was created

🔐 Passwords **must be hashed** before storage.

2. Collection: subjects

Stores **academic subjects**, such as *Mathematics*, *Physics*, etc.

Field	Data Type	Description
_id	`ObjectId`	Auto-generated by MongoDB
name	`String`	Name of the subject

3. Collection: topics

Stores **specific topics** under each subject (e.g., *Algebra* under *Mathematics*).

Field	Data Type	Description
_id	`ObjectId`	Auto-generated
name	`String`	Topic name (e.g., Calculus, Mechanics)
subjectId	`ObjectId`	References the `_id` field of the `subjects` collection

🔗 `subjectId` is a **foreign key** linking each topic to a subject.

4. Collection: questions

Stores multiple-choice questions.

Field	Data Type	Description
_id	`ObjectId`	Auto-generated
questionText	`String`	The question prompt
options	`Object`	4 answer options: `{ "A": "...", "B": "...", "C": "...", "D": "..." }`
correctAnswer	`String`	One of `"A"`, `"B"`, `"C"`, or `"D"` as the correct answer
subjectId	`ObjectId`	References `_id` in the `subjects` collection
topicId	`ObjectId`	References `_id` in the `topics` collection

`createdBy`	`ObjectId`	References `_id` in the `users` collection (who created the question)
`createdAt`	`Date`	Creation timestamp

🧠 You can later expand `options` to support random shuffling or metadata.

5.Relationship Summary:

- `topics.subjectId` → references `subjects._id`
- `questions.subjectId` & `questions.topicId` → references `subjects._id` and `topics._id`
- `questions.createdBy` → references `users._id`

6. Sample Data:

```
sample_data/
├── subjects.json
├── topics.json
└── questions.json
```

✅ `subjects.json`

```
[
  { "_id": { "$oid": "64aee200c438b927a9cfa222" }, "name": "Mathematics" },
  { "_id": { "$oid": "64aee201c438b927a9cfa223" }, "name": "Physics" }
]
```

✅ `topics.json`

```
[
  {
    "_id": { "$oid": "64aee210c438b927a9cfa333" },
    "name": "Algebra",
    "subjectId": { "$oid": "64aee200c438b927a9cfa222" }
  },
  {
    "_id": { "$oid": "64aee211c438b927a9cfa334" },
    "name": "Mechanics",
    "subjectId": { "$oid": "64aee201c438b927a9cfa223" }
  }
]
```

✅ `questions.json`

```
[
  {
    "_id": { "$oid": "64aee300c438b927a9cfa444" },
    "questionText": "What is the solution to 2x + 3 = 7?",
    "options": {
      "A": "x = 1",
      "B": "x = 2",
      "C": "x = 3",
      "D": "x = 4"
    },
    "correctAnswer": "B",
    "topicId": { "$oid": "64aee210c438b927a9cfa333" },
    "subjectId": { "$oid": "64aee200c438b927a9cfa222" },
    "createdBy": { "$oid": "64aee188c438b927a9cfa111" },
    "createdAt": { "$date": "2025-07-01T10:00:00.000Z" }
  },
  {
    "_id": { "$oid": "64aee301c438b927a9cfa445" },
    "questionText": "What is the acceleration due to gravity on Earth?",
```

```
    "options": {
      "A": "8.9 m/s²",
      "B": "10.0 m/s²",
      "C": "9.8 m/s²",
      "D": "9.0 m/s²"
    },
    "correctAnswer": "C",
    "topicId": { "$oid": "64aee211c438b927a9cfa334" },
    "subjectId": { "$oid": "64aee201c438b927a9cfa223" },
    "createdBy": { "$oid": "64aee188c438b927a9cfa111" },
    "createdAt": { "$date": "2025-07-01T11:00:00.000Z" }
  }
]
```


V. HÌNH THỨC NỘP BÀI

- **Gửi bài tập dưới dạng file nén (.zip/.rar)**, gồm:
  - **Chụp screenshoot toàn màn hình: Thư mục, comment tác giả đầu mỗi file source code, cấu trúc vscode, kết quả thực thi đầy đủ**.
  - **source code**
  - **File tài liệu báo cáo (.PDF hoặc .DOCX)**.

- **Đặt tên file nén theo format:**
  - **[MãSV]_[Tên]_assignment3.zip**
    Ví dụ: SE12345_NguyenVanA_ **assignment3**.zip

- **Hạn chót nộp bài:** theo lịch Edunext

VI. LƯU Ý QUAN TRỌNG

X Bài nộp không đầy đủ hoặc thiếu file báo cáo sẽ bị trừ điểm.

X Mọi hành vi sao chép code sẽ bị xử lý theo quy định của nhà trường.

X Sinh viên cần kiểm tra kỹ lưỡng trước khi nộp bài.

- **Được sử dụng AI để phân tích và thực hiện bài.**