# MERN Stack Todo Application - Project Documentation

**Project Title:** MERN Stack Todo Application Deployment on AWS EC2

**Author:** Sally Munga

**Date:** 28/08/2025

**Project Repository:** [StegHub\_DevOps/project2 at main · 123Origami/StegHub\_DevOps](https://github.com/123Origami/StegHub_DevOps/tree/main/project2)

**LinkedIn URL: https://www.linkedin.com/posts/sally-m-09546a245\_mernstack-fullstackdevelopment-aws-activity-7366896403454889986-Jw8j?utm\_source=share&utm\_medium=member\_desktop&rcm=ACoAADzCKfEBZK3LPTQ5D0pi879scjlAX0g4FTk**

## Project Overview

This project involved the complete development and deployment of a full-stack MERN (MongoDB, Express.js, React, Node.js) Todo application on an AWS EC2 instance. The application provides full CRUD (Create, Read, Update, Delete) functionality through a responsive web interface, demonstrating end-to-end web development capabilities.

## Technical Architecture

Frontend: React.js with functional components  
Backend: Node.js with Express.js framework  
Database: MongoDB Atlas (Cloud Database)  
Infrastructure: AWS EC2 Ubuntu 22.04 LTS instance  
Development Tools: Postman (API testing), Concurrently, Nodemon

## Key Features Implemented

User Interface  
 - Responsive design that works on both desktop and mobile devices  
 - Clean, modern interface with intuitive task management  
 - Real-time updates without page refresh

Backend API  
 - RESTful API with complete CRUD endpoints  
 - Secure database connections using environment variables  
 - CORS configuration for cross-origin requests  
 - Error handling and validation

Database Management  
 - MongoDB Atlas cloud database implementation  
 - Secure connection strings with proper authentication  
 - Data modeling with Mongoose ODM

Deployment Infrastructure  
 - AWS EC2 instance configuration  
 - Security group management for port access  
 - Environment setup and dependency management

## Technical Implementation Details

Backend Development:  
- Configured Express.js server with middleware handling  
- Implemented route controllers for todo operations  
- Established MongoDB connection with Mongoose  
- Environment variable configuration for security  
  
Frontend Development:  
- React component architecture with state management  
- Axios for API communication between frontend and backend  
- CSS styling with responsive design principles  
- Proxy configuration for seamless API calls during development

## Security Measures Implemented

1. Environment variables for sensitive data  
2. MongoDB Atlas network access restrictions  
3. AWS Security Group configuration  
4. Input validation and error handling  
5. CORS policy implementation

## Development Challenges & Solutions

Challenge 1: Database connection issues  
Solution: Proper configuration of MongoDB Atlas whitelist IP addresses and connection string parameters  
  
Challenge 2: Cross-origin requests between frontend and backend  
Solution: Implemented CORS headers in Express.js and proxy configuration in React  
  
Challenge 3: Concurrent development of frontend and backend  
Solution: Used Concurrently to run both servers simultaneously during development

## Project Highlights

✅ Full-stack application development  
✅ Cloud database implementation  
✅ RESTful API design and implementation  
✅ Responsive frontend interface  
✅ AWS infrastructure management  
✅ Professional development workflow  
✅ Comprehensive testing procedures

## Skills Demonstrated

Frontend Development: React.js, JavaScript, CSS, HTML5  
Backend Development: Node.js, Express.js, REST API design  
Database Management: MongoDB, Mongoose ODM, database design  
DevOps & Deployment: AWS EC2, Linux server management, security configuration  
Development Tools: Git, Postman, NPM, Concurrent development

## Future Enhancements

1. User authentication and authorization  
2. Task categories and tags  
3. Due dates and reminders  
4. Drag-and-drop interface  
5. Mobile application version  
6. Real-time collaboration features

## Conclusion

This project successfully demonstrates the complete lifecycle of a modern web application, from initial setup to deployment. The MERN stack implementation shows proficiency in both frontend and backend development, while the AWS deployment illustrates cloud infrastructure management capabilities. The application serves as a foundation for more complex web applications and showcases professional-grade development practices.