MEAN Stack Architecture Document

1. Introduction

Application Name: MyMeanApp

Version: 1.0.0

Description: A web-based application for task management and collaboration, built using MongoDB, Express.js, Angular, and Node.js.

---

2. System Overview

The MEAN stack architecture leverages MongoDB for data storage, Express.js and Node.js for server-side logic, and Angular for the client-side interface. This document describes the system architecture, including the application layers, interactions, and deployment strategy.

---

3. Architecture Diagram

Architecture Diagram:

![Architecture Diagram](https://via.placeholder.com/800x400.png?text=Architecture+Diagram)

---

4. Architecture Components

4.1. Frontend (Angular)

Responsibilities:

- **User Interface:** Provides the web interface for users to interact with the application.
- **Client-Side Logic:** Handles user interactions, forms, and data presentation.
- **API Communication:** Sends HTTP requests to the backend API to fetch or send data.

Technologies:

- **Framework:** Angular 15
- **State Management:** NgRx or Angular Services
- **Routing:** Angular Router
- **Build Tool:** Angular CLI

4.2. Backend (Node.js and Express.js)

Responsibilities:

- **API Handling:** Manages HTTP requests from the frontend, processes them, and returns responses.
- **Business Logic:** Implements core application logic and workflows.
- **Middleware:** Handles authentication, logging, error handling, and other tasks.

Technologies:

- **Framework:** Express.js
- **Server:** Node.js
- **Authentication:** JWT (JSON Web Tokens)
- **Middleware:** Custom and third-party middleware for security, logging, etc.

4.3. Database (MongoDB)

Responsibilities:

- **Data Storage:** Stores application data such as user profiles, tasks, and projects.
- **Data Retrieval:** Supports complex queries and indexing for efficient data access.
- **Backup and Recovery:** Ensures data integrity and availability through regular backups.

Technologies:

- **Database:** MongoDB 5.0
- **ODM:** Mongoose (for schema definition and interaction)

4.4. Web Server (Nginx)

Responsibilities:

- **Reverse Proxy:** Routes incoming requests to the appropriate backend service.
- **Load Balancing:** Distributes incoming traffic across multiple instances of the backend service.
- **SSL Termination:** Handles HTTPS connections and encrypts data in transit.

Technologies:

- **Web Server:** Nginx
- **Configuration:** Proxying, load balancing, and SSL configuration

---

5. Deployment Architecture

5.1. Development Environment

- **Server:** dev-server.mymeanapp.com

- **IP Address:** 192.168.1.10
- **Services:**
  - **Frontend:** Angular app served by Nginx
  - **Backend:** Node.js/Express.js API
  - **Database:** MongoDB instance

5.2. Staging Environment

- **Server:** staging-server.mymeanapp.com
- **IP Address:** 192.168.1.11
- **Services:**
  - **Frontend:** Angular app served by Nginx
  - **Backend:** Node.js/Express.js API
  - **Database:** MongoDB instance

5.3. Production Environment

- **Server:** prod-server.mymeanapp.com
- **IP Address:** 192.168.1.12
- **Services:**
  - **Frontend:** Angular app served by Nginx
  - **Backend:** Node.js/Express.js API
  - **Database:** MongoDB instance

Deployment Strategy:

- **Continuous Integration:** Automated builds and testing through CI/CD pipelines.
- **Continuous Deployment:** Automated deployment to staging and production environments.
- **Rollback Strategy:** Versioned deployments with the ability to roll back to previous versions.

---

6. Security

6.1. Authentication

- **JWT Tokens:** Used for securing API endpoints and managing user sessions.
- **OAuth:** Optionally used for third-party authentication (e.g., Google, Facebook).

6.2. Authorization

- **Role-Based Access Control (RBAC):** Manages user permissions and access levels.

6.3. Data Protection

- **Encryption:** Data is encrypted in transit using TLS/SSL.
- **Data Storage:** Sensitive data is encrypted at rest.

6.4. Security Best Practices

- **Input Validation:** Ensures data integrity and protects against injection attacks.
- **Rate Limiting:** Prevents abuse of the API by limiting the number of requests from a single source.
- **Logging and Monitoring:** Monitors for suspicious activity and logs important events.

---

7. Performance and Scalability

7.1. Load Balancing

- **Nginx:** Used for distributing traffic across multiple backend instances.

7.2. Caching

- **Cache Strategy:** Implemented at multiple layers, including HTTP caching, in-memory caching (e.g., Redis), and database caching.

7.3. Horizontal Scaling

- **Frontend:** Angular app can be scaled horizontally by deploying multiple instances behind a load balancer.
- **Backend:** Node.js instances can be scaled horizontally to handle increased load.
- **Database:** MongoDB supports sharding for distributing data across multiple servers.

---

8. Monitoring and Logging

8.1. Monitoring

- **Tools:** Prometheus, Grafana, or similar tools for real-time monitoring and alerting.

8.2. Logging

- **Logs:** Collected from frontend, backend, and database.
- **Log Management:** Centralized logging using tools like ELK Stack (Elasticsearch, Logstash, Kibana) or similar solutions.

8.3. Incident Management

- **Incident Response Plan:** Procedures for handling and resolving incidents, including escalation paths and communication plans.

---

9. Development and Testing

9.1. Development Workflow

- **Version Control:** Git-based workflow with branching strategies (e.g., Git Flow).
- **Code Reviews:** Peer reviews and automated code quality checks.

9.2. Testing

- **Unit Testing:** For individual components and services.
- **Integration Testing:** To test interactions between components.
- **End-to-End Testing:** Simulates real user scenarios to validate the complete application workflow.

---

10. Conclusion

This document provides a high-level overview of the MEAN stack architecture for MyMeanApp. It outlines the system components, deployment strategies, security measures, and performance considerations. For further details or specific implementation guidelines, please refer to the respective sections or consult with the development and operations teams.

End of Document

Feel free to adjust and expand on this document based on the specifics of your application and any additional requirements you may have.