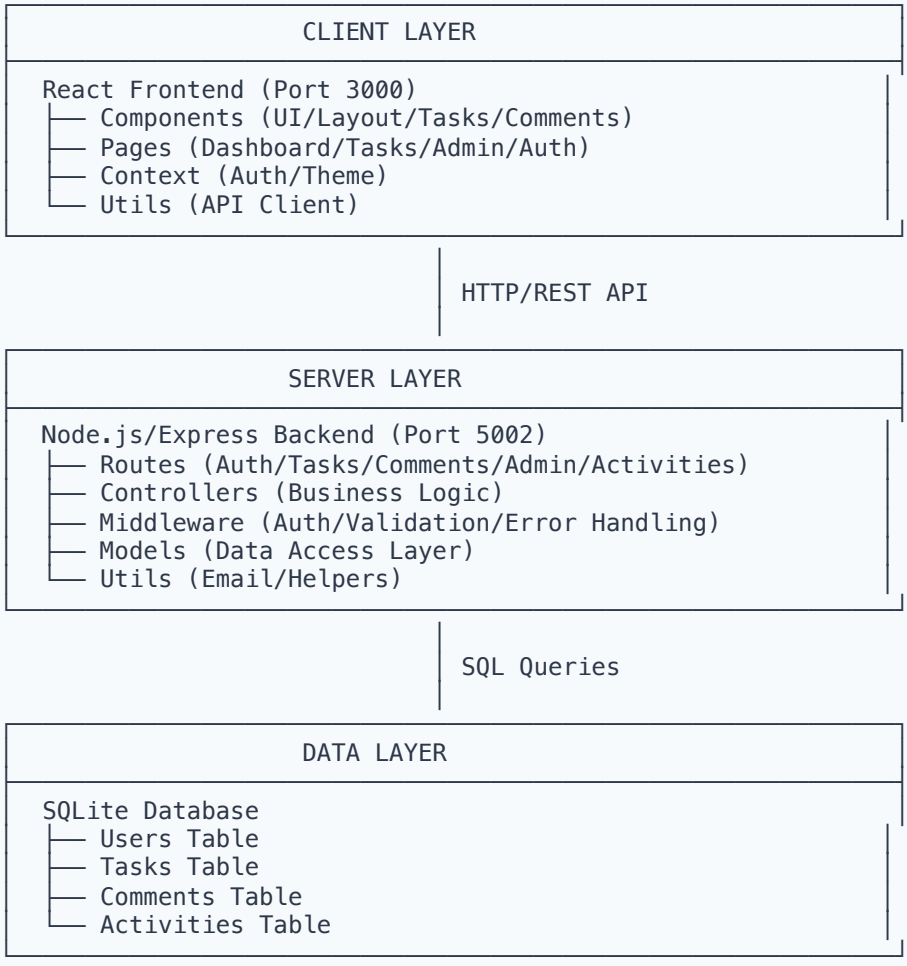
🏗️ System Architecture

MVP Task Collaboration Platform

MVP Task Collaboration Platform - Architecture Overview

🏗️ System Architecture

The MVP Task Collaboration Platform follows a modern full-stack architecture with clear separation of concerns between frontend, backend, and data layers.



🎯 Core Components

Frontend Architecture (React + TypeScript)

Technology Stack:

- React 18 with TypeScript
- React Router for navigation
- Axios for API communication
- Context API for state management

Component Structure:

```
src/
├── components/
│   ├── UI/           # Reusable UI components
│   ├── Layout/       # Header, navigation
│   ├── Tasks/        # Task-specific components
│   └── Comments/     # Comment components
├── pages/            # Route-level components
├── context/          # Global state management
├── types/            # TypeScript definitions
└── utils/            # API client & helpers
```

Key Features:

- Component-based architecture
- TypeScript for type safety
- Context API for authentication and theme
- Responsive design system
- Real-time updates via polling

Backend Architecture (Node.js + Express)

Technology Stack:

- Node.js with Express framework
- JWT for authentication
- bcryptjs for password hashing

- express-validator for input validation
- Nodemailer for email functionality

Layer Structure:

```
src/
├── routes/           # API endpoint definitions
├── controllers/      # Business logic handlers
├── middleware/       # Auth, validation, error handling
├── models/           # Data access layer
├── utils/            # Helper functions
└── db/               # Database configuration
```

Key Features:

