Hospital Backend System Documentation

System Architecture & Design Decisions

Authentication & Security

1. **JWT Authentication**
   - Uses JSON Web Tokens for stateless authentication
   - Tokens expire after 24 hours
   - All sensitive endpoints require valid JWT in Authorization header
2. **Password Security**
   - Passwords are hashed using bcrypt with salt rounds of 10
   - No plain-text passwords stored in database
3. **End-to-End Encryption**
   - Patient notes are encrypted using AES-256
   - Each note has a unique IV (Initialization Vector)
   - Encryption key stored in environment variables
   - Only authenticated doctors and patients can decrypt notes

Database Schema

1. **User Model**
   - Supports both doctor and patient roles
   - References between doctors and patients using MongoDB relationships
   - Indexes on email for quick lookups
2. **Note Model**
   - Stores encrypted content and IV separately
   - References to both doctor and patient
   - Timestamp for tracking note history
3. **ActionableStep Model**
   - Supports both checklist items and scheduled plans
   - Tracks completion status and repeat counts

    - References to patient and original note

LLM Integration

1. **Google Gemini Implementation**
    - Uses Gemini Pro model for note analysis
    - Structured prompt for consistent output format
    - Extracts both immediate tasks and scheduled actions
2. **Processing Pipeline**
    - Raw note text is processed before encryption
    - LLM output is parsed and validated
    - Automatic scheduling of extracted actions

Scheduling System

1. **Dynamic Scheduling**
    - Uses node-schedule for task scheduling
    - Supports recurring tasks with missed task handling
    - Automatically extends schedules for missed check-ins
2. **Reminder Management**
    - Cancels existing reminders when new notes are added
    - Tracks completion status for each reminder
    - Supports flexible scheduling patterns

Testing Framework

Test Setup

1. **Framework & Libraries**
    - Jest as primary testing framework
    - Supertest for API endpoint testing
    - MongoDB Memory Server for database testing

```
npm install --save-dev jest supertest mongodb-memory-server
```

2. **Test Environment**
    - Isolated test database using MongoDB Memory Server
    - Automatic test database cleanup between tests

- Environment variable management for testing

Test Coverage

1. **Unit Tests**
    - Model validations and methods
    - Utility functions
    - Service layer functions

npm run test:unit

2. **Integration Tests**
    - API endpoints
    - Database operations
    - Authentication flows

npm run test:integration

3. **End-to-End Tests**
    - Complete user workflows
    - System integration points
    - Error handling scenarios

npm run test:e2e

Test Categories

1. **Authentication Tests**
    - User registration
    - Login functionality
    - JWT token validation
    - Password hashing verification
2. **Doctor Functionality Tests**
    - Patient list retrieval
    - Note submission
    - Actionable step creation
    - Encryption/decryption
3. **Patient Functionality Tests**

   - Doctor selection
   - Actionable step management
   - Reminder completion
   - Data access controls
4. **LLM Integration Tests**
   - Note processing
   - Action extraction
   - Error handling
   - Response formatting
5. **Scheduler Tests**
   - Reminder creation
   - Missed check-in handling
   - Schedule cancellation
   - Repeat logic

Running Tests

```
# Run all tests
npm test

# Run tests in watch mode
npm run test:watch

# Run tests with coverage report
npm run test:coverage

# Run specific test suites
npm test -- auth.test.js
npm test -- doctor.test.js
npm test -- patient.test.js
```

API Documentation

The API documentation is available through Swagger UI at /api-docs when running the server. The API is organized into three main sections:

1. **Auth Routes** (/api/auth)
   - User registration and login
   - Role-based account creation
2. **Doctor Routes** (/api/doctor)
   - Patient management
   - Note submission and processing
   - Actionable step creation
3. **Patient Routes** (/api/patient)
   - Doctor selection
   - Actionable step management
   - Progress tracking

Environment Variables

Required environment variables:

MONGODB_URI=your_mongodb_connection_string

JWT_SECRET=your_jwt_secret_key

ENCRYPTION_KEY=your_aes_encryption_key

GOOGLE_API_KEY=your_google_api_key

PORT=5000

NODE_ENV=development

Security Considerations

1. **Data Protection**
   - All sensitive data is encrypted at rest
   - Separate encryption keys for different data types
   - Regular key rotation recommended
2. **Access Control**

    - Role-based access control (RBAC)
    - Separate routes for doctors and patients
    - Middleware validation for all protected routes
3. **API Security**
    - Rate limiting implementation recommended
    - CORS configuration required for production
    - Input validation on all endpoints

Deployment Guidelines

1. **Prerequisites**
    - Node.js 14+ required
    - MongoDB 4.4+ recommended
    - SSL/TLS configuration for production
2. **Installation**

```
# Installation

npm install
```

3. **Running the Server**

```
# Development
npm run dev

# Production
npm start

# Run tests
npm test
```

Error Handling

The system implements comprehensive error handling:

- Validation errors return 400
- Authentication errors return 401
- Authorization errors return 403
- Not found errors return 404

- Server errors return 500

Scaling Considerations

1. **Database Optimization**
   - Implement MongoDB indexes
   - Consider sharding for large datasets
   - Use connection pooling
2. **Performance**
   - Cache frequently accessed data
   - Implement pagination for large lists
   - Use bulk operations where possible
3. **High Availability**
   - Deploy multiple instances
   - Implement load balancing
   - Use MongoDB replica sets

Test Coverage Goals

The system aims to maintain:

- Minimum 80% overall test coverage
- 100% coverage for security-critical paths
- Integration tests for all API endpoints
- Unit tests for all utility functions
- End-to-end tests for critical user workflows