Step 1: Project Setup

1. **Create Your Project Folder & Initialize**

Open your terminal and run:

```
mkdir store-app
cd store-app
npm init -y
```

Then, open your generated package.json and add the following line to enable ES modules:

```
{
  "type": "module",
  ...
}
```

2. **Install Required Dependencies**

Run this command to install Express, EJS, and Mongoose:

```
npm install express ejs mongoose
```

Step 2: Organize Your Folder Structure

Inside your project folder, create the following directories and file:

```
store-app/
├── models/
│   ├── Product.js
│   ├── User.js
│   └── Order.js
├── routes/
│   └── products.js
├── views/
│   ├── partials/
│   │   ├── header.ejs
│   │   └── footer.ejs
│   ├── index.ejs
```

```
│   └── products/
│       └── index.ejs
├── public/
│   └── css/
│       └── styles.css
└── app.js
```

Each folder is used as follows:

• **models/**: Contains Mongoose schemas.

• **routes/**: Contains Express route files.

• **views/**: Contains EJS templates (with partials in the partials/ folder).

• **public/**: Contains static assets (like CSS).

• **app.js**: The main application file.

Step 3: Connect to MongoDB Atlas and Set Up the Server

1. **Create app.js**

In the root of your project, create app.js with the following content.

Replace with your actual MongoDB Atlas connection string:

```
import express from 'express';
import mongoose from 'mongoose';
import path from 'path';
import { fileURLToPath } from 'url';
import productsRouter from './routes/products.js';

const app = express();

// Set __dirname for ES modules
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// Set EJS as the templating engine
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Serve static files from the public folder
app.use(express.static(path.join(__dirname, 'public')));
```

```
// Middleware to parse JSON and URL-encoded data
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Connect to MongoDB Atlas
mongoose.connect('<your-mongodb-atlas-connection-string>', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('Connected to MongoDB Atlas'))
.catch(err => console.error('MongoDB Atlas connection error:', err));

// Home route - render the homepage
app.get('/', (req, res) => {
  res.render('index', { title: 'Home' });
});

// Mount products route
app.use('/products', productsRouter);

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Step 4: Create Mongoose Models

1. **Product Model (models/Product.js)**

```
import mongoose from 'mongoose';

const productSchema = new mongoose.Schema({
  name: { type: String, required: true },
  description: String,
  price: { type: Number, required: true },
  imageUrl: String,
});

export default mongoose.model('Product', productSchema);
```

2. **User Model (models/User.js)**

```
import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
```

```
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

export default mongoose.model('User', userSchema);
```

3. **Order Model (models/Order.js)**

```
import mongoose from 'mongoose';

const orderSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
  products: [
    {
      product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product',
required: true },
      quantity: { type: Number, default: 1 },
    },
  ],
  createdAt: { type: Date, default: Date.now },
});

export default mongoose.model('Order', orderSchema);
```

Step 5: Set Up the Products Route

1. **Create routes/products.js**

In the routes folder, create products.js with the following content:

```
import express from 'express';
import Product from '../models/Product.js';

const router = express.Router();

// GET /products – fetch and display all products
router.get('/', async (req, res) => {
  try {
    const products = await Product.find({});
    res.render('products/index', { title: 'Product Listings', products });
  } catch (err) {
    console.error(err);
    res.status(500).send('Server Error');
  }
```

```
});

export default router;
```

Step 6: Create EJS Views

1. **Header Partial (views/partials/header.ejs)**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <header>
    <h1>Store App</h1>
    <nav>
      <a href="/">Home</a>
      <a href="/products">Products</a>
    </nav>
  </header>
```

2. **Footer Partial (views/partials/footer.ejs)**

```
  <footer>
    <p>&copy; 2025 Store App</p>
  </footer>
</body>
</html>
```

3. **Home Page (views/index.ejs)**

Create index.ejs in the views folder:

```
<% include partials/header.ejs %>
<div class="container">
  <h2>Welcome to the Store App!</h2>
  <p>This is your homepage connected to MongoDB Atlas.</p>
</div>
<% include partials/footer.ejs %>
```

4. **Products Listing Page (views/products/index.ejs)**

Create index.ejs in the views/products folder:

```
<% include ../partials/header.ejs %>
<div class="container">
  <h2><%= title %></h2>
  <div class="product-grid">
    <% products.forEach(product => { %>
      <div class="product-card">
        <img src="<%= product.imageUrl %>" alt="<%= product.name %>">
        <h3><%= product.name %></h3>
        <p><%= product.description %></p>
        <p>$<%= product.price.toFixed(2) %></p>
        <a href="/products/<%= product._id %>">View Details</a>
      </div>
    <% }) %>
  </div>
</div>
<% include ../partials/footer.ejs %>
```

Step 7: Add Basic Styling

1. **Create CSS File (public/css/styles.css)**

In the public/css folder, create styles.css with the following content:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background: #f4f4f4;
}

header, footer {
  background: #333;
  color: white;
  padding: 10px;
  text-align: center;
}

.container {
  width: 90%;
  margin: 20px auto;
}
```

```
.product-grid {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.product-card {
  background: white;
  padding: 10px;
  border: 1px solid #ddd;
  flex: 1 1 200px;
  text-align: center;
}

.product-card img {
  max-width: 100%;
  height: auto;
}
```

Step 8: Run and Test Your Application

1. **Start the Server**

In your terminal, run:

```
node app.js
```

2. **Test in the Browser**

• Visit http://localhost:3000 to see the homepage.

• Visit http://localhost:3000/products to view the dynamic product listings (once you have product data in your MongoDB Atlas database).

This complete guide sets up your project with an external MongoDB Atlas connection, defines the data models, builds the Express server with ES modules, creates the necessary routes, and sets up dynamic EJS views with basic styling. Follow each step carefully, and if you encounter any issues or need clarification, feel free to ask. Happy coding!