YourStyle Backend - Project Summary

🎉 Complete Production-Ready Backend System

This document provides an overview of the complete backend system that has been created for the YourStyle furniture e-commerce website.

📦 What's Included

✅ Core Backend Components

- **Node.js/Express API Server** - RESTful API with proper routing and middleware
- **PostgreSQL Database** - Relational database with proper schema design
- **JWT Authentication** - Secure token-based authentication system
- **Password Security** - bcrypt hashing with salt rounds
- **Input Validation** - express-validator for all user inputs
- **CORS Configuration** - Properly configured for your domains
- **Security Headers** - Helmet.js for security best practices

✅ Database Infrastructure

- **Docker Compose Setup** - PostgreSQL + pgAdmin in containers
- **Migration System** - Automated database schema creation
- **Seed Scripts** - Import initial products data
- **Proper Indexing** - Optimized queries with database indexes

✅ API Endpoints

- **Authentication APIs** - Register, Login with JWT
- **Product APIs** - Get all, Get by ID, Search
- **Cart APIs** - Full CRUD operations (Create, Read, Update, Delete)
- **Order APIs** - Create orders, View order history

✅ Configuration Files

- **Docker Compose** - Container orchestration
- **Environment Variables** - Secure configuration management
- **Apache Reverse Proxy** - Route /api/* requests to backend
- **.gitignore** - Prevent sensitive files from version control

✅ Comprehensive Documentation

- **SETUP.md** - Step-by-step installation guide for Windows/XAMPP
- **README.md** - Complete API documentation with examples
- **PROJECT_SUMMARY.md** - This overview document

📁 Project Structure

```
yourstyle-backend/
├── config/
│   └── database.js          # PostgreSQL connection configuration
├── controllers/
│   ├── authController.js       # Register & login logic
│   ├── cartController.js       # Cart CRUD operations
│   ├── ordersController.js     # Order creation & history
│   └── productsController.js   # Product listing & search
├── middleware/
│   └── auth.js                 # JWT authentication middleware
├── migrations/
│   ├── 001_create_tables.sql  # Database schema SQL
│   └── run-migrations.js       # Migration runner script
├── routes/
│   ├── auth.js                 # Authentication routes
│   ├── cart.js                 # Cart routes (protected)
│   ├── orders.js               # Order routes (protected)
│   └── products.js             # Product routes (public)
├── seeds/
│   ├── products.json           # Initial product data
│   └── seed-products.js        # Seed script runner
├── utils/
│   └── jwt.js                  # JWT generation & verification
├── .env.example               # Environment variables template
├── .gitignore                 # Git ignore rules
├── apache-reverse-proxy.conf  # Apache configuration for proxy
├── docker-compose.yml         # Docker services configuration
├── package.json               # Node.js dependencies & scripts
├── README.md                  # API documentation
├── SETUP.md                   # Installation guide
├── server.js                  # Main Express server
└── yarn.lock                  # Dependency lock file
```

🗄️ Database Schema

Tables Created:

1. **users** - User accounts
   - id, email, password_hash, name, phone, created_at

2. **products** - Product catalog
   - id, name, category, price_cents, image_url, description, created_at

3. **cart** - Shopping cart items
   - id, user_id, product_id, quantity, created_at, updated_at

4. **orders** - Order history
   - id, user_id, total_cents, status, created_at

5. **order_items** - Order line items
   - id, order_id, product_id, quantity, price_cents, created_at

All tables include proper foreign key relationships and indexes for performance.

🔐 Security Features

- ✅ **Password Hashing** - bcrypt with 10 salt rounds
- ✅ **JWT Authentication** - Secure token-based auth (7 day expiration)
- ✅ **Input Validation** - All user inputs validated and sanitized
- ✅ **SQL Injection Prevention** - Parameterized queries only
- ✅ **CORS Protection** - Only specified domains allowed
- ✅ **Security Headers** - Helmet.js middleware
- ✅ **Environment Variables** - Sensitive data not in code

🚀 Quick Start Guide

1. Prerequisites

- Windows with XAMPP installed
- Docker Desktop installed
- Node.js installed

2. Setup (5 minutes)

```
# Navigate to backend directory
cd C:\projects\yourstyle-backend

# Install dependencies
npm install

# Configure environment
copy .env.example .env
# Edit .env with your settings

# Start Docker containers
docker-compose up -d

# Run migrations (create tables)
npm run migrate

# Seed products data
npm run seed

# Start backend server
npm run dev
```

3. Configure Apache

- Copy `apache-reverse-proxy.conf` to XAMPP
- Enable proxy modules in `httpd.conf`
- Restart Apache

4. Test API

Visit: http://localhost:3000/health

📡 API Endpoints Overview

Public Endpoints (No Authentication Required)

Method	Endpoint	Description
POST	`/api/auth/register`	Create new user account
POST	`/api/auth/login`	Login and get JWT token
GET	`/api/products`	Get all products (filter by category)
GET	`/api/products/:id`	Get single product details
GET	`/api/search?q=query`	Search products

Protected Endpoints (JWT Required)

