Barcode Scanner Backend

A high-performance RESTful API backend for a barcode scanner application with user management, product tracking, and license management system.

Table of Contents

Project Structure

```
├── config/
│   └── db.js                 # Database configuration
├── controllers/
│   ├── userController.js     # User management logic
│   ├── productController.js  # Product management logic
│   └── licenseController.js  # License management logic
├── middlewares/
│   ├── auth.js               # Authentication middleware
│   └── securityMiddleware.js # Security features
├── models/
│   ├── User.js               # User model
│   ├── Product.js            # Product model
│   ├── License.js            # License model
│   └── AuditLog.js           # Audit logging model
├── routes/
│   ├── userRoutes.js         # User endpoints
│   ├── productRoutes.js      # Product endpoints
│   └── licenseRoutes.js      # License endpoints
├── tests/
│   ├── setup.js              # Test configuration
│   └── *.test.js             # Test files
├── utils/
│   └── jwt.js                # JWT utilities
├── app.js                    # Application entry point
└── package.json
```

Features

Core Features

- User Authentication & Authorization
- Product Management
- Barcode Scanning & Validation
- License Management System
- Activity Auditing
- Security Measures

License System Features

- Multiple License Tiers
- Trial License Support
- Device Binding
- Usage Tracking
- Anti-tampering Measures

Technologies Used

- Node.js
- Express.js
- MongoDB with Mongoose
- Redis for Caching
- JWT for Authentication
- Jest for Testing

Setup & Installation

1. Clone the repository:

```
git clone [repository-url]
cd barcode-scanner-backend
```

2. Install dependencies:

```
npm install
```

3. Set up environment variables:

```
cp .env.example .env
```

4. Start the server:

```
npm start
```

For development:

```
npm run dev
```

Environment Variables

Create a `.env` file with:

```
PORT=3000
MONGO_URI=mongodb://localhost:27017/barcode-scanner
JWT_SECRET=your_jwt_secret
REDIS_URL=redis://localhost:6379
```

API Documentation

User Management

```
POST   /api/users/register   - Register new user
POST   /api/users/login      - User login
GET    /api/users/profile    - Get user profile
PUT    /api/users/profile    - Update profile
DELETE /api/users            - Delete account
```

Product Management

```
POST   /api/products         - Create product
GET    /api/products         - List products
GET    /api/products/:id     - Get product details
PUT    /api/products/:id     - Update product
DELETE /api/products/:id     - Delete product
```

License Management

```
POST   /api/license/create   - Create new license
POST   /api/license/trial    - Create trial license
POST   /api/license/verify   - Verify license
GET    /api/license/list     - List all licenses (Admin)
PUT    /api/license/revoke/:key - Revoke license (Admin)
```

License System

License Tiers

1. **Trial License**

```
{
    "duration": "30 days",
    "scanLimit": 100,
    "features": ["basic_scan", "history"]
}
```

2. **Basic License**

```
{
    "duration": "180 days",
    "scanLimit": 1000,
    "features": ["basic_scan", "history", "export"]
}
```

3. **Premium License**

```
{
    "duration": "365 days",
    "scanLimit": 10000,
    "features": ["basic_scan", "history", "export", "bulk_scan", "analytics"]
}
```

4. **Enterprise License**

```
{
    "duration": "365 days",
    "scanLimit": "unlimited",
    "features": ["basic_scan", "history", "export", "bulk_scan", "analytics", "api_access", "priority_support"]
}
```

Security Features

1. Authentication & Authorization

- JWT-based authentication
- Role-based access control
- Session management
- Password hashing with bcrypt

2. Rate Limiting

- Login attempts limiting
- API rate limiting
- IP-based restrictions

3. Data Security

- Input sanitization
- XSS protection
- SQL injection prevention
- MongoDB injection prevention

4. License Protection

- Device fingerprinting
- License key encryption
- Usage tracking
- Anti-tampering measures
- IP tracking

Testing

Run tests:

```
# Run all tests
npm test

# Run with coverage
npm run test:coverage
```

Test files structure:

```
tests/
├── basic.test.js        # Basic connectivity tests
├── user.test.js         # User operations tests
├── product.test.js      # Product operations tests
├── license.test.js      # License system tests
└── security.test.js     # Security features tests
```

Performance Optimizations

1. Database Optimizations

- Proper indexing
- Lean queries
- Compound indexes
- Query optimization

2. Caching Strategy

- Redis caching for licenses
- Query result caching
- Rate limit caching

3. Security With Performance

- Efficient encryption
- Optimized validation
- Smart rate limiting

Contributing

1. Fork the repository
2. Create your feature branch
3. Commit your changes
4. Push to the branch
5. Create a new Pull Request

License