- RESTful API design
- JWT-based authentication
- Role-based access control
- Input validation and sanitization
- Comprehensive error handling

Database Architecture (SQLite)

Schema Design:

```
Users Table:
├── id (Primary Key)
├── name
├── email (Unique)
├── password (Hashed)
├── role (user/admin)
├── resetPasswordToken
├── resetPasswordExpire
└── timestamps
Tasks Table:
├── id (Primary Key)
├── title
├── description
├── status (Todo/In Progress/Done)
├── priority (Low/Medium/High)
├── createdBy (Foreign Key → Users)
├── assignedTo (Foreign Key → Users)
└── timestamps
Comments Table:
├── id (Primary Key)
├── taskId (Foreign Key → Tasks)
├── userId (Foreign Key → Users)
├── content
└── timestamps
Activities Table:
├── id (Primary Key)
├── type (task_created/task_updated/comment_added)
├── description
├── userId (Foreign Key → Users)
├── taskId (Foreign Key → Tasks)
└── timestamp
```

🔄 Data Flow

Authentication Flow

- User Registration/Login

```
├── Frontend sends credentials
├── Backend validates & hashes password
├── JWT token generated
└── Token stored in frontend context
```

  - Protected Requests

```
├── Frontend includes JWT in headers
├── Backend middleware validates token
├── User data attached to request
└── Route handler processes request
```

Task Management Flow

```
• Task Creation
    ├── User submits task form
    ├── Frontend validates input
    ├── API request to backend
    ├── Backend validates & saves to DB
    ├── Activity log created
    └── Response sent to frontend
  • Real-time Updates
      ├── Frontend polls API every 5 seconds
      ├── Backend returns latest data
      ├── Frontend compares with current state
      └── UI updates if changes detected
```

Security Architecture

Authentication & Authorization

- **JWT Tokens:** Stateless authentication with 7-day expiration
- **Password Security:** bcryptjs hashing with salt rounds
- **Role-based Access:** User/Admin role separation
- **Protected Routes:** Middleware-based route protection

Input Validation

- **Frontend Validation:** Real-time form validation
- **Backend Validation:** express-validator for all endpoints
- **Data Sanitization:** Input trimming and cleaning
- **SQL Injection Prevention:** Parameterized queries

Security Headers & CORS

- **CORS Configuration:** Restricted to localhost origins
- **Input Limits:** JSON payload size limits
- **Error Handling:** Sanitized error responses

📊 Performance Considerations

Frontend Optimization

- **Component Memoization:** Prevent unnecessary re-renders
- **Lazy Loading:** Code splitting for better performance
- **Efficient State Management:** Context API for global state
- **Optimized Polling:** 5-second intervals for real-time updates

Backend Optimization

- **Database Indexing:** Primary keys and foreign keys
- **Pagination:** Limit data transfer for large datasets
- **Caching Strategy:** Ready for Redis implementation
- **Connection Pooling:** SQLite connection management

🔧 Development Patterns

Frontend Patterns

- **Component Composition:** Reusable UI components
- **Custom Hooks:** Shared logic extraction
- **Context Providers:** Global state management
- **TypeScript Interfaces:** Type safety and documentation

Backend Patterns

- **MVC Architecture:** Model-View-Controller separation
- **Middleware Chain:** Request processing pipeline
- **Repository Pattern:** Data access abstraction
- **Error Handling:** Centralized error management

🚀 Deployment Architecture

Development Environment

```
Frontend (localhost:3000) → Backend (localhost:5002) → SQLite DB
```

Production Considerations

```
Frontend (Static Files) → Load Balancer → Backend Instances → Database
                                                ↓
                                          Email Service
```

Scalability Options:

- **Database:** Migrate to PostgreSQL/MySQL
- **Caching:** Redis for session management
- **File Storage:** AWS S3 for attachments

- **Real-time:** WebSocket implementation
- **Monitoring:** Application performance monitoring

📈 Future Enhancements

Technical Improvements

- **WebSocket Integration:** Real-time collaboration
- **Microservices:** Service decomposition
- **Containerization:** Docker deployment
- **CI/CD Pipeline:** Automated testing and deployment

Feature Extensions

- **File Attachments:** Task file uploads
- **Notifications:** Email/Push notifications
- **Advanced Search:** Full-text search capabilities
- **Reporting:** Analytics and reporting dashboard

Architecture Version: 1.0.0
Last Updated: December 2025