



# Hands-On Backend Development with MERN Stack

Author: Pi

Date: 2025.07.02

# CONTENTS

- 1. Welcome to the backend development Workshop
- 3. Setting Up the Environment
- 5. Express.js - Building the Backend
- 7. MongoDB and Mongoose
- 9. Middleware in Express
- 11. Bcrypt for Password Hashing
- 13. CORS for Cross-Origin Requests
- 15. Local Storage, Cookies, Headers
- 17. Hands-On Project
- 2. What is the MERN Stack?
- 4. Project Folder Structure
- 6. Async/Await in Node.js
- 8. Environment Variables with .env
- 10. Zod Validation
- 12. Axios for API Requests
- 14. Postman for API Testing
- 16. Dev Tools / Inspect
- 18. Wrap-Up and Q&A



01

*Welcome to the backend development  
Workshop*

---



# Welcome to the backend development Workshop

## Objective

Learn to build a backend using the MERN stack

## Agenda

- MERN Stack Overview
- Node.js, Express, MongoDB, Mongoose
- Tools: npm, Postman, Dev Tools
- Security: Middleware, Bcrypt, CORS
- Data Handling: Async/Await, Axios, Zod
- Storage: Local Storage, Cookies, Headers

02

## *What is the MERN Stack?*

---

# What is the MERN Stack?

## MERN

- MongoDB: NoSQL database for storing JSON-like data
- Express: Node.js framework for building RESTful APIs
- React: Frontend library (not covered in depth today)
- Node.js: JavaScript runtime for server-side development

## Why MERN?

- Full JavaScript ecosystem
- Scalable and flexible for modern apps



03

## *Setting Up the Environment*

---

# Setting Up the Environment

## Node.js & npm

- npm: Node Package Manager, installs dependencies
- Initialize project: npm init -y

## package.json

- Stores project metadata and dependencies

```
{ "name": "mern-workshop", "version": "1.0.0",  
"dependencies": { "express": "^4.18.2" } }
```

04

## *Project Folder Structure*

---

# Project Folder Structure

Recommended Structure:

```
/mern-workshop |---/node_modules |---/controllers |---/models |---/routes |---/middleware  
|---.env |---.gitignore |---index.js |---package.json
```

## Purpose

- controllers: Business logic
- models: Database schemas
- routes: API endpoints
- middleware: Request processing

05

## *Express.js - Building the Backend*

---

# Express.js - Building the Backend

## 1 What is Express?

- Lightweight framework for Node.js
- Simplifies routing and middleware

## 2 Basic Boilerplate

```
const express = require('express'); const app = express(); app.use(express.json()); app.get('/', (req, res) => { res.send('Hello, MERN Workshop!'); }); app.listen(3000, () => { console.log('Server running on http://localhost:3000'); });
```

The background of the slide is a dark, moody landscape. In the foreground, there are large, undulating sand dunes. Behind them, a range of mountains is visible against a dark sky. The overall tone is mysterious and calm.

06

## *Async/Await in Node.js*

---

# Async/Await in Node.js

## Why Async/Await?

- Simplifies asynchronous code (e.g., database queries)
- Replaces callbacks and promises

## Example

```
async function fetchData() { try { const data =  
    await someAsyncOperation(); return data; }  
catch (error) { console.error('Error:', error); }}
```

07

## *MongoDB and Mongoose*

---

# MongoDB and Mongoose

## 1 MongoDB

NoSQL database; stores data  
in JSON-like documents

## 2 Mongoose

ODM (Object Data Modeling)  
for MongoDB

## 3 Example Schema

```
const mongoose = require('mongoose'); const userSchema = new mongoose.Schema({  
  name: String, email:  
  type: String, unique: true } }); const User = mongoose.model('User', userSchema);
```

## Connection

```
mongoose.connect('mongodb://localhost:27017/mern-workshop');
```

08

## *Environment Variables with .env*

---

# Environment Variables with .env



## Purpose

Store sensitive data (e.g., DB URI, API keys)



## Setup

- Install: npm install dotenv
- Create .env:

MONGODB\_URI=mongodb://localhost:27017/mern-workshop PORT=3000

## Use in code

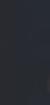
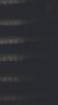
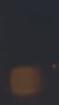
```
require('dotenv').config(); console.log(process.env.MONGODB_URI);
```

.gitignore

node\_modules/.env

09

## *Middleware in Express*



# Middleware in Express

## What is Middleware?

- Functions that process requests before routes
- Examples: Logging, authentication

## Example

```
const logger = (req, res, next) => {  
  console.log(` ${req.method} ${req.url}`);  next();};  
app.use(logger);
```

10

*Zod Validation*

---

# Zod Validation

1

## Purpose

Validate incoming data

2

## Setup

npm install zod

3

## Example

```
const { z } = require('zod'); const userSchema = z.object({ name: z.string().min(3), email: z.string().email() });
app.post('/users', (req, res) => {
  try {
    userSchema.parse(req.body);
    res.status(201).send('Valid data');
  } catch (error) {
    res.status(400).send(error.errors);
  }
});
```

11

## *Bcrypt for Password Hashing*

---

# Bcrypt for Password Hashing

1

## Purpose

Securely hash passwords

2

## Setup

npm install bcrypt

3

## Example

```
const bcrypt = require('bcrypt');  
async function hashPassword(password) {  
  const salt = await bcrypt.genSalt(10);  
  return await bcrypt.hash(password, salt);}  
  
async function verifyPassword(password, hash) {  
  return await bcrypt.compare(password, hash);}
```

12

## *Axios for API Requests*

---

# Axios for API Requests

1

## Purpose

Make HTTP requests from  
Node.js or frontend

2

## Setup

npm install axios

3

## Example

```
const axios = require('axios');  
async function fetchUsers() {  
  try {  
    const response = await  
    axios.get('http://localhost:3000/users');  
    console.log(response.data);  
  } catch (error) {  
    console.error('Error:',  
    error);  
  }  
}
```

13

## *CORS for Cross-Origin Requests*

---

# CORS for Cross-Origin Requests

Purpose

Allow frontend to access backend

Setup

npm install cors

Example

```
const cors = require('cors'); app.use(cors({ origin:  
  'http://localhost:3000' }));
```

14

## *Postman for API Testing*

---

# Postman for API Testing



## What is Postman?

- Tool to test API endpoints  
(GET, POST, etc.)



## Key Features

- Send requests and view responses
- Save collections for repeated testing



## Example

- Create a POST request to <http://localhost:3000/users>
- Body: { "name": "John",  
"email": "john@example.com" }

15

# *Local Storage, Cookies, Headers*



# LocalStorage, Cookies, Headers

## 1 Local Storage

- Browser storage for key-value pairs
- Example: `localStorage.setItem('token', 'abc123');`

```
const cookieParser = require('cookie-parser'); app.use(cookieParser()); app.get('/set-cookie', (req, res) => { res.cookie('session', '12345', { maxAge: 900000 }); res.send('Cookie set'); });
```

## Headers

- Send metadata (e.g., Authorization)

```
app.get('/protected', (req, res) => { const token = req.headers['authorization']; res.send(`Token: ${token}`); });
```

## 2 Cookies

- Store data sent with requests
- Setup: `npm install cookie-parser`

The background of the slide features a dark, circular staircase with many steps, viewed from above. The stairs are arranged in a radial pattern, creating a sense of depth and perspective. The lighting is dramatic, with the steps appearing in shades of brown and tan against a dark background.

16

*Dev Tools / Inspect*

---

# Dev Tools / Inspect



## Purpose

Debug backend and frontend



## Key Features

- Network tab: Monitor API requests
- Console: View logs and errors



## Example

- Open Chrome DevTools (F12)  
> Network > Test API call

A dark blue background featuring a network of glowing yellow nodes connected by white lines, resembling a star map or a complex web.

17

## *Hands-On Project*

---

# Hands-On Project

## Task

Build a simple User API

- Endpoints:
  - POST /users: Create user (validate with Zod, hash password with Bcrypt)
  - GET /users: List all users
- Use MongoDB/Mongoose for storage
- Test with Postman

## Steps

- Set up Express server
- Create User model with Mongoose
- Add routes and middleware
- Test endpoints

## Deliverable

Working API with 2 endpoints

18

*Wrap-Up and Q&A*

---

# Wrap-Up and Q&A

## Key Takeaways

- MERN stack for full JavaScript development
- Tools like npm, Postman, and DevTools streamline development
- Security with Bcrypt, CORS, and middleware

## Next Steps

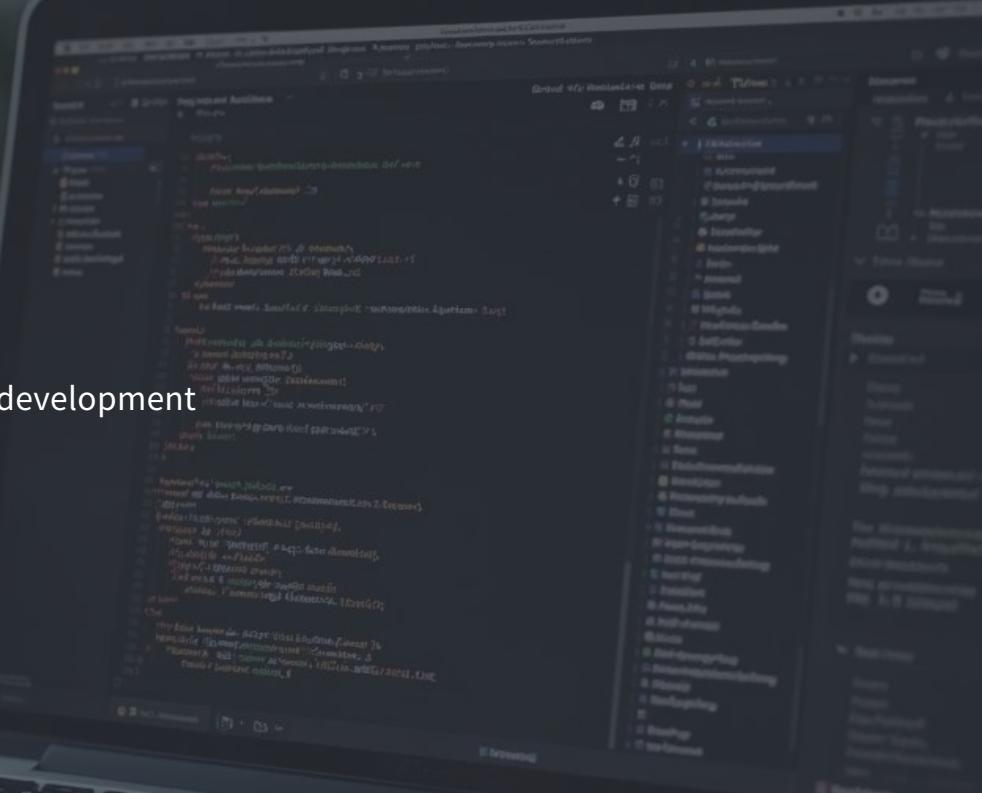
- Deploy backend to platforms like Render or Vercel

## Q&A

Ask away!

## Visuals

Use minimal text, large fonts, and code blocks with syntax highlighting. Include a diagram for folder structure (Slide 4) and a screenshot of Postman (Slide 14).



```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');

// Load environment variables from .env file
dotenv.config();

// Initialize Express
const app = express();
app.use(cors());
app.use(express.json());

// Connect to MongoDB
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('MongoDB connected!'))
  .catch(error => console.error(`MongoDB connection error: ${error}`));

// Set up routes
const userRoutes = require('./routes/User');
const authRoutes = require('./routes/Auth');

app.use('/api/users', userRoutes);
app.use('/api/auth', authRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

# Thank You