M-BOP SYSTEM ARCHITECTURE

Mini Business Operations Platform

Technical Documentation

Enterprise-Grade Full-Stack Application

React 19 • Node.js • MongoDB • Express.js • JWT Authentication

---

EXECUTIVE SUMMARY

M-BOP (Mini Business Operations Platform) is a comprehensive full-stack web application engineered for streamlined business operations management with enterprise-level role-based access control. The system implements a modern three-tier architecture ensuring clear separation of concerns across presentation, business logic, and data persistence layers.

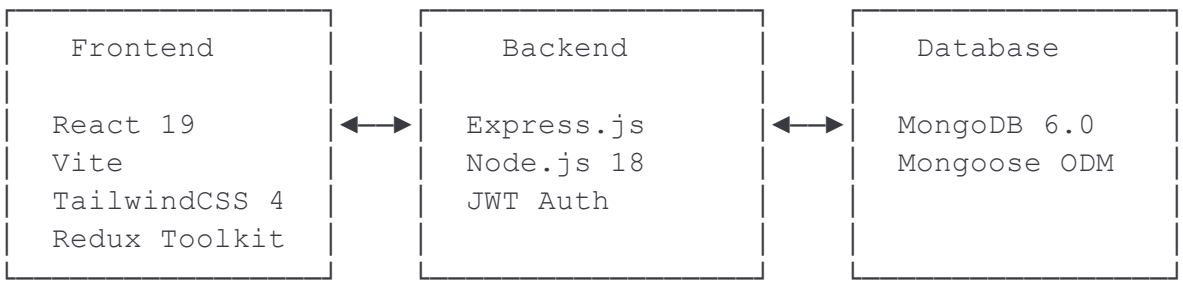
---

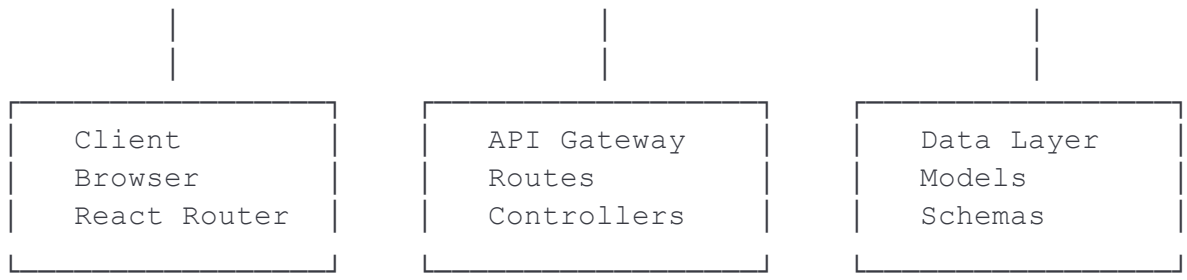
1. HIGH-LEVEL ARCHITECTURE

1.1 System Overview

The platform delivers a scalable, maintainable solution built on industry-standard technologies, providing robust functionality for managing clients, projects, tasks, and user roles with granular permission controls.

1.2 Architecture Diagram





---

2. COMPONENT ARCHITECTURE

2.1 Frontend Component Hierarchy

```
App.jsx
├── AdminLayout
│   ├── Dashboard
│   ├── ClientManagement
│   ├── ProjectManagement
│   ├── TaskManagement
│   └── UserManagement
├── StaffLayout
│   ├── StaffDashboard
│   ├── StaffProjects
│   └── StaffTasks
└── AuthLayout
    ├── Login
    └── Register
```

2.2 Backend Service Architecture

```
src/
├── config/          # Database
├── controllers/     # Business logic handlers
├── middleware/      # Auth, validation
├── models/          # MongoDB schemas
├── routes/          # API endpoint definitions
└── utils/           # Helper functions
```

---

3. API STRUCTURE & DESIGN

3.1 RESTful API Endpoints

Authentication & Authorization

```
POST   /api/auth/login          # User authentication
POST   /api/auth/logout         # User logout
GET    /api/auth/getInfo         # Token validation
```

Admin Management

```
GET    /api/admin/dashboard     # Admin dashboard analytics
GET    /api/admin/clients       # List all clients
POST   /api/admin/clients       # Create new client
PUT    /api/admin/clients/:id   # Update client
DELETE /api/admin/clients/:id   # Delete client

GET    /api/admin/projects      # List all projects
POST   /api/admin/projects      # Create project
PUT    /api/admin/projects/:id  # Update project
DELETE /api/admin/projects/:id  # Delete project

GET    /api/admin/tasks         # List all tasks
POST   /api/admin/tasks         # Create task
PUT    /api/admin/tasks/:id     # Update task status
DELETE /api/admin/tasks/:id     # Delete task

GET    /api/admin/activity      # System activity logs
GET    /api/admin/staff         # Staff management
```

Staff Management

```
GET    /api/staff/dashboard     # Staff dashboard
GET    /api/staff/projects      # Staff assigned projects
PUT    /api/staff/projects/:id  # Update project status
GET    /api/staff/tasks         # Staff assigned tasks
PUT    /api/staff/tasks/:id     # Update task status
```

---

4. DATABASE SCHEMA DESIGN

4.1 Core Collections

Users Collection

javascript

```
{
  _id: ObjectId,
  name: String,
  email: String,
  password: String,
  role: {
    type: String,
    enum: ['admin', 'staff']
  },
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Clients Collection

javascript

```
{
  _id: ObjectId,
  name: String,
  company: String,
  email: String,
  phone: String,
  address: String,
  status: {
    type: String,
    enum: ['New', 'Active', 'Paused', 'Closed']
  },
  createdBy: { type: ObjectId, ref: 'User' },
  createdAt: Date,
  updatedAt: Date
}
```

Projects Collection

javascript

```
{
  _id: ObjectId,
  title: String,
  description: String,
  clientId: { type: ObjectId, ref: 'Client' },
  status: {
    type: String,
```

```
    enum: ['New', 'In Progress', 'Completed', 'Paused', 'Closed']
  },
  timeline: {
    startDate: Date,
    endDate: Date
  },
  assignedDevelopers: [{ type: ObjectId, ref: 'User' }],
  createdBy: { type: ObjectId, ref: 'User' },
  createdAt: Date,
  updatedAt: Date
}
```

Tasks Collection

javascript

```
{
  _id: ObjectId,
  title: String,
  description: String,
  projectId: { type: ObjectId, ref: 'Project' },
  assignedTo: { type: ObjectId, ref: 'User' },
  status: {
    type: String,
    enum: ['Pending', 'In Progress', 'Completed']
  },
  dueDate: Date,
  createdBy: { type: ObjectId, ref: 'User' },
  createdAt: Date,
  updatedAt: Date
}
```

ActivityLogs Collection

javascript

```
{
  _id: ObjectId,
  action: String,
  performedBy: { type: ObjectId, ref: 'User' },
  entityType: String,
  entityId: { type: ObjectId, refPath: 'entityType' },
  details: String,
  createdAt: Date,
  updatedAt: Date
}
```

4.2 Database Relationships

```
Users (1) ——— (Many) Projects (createdBy)
Users (Many) —— (Many) Projects (assignedDevelopers)
Clients (1) ——— (Many) Projects
Projects (1) —— (Many) Tasks
Users (1) ——— (Many) Tasks (assignedTo)
Users (1) ——— (Many) ActivityLogs (performedBy)
```

---

5. TECHNOLOGY STACK & JUSTIFICATION

5.1 Frontend Technology Choices

React 19 Latest stable version with improved performance and hooks

Vite Fast build tool with excellent developer experience

TailwindCSS 4 Utility-first CSS for rapid UI development

Redux Toolkit Predictable state management with RTK Query

React Router DOM Declarative routing for single-page application

Axios Promise-based HTTP client for API communication

5.2 Backend Technology Choices

Node.js 18 LTS version with stable performance and security

Express.js Minimalist web framework with middleware support

MongoDB 6.0 Document database with flexible schema

Mongoose ODM Elegant MongoDB object modeling

JWT Stateless authentication for scalable applications

bcrypt Secure password hashing algorithm

5.3 Development & Deployment

Render Platform for backend deployment

Netlify Global CDN for frontend deployment

Git/GitHub Version control and collaboration

---

6. SECURITY IMPLEMENTATION

6.1 Authentication & Authorization

JWT-based Authentication Stateless tokens with configurable expiration

Role-Based Access Control (RBAC) Admin and Staff roles with permission levels

Password Security bcrypt hashing with salt rounds

Session Management Secure HTTP-only cookies

---

7. APPLICATION FLOW & FEATURES

7.1 Role-Based Access Control

Admin Features

- Full system access and user management
- Client, project, and task management
- System analytics and reporting
- Activity log monitoring

Staff Features

- View assigned projects and tasks
- Update project and task status
- Personal dashboard view
- Limited to own assignments

CONCLUSION

The M-BOP system represents a robust, scalable solution for business operations management. Built with modern technologies and industry best practices, the platform ensures secure, efficient handling of client relationships, project workflows, and task assignments through an intuitive role-based interface.

Developed with precision and attention to enterprise-grade standards