Online Auction System Documentation

A Full-Stack MERN Application

Table of Contents

Project Overview

The Online Auction System is a modern web application built using the MERN stack (MongoDB, Express.js, React.js, Node.js). It provides a platform for users to create and participate in online auctions.

Core Features

- User authentication and authorization
- Real-time auction bidding
- Dynamic auction listings
- Secure payment processing
- Responsive design for all devices

Technology Stack

- **Frontend**: React.js, CSS3, HTML5
- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Authentication**: JWT
- **Security**: Bcrypt

Technical Architecture

Frontend Architecture

```
// App.js - Main Component Structure
function App() {
  return (
    <Router>
      <div className="app">
        <header className="app-header">
          {/* Navigation and Auth Components */}
        </header>
        <main className="app-main">
          <Routes>
            <Route path="/" element={<Landing />} />
            <Route path="/dashboard" element={<ProtectedRoute><Dashboard /></ProtectedRoute>} />
            <Route path="/auction/:id" element={<ProtectedRoute><AuctionItem /></ProtectedRoute>} />
            {/* Other Routes */}
          </Routes>
        </main>
        <footer className="app-footer">
          {/* Footer Content */}
        </footer>
      </div>
    </Router>
  );
}
```

Backend Architecture

```
// server.js - Main Server Structure
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

// Middleware Configuration
app.use(express.json());
app.use(cors());

// Database Connection
mongoose.connect(MONGODB_URI)
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('MongoDB connection error:', err));

// Authentication Middleware
const authenticate = (req, res, next) => {
  const token = req.headers.authorization?.split(' ')[1];
  // Token verification logic
};

// Routes Implementation
app.post('/signup', async (req, res) => {
  // User registration logic
});

app.post('/auction', authenticate, async (req, res) => {
  // Auction creation logic
});
```

Features Implementation

1. User Authentication

- JWT-based authentication system
- Password hashing with bcrypt
- Protected routes implementation
- Session management

2. Auction Management

- Create new auctions
- Real-time bid updates
- Automatic auction closure
- Bid validation

3. Dashboard Features

- Filter and sort auctions
- Status indicators
- Time remaining display
- Bid history

4. Security Features

- Input validation
- XSS protection
- CSRF prevention
- Error handling

Code Structure

Frontend Structure

```
frontend/
├── public/
├── src/
│   ├── components/
│   │   ├── AuctionItem.js
│   │   ├── Dashboard.js
│   │   ├── Landing.js
│   │   ├── PostAuction.js
│   │   ├── Signin.js
│   │   └── Signup.js
│   ├── App.js
│   ├── App.css
│   └── index.js
```

Backend Structure

```
backend/
├── server.js
├── models/
│   ├── User.js
│   └── Auction.js
├── routes/
│   ├── auth.js
│   └── auctions.js
└── middleware/
    └── auth.js
```

Setup Instructions

1. Clone the Repository

```
git clone https://github.com/NSAjay2279/Online-Auction.git
cd Online-Auction
```

2. Install Dependencies

```
# Backend setup
cd backend
npm install

# Frontend setup
cd ../frontend
npm install
```

3. Environment Configuration

```
# Backend .env
PORT=5001
MONGODB_URI=mongodb://127.0.0.1:27017/auctionDB
SECRET_KEY=your_secret_key

# Frontend .env
REACT_APP_API_URL=http://localhost:5001
```

4. Start the Application

```
# Start backend
cd backend
npm start

# Start frontend (in a new terminal)
cd frontend
npm start
```

API Documentation

Authentication Endpoints

POST /signup

```
{
  "username": "string",
  "password": "string"
}
```

POST /signin

```
{
  "username": "string",
  "password": "string"
}
```

Auction Endpoints

POST /auction

```
{
  "itemName": "string",
  "description": "string",
  "startingBid": "number",
  "closingTime": "Date"
}
```

GET /auctions

Query Parameters:

- active: boolean
- sort: string (price, time)

POST /bid/:id

```
{
  "bid": "number"
}
```

Testing

Running Tests

```
# Frontend tests
cd frontend
npm test

# Backend tests
cd backend
npm test
```

Deployment

Frontend Deployment

1. Build the React application

```
cd frontend
npm run build
```

Backend Deployment

1. Configure production environment variables
2. Set up MongoDB Atlas connection
3. Deploy to hosting service (e.g., Heroku, AWS)

Future Enhancements

1. Real-time Features
   - WebSocket integration
   - Live chat system
   - Instant notifications
2. Additional Features
   - Image upload
   - Payment integration
   - Advanced search
   - User ratings
3. Performance Optimizations
   - Caching implementation
   - Load balancing
   - CDN integration