



PPG INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)
Recognized by UGC | Accredited by NAAC with A | ISO Certified Institution
NH 209, Sathy Main Road, Saravanampatti, Coimbatore – 641035, Tamil Nadu



NM1087 - MERN Stack

Department of

COMPUTER SCIENCE AND ENGINEERING



NM1087 - Full-Stack Web Development with the MERN Stack

NAME:..... **ROLL NO:**

SEMESTER: **BRANCH:**

.....

Certified bonafide record of work done by.....

Place: COIMBATORE

Date:

Staff In-Charge

Head of the Department

University Register Number:

Submitted for the University Practical Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAM



Introduction to MERN Stack

What is MERN Stack?

MERN Stack is a powerful combination of technologies used for building modern web applications. It consists of:

- **MongoDB:** A NoSQL document database that stores data in flexible, JSON-like documents
- **Express.js:** A minimal and flexible Node.js web application framework
- **React.js:** A JavaScript library for building user interfaces, particularly single-page applications
- **Node.js:** A JavaScript runtime built on Chrome's V8 JavaScript engine

Advantages of MERN Stack

The MERN stack offers several compelling advantages for web development:

Single Language Development: All components use JavaScript, enabling developers to work across the entire application stack with one language. This reduces context switching and allows for better code reuse between frontend and backend.

Rich Ecosystem: Each technology in the MERN stack has a vast ecosystem of libraries, tools, and community support. NPM provides access to hundreds of thousands of packages that can accelerate development.

Rapid Development: The component-based architecture of React combined with the flexibility of Express.js allows for rapid prototyping and development cycles.

Scalability: MongoDB's horizontal scaling capabilities, combined with Node.js's event-driven architecture, make MERN applications highly scalable.

Real-time Applications: The stack is particularly well-suited for real-time applications like chat systems, live updates, and collaborative tools.

Architecture Overview

A typical MERN application follows a three-tier architecture:

Presentation Layer (React): Handles user interface, user interactions, and presentation logic. Components manage their own state and communicate with the backend through HTTP requests.



Application Layer (Node.js + Express): Manages business logic, authentication, authorization, and API endpoints. It serves as an intermediary between the frontend and database.

Data Layer (MongoDB): Stores and manages application data. Provides flexible schema design and powerful querying capabilities.

Environment Setup and Prerequisites

System Requirements

Before starting MERN stack development, ensure your system meets these requirements:

- **Operating System:** Windows 10/11, macOS 10.14+, or Linux (Ubuntu 18.04+ recommended)
- **RAM:** Minimum 8GB (16GB recommended for smooth development)
- **Storage:** At least 10GB free space for development tools and project files
- **Internet Connection:** Stable connection for downloading packages and accessing cloud services

Required Software Installation

Node.js and NPM

Node.js is the foundation of our backend development. Download and install the latest LTS version from the official website.

Installation Steps:

1. Visit <https://nodejs.org/en/download/>
2. Download the LTS version for your operating system
3. Run the installer with default settings
4. Verify installation by opening terminal/command prompt and running:



bash

node --version

npm --version

Version Management: Consider using Node Version Manager (nvm) for managing multiple Node.js versions on the same machine. This is particularly useful when working on multiple projects with different Node.js requirements.

Visual Studio Code

VS Code is the recommended IDE for MERN stack development due to its excellent JavaScript support and extensive extension ecosystem.

Installation Steps:

1. Visit <https://code.visualstudio.com/download>
2. Download the appropriate version for your OS
3. Install with default settings

Essential Extensions:

- ES7+ React/Redux/React-Native snippets
- Auto Rename Tag
- Bracket Pair Colorizer
- GitLens
- Prettier - Code formatter
- ESLint
- Thunder Client (for API testing)
- MongoDB for VS Code

MongoDB Setup

MongoDB can be installed locally or used through cloud services like MongoDB Atlas.



Local Installation:

1. Visit <https://www.mongodb.com/try/download/community>
2. Download MongoDB Community Server
3. Install with default settings
4. Start MongoDB service

MongoDB Atlas Setup (Recommended for beginners):

1. Create account at <https://www.mongodb.com/atlas>
2. Create a new cluster (free tier available)
3. Configure network access and database user
4. Get connection string for application

Development Tools Setup

Git Configuration

Version control is essential for any development project. Configure Git with your credentials:

```
bash
```

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your.email@example.com"
```

Package Manager Selection

While NPM comes with Node.js, consider using Yarn for faster package installation and better dependency management:

```
bash
```

```
npm install -g yarn
```



Project Structure and Initial Setup

Creating the Project Structure

A well-organized project structure is crucial for maintainability and scalability. Create the following directory structure:

mern-project/

- |— client/ # React frontend application
- |— server/ # Node.js backend application
- |— README.md # Project documentation
- |— .gitignore # Git ignore rules
- |— package.json # Root package configuration

Frontend Setup with Vite

Modern React development benefits from faster build tools. Vite provides significantly faster development server startup and hot module replacement compared to Create React App.

Creating React Application:

```
bash
mkdir mern-project && cd mern-project
npm create vite@latest client -- --template react
cd client
npm install
```

Additional Frontend Dependencies:

```
bash
npm install bootstrap axios react-router-dom
npm install -D eslint prettier
```



Package Explanations:

- **bootstrap**: CSS framework for responsive design
- **axios**: HTTP client for API requests
- **react-router-dom**: Client-side routing for single-page applications
- **eslint**: Code linting for maintaining code quality
- **prettier**: Code formatting for consistent style

Backend Setup with Express

The backend server handles API requests, business logic, and data processing.

Creating Express Server:

```
bash
```

```
cd .. # Return to project root
```

```
mkdir server && cd server
```

```
npm init -y
```

```
npm install express mongoose cors dotenv
```

```
npm install -D nodemon
```

Package Explanations:

- **express**: Fast, unopinionated web framework for Node.js
- **mongoose**: MongoDB object modeling for Node.js
- **cors**: Cross-Origin Resource Sharing middleware
- **dotenv**: Loads environment variables from .env file
- **nodemon**: Automatically restarts server during development

Environment Configuration

Create environment files for secure configuration management:



server/.env:

PORT=3001

MONGODB_URI=mongodb://localhost:27017/mern-crud

NODE_ENV=development

server/.env.example:

PORT=3001

MONGODB_URI=your_mongodb_connection_string

NODE_ENV=development

Package.json Scripts Configuration

server/package.json scripts:

json

```
{  
  "scripts": {  
    "start": "node index.js",  
    "dev": "nodemon index.js",  
    "test": "echo \"Error: no test specified\" && exit 1"  
  }  
}
```



client/package.json scripts (usually pre-configured by Vite):

json

```
{  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview"  
  }  
}
```

MERN Stack Web Application

Aim:

To develop a full-stack web application using the MERN stack (MongoDB, Express.js, React.js, Node.js) that provides a responsive frontend, a robust backend API, and efficient data storage and retrieval functionalities, ensuring user-friendly interaction and seamless performance across platforms.

Algorithm for Developing a MERN Stack Web Application:

1. Requirement Analysis and Planning

- Identify the purpose, target users, and core features of the application.
- Design UI/UX wireframes and database schema.

2. Set Up Development Environment

- Install Node.js, npm, and MongoDB.
- Set up version control (Git + GitHub).

3. Initialize Backend with Node.js and Express.js

- Create a new project using npm init.
- Set up Express server and configure middleware (e.g., body-parser, CORS).
- Define RESTful API routes (CRUD operations).



- Connect to MongoDB using Mongoose.

4. Design the MongoDB Database

- Define Mongoose schemas and models for each data entity.
- Establish relationships (if any) between collections.

5. Develop Frontend Using React.js

- Initialize React app using create-react-app.
- Structure components based on the application layout.
- Use React Router for navigation.
- Use Axios or Fetch API to connect frontend to backend APIs.

6. Integrate Frontend and Backend

- Test API calls from React to Express.
- Handle JSON responses and display dynamic content.
- Implement form handling, data submission, and validation.

7. Authentication & Authorization (Optional but Recommended)

- Implement user authentication using JWT (JSON Web Tokens).
- Set up protected routes on frontend and backend.

8. Testing and Debugging

- Perform unit testing (Jest, Mocha) and manual testing of UI/API.
- Use Postman to test backend routes.

9. Deploy the Application

- Host frontend on platforms like Vercel or Netlify.
- Host backend and database on platforms like Render, Heroku, or MongoDB Atlas.

10. Maintenance and Updates

- Monitor logs and usage.
- Fix bugs and add new features based on user feedback



Program:

Frontend

```
import React from 'react';

import Login from './components/Login';

import Signup from './components/Signup';

import BlogList from './components/BlogList';

import './App.css';

const App = () => {

  return (

    <div>

      <h1>Blog Application</h1>

      <Login />

      <Signup />

      <BlogList />

    </div>

  );

};

export default App;

import React, { useEffect, useState } from 'react';

import './styles/BlogList.css';

const BlogList = () => {

  const [blogs, setBlogs] = useState([]);
```



```
useEffect(() => {

  const fetchBlogs = async () => {

    try {

      const response = await fetch('http://localhost:5000/api/blog/all');

      const data = await response.json();

      setBlogs(data);

    } catch (error) {

      console.error('Error fetching blogs:', error);

    }

  };

  fetchBlogs();

}, []);

return (

  <div className="blog-list">

    <h2>Blogs</h2>

    <div className="blog-cards">

      {blogs.map((blog) => (

        <div key={blog._id} className="blog-card">

          <h3>{blog.topic}</h3>

          <p>{blog.descp}</p>

          <small>By: {blog.username}</small>

        </div>

      ))}

    </div>

  )}
```



</div>

);

};

export default BlogList;

Backend

```
const express = require('express');
```

```
const cors = require('cors');
```

```
const mongoose = require('mongoose');
```

```
const userRoutes = require('./Routes/userRoutes');
```

```
const blogRoutes = require('./Routes/blogRoutes');
```

```
const app = express();
```

```
const PORT = 5000;
```

```
// Middleware
```

```
app.use(express.json()); // <-- Ensures JSON bodies are parsed
```

```
app.use(cors());
```

```
// Routes
```

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

```
app.use('/api/blog', blogRoutes);
```

```
// MongoDB Connection
```

```
mongoose.connect('mongodb://localhost:27017/yourDatabaseName', {
```

```
  useUrlParser: true,
```

```
  useUnifiedTopology: true,
```



```
})
```

```
.then(() => console.log("Connected to MongoDB"))
```

```
.catch(err => console.error("MongoDB connection error:", err));
```

```
// Start Server
```

```
app.listen(PORT, () => console.log(Server running on http://localhost:${PORT}));
```

Output:

Landing page

Welcome to my Blog

Welcome to my blog! Here you'll find a wide range of articles, tutorials, and resources designed to help you grow as a developer. Whether you're interested in web development, software engineering, programming languages, or best practices in the tech industry, there's something here for everyone. Dive in and explore the content to expand your knowledge and skills.

[View all posts](#)

Want to learn HTML, CSS and JavaScript by building fun and engaging projects?

Check our 100 js projects website and start building your own projects

100 JS Projects Website

Top JS Project Ideas

For Beginners



Blog page

Recent Posts



Result:

A responsive MERN stack web application was successfully developed with seamless frontend-backend integration. It ensures efficient data handling, user-friendly interaction, and cross-platform performance.