Method	Endpoint	Description
GET	`/api/cart`	Get user’s cart
POST	`/api/cart`	Add item to cart
PUT	`/api/cart/:id`	Update cart item quantity
DELETE	`/api/cart/:id`	Remove item from cart
POST	`/api/orders`	Create order from cart
GET	`/api/orders`	Get order history

🎯 Features Implemented

Authentication System

- ✅ User registration with validation
- ✅ Secure login with JWT tokens
- ✅ Password hashing with bcrypt
- ✅ Email uniqueness validation
- ✅ Token expiration (7 days default)

Product Management

- ✅ List all products
- ✅ Filter by category (tables, chairs, lamps)
- ✅ Search by name/description
- ✅ Get single product details

- ✅ Seeded with initial 9 products

Shopping Cart

- ✅ Add items to cart
- ✅ Update quantities
- ✅ Remove items
- ✅ View cart with line totals
- ✅ Cart persists in database
- ✅ User-specific carts

Order Processing

- ✅ Create orders from cart
- ✅ Transaction safety (all or nothing)
- ✅ Order history with details
- ✅ Cart clears after order
- ✅ Price snapshots preserved

🔧 Configuration Details

Environment Variables (.env)

```
# Server
PORT=3000
NODE_ENV=production

# Database
DB_HOST=localhost
DB_PORT=5432
DB_NAME=yourstyle_db
DB_USER=yourstyle_user
DB_PASSWORD=your_secure_password

# JWT
JWT_SECRET=your_random_secret_32_chars_min
JWT_EXPIRES_IN=7d

# CORS
CORS_ORIGIN=http://yourstyle.space,http://plitka.live

# pgAdmin
PGADMIN_EMAIL=admin@yourstyle.space
PGADMIN_PASSWORD=admin_password
```

Docker Services

- **PostgreSQL** - Port 5432 (persistent data)
- **pgAdmin** - Port 5050 (web UI for database management)

Node.js Server

- **Development** - Port 3000 with nodemon (auto-restart)
- **Production** - Port 3000 with PM2 recommended

📋 Available NPM Scripts

```
npm start           # Start server (production)
npm run dev         # Start with nodemon (development)
npm run migrate     # Run database migrations
npm run seed        # Seed products data
```

🌐 Apache Integration

The `apache-reverse-proxy.conf` file configures Apache to:
1. Serve static frontend from DocumentRoot
2. Proxy `/api/*` requests to Node.js backend (port 3000)
3. Handle CORS headers
4. Support both HTTP (port 80) and HTTPS (port 443)

📖 Documentation Files

SETUP.md

Comprehensive setup guide including:
- Docker Desktop installation
- Node.js installation
- Database setup steps
- Apache configuration
- SSL/TLS setup
- Troubleshooting common issues
- Production deployment notes
- Linux deployment alternative

README.md

Complete API documentation with:
- All endpoint specifications
- Request/response examples
- Authentication flow
- Error handling
- Database schema
- Security features
- Testing examples (cURL, Postman)

🔍 Testing the System

1. Health Check

```
curl http://localhost:3000/health
```

2. Register User

```
curl -X POST http://localhost:3000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"test123","name":"Test User"}'
```

3. Get Products

```
curl http://localhost:3000/api/products
```

4. Through Apache Proxy

```
curl http://yourstyle.space/api/products
```

🛡️ Production Recommendations

Before Going Live:

1. ✅ Change all default passwords in `.env`
2. ✅ Use strong JWT secret (32+ random characters)
3. ✅ Set `NODE_ENV=production`
4. ✅ Configure SSL certificates (Let's Encrypt)
5. ✅ Set up PM2 for process management
6. ✅ Configure automatic database backups
7. ✅ Set up monitoring and logging
8. ✅ Implement rate limiting
9. ✅ Regular security updates
10. ✅ Test all endpoints thoroughly

📦 Dependencies Used

Production Dependencies:

- **express** - Web framework
- **pg** - PostgreSQL client
- **bcrypt** - Password hashing
- **jsonwebtoken** - JWT authentication
- **cors** - CORS middleware

- **dotenv** - Environment variables
- **helmet** - Security headers
- **express-validator** - Input validation
- **morgan** - HTTP request logger

Development Dependencies:

- **nodemon** - Auto-restart on file changes

🎓 Learning Resources

Understanding the Code:

- **server.js** - Start here to see how everything connects
- **routes/** - See how URLs map to controllers
- **controllers/** - Business logic for each feature
- **middleware/auth.js** - How JWT authentication works
- **config/database.js** - Database connection pooling

🐛 Common Issues & Solutions

“Cannot connect to database”

→ Ensure Docker containers are running: `docker ps`

“Port 3000 already in use”

→ Find and kill process or change PORT in .env

“CORS error”

→ Check CORS_ORIGIN in .env includes your domain

“JWT token invalid”

→ Token expired (7 days), user needs to login again

See SETUP.md for detailed troubleshooting.

📈 Next Steps

Potential Enhancements:

1. **Email Verification** - Verify email addresses on registration
2. **Password Reset** - Forgot password functionality
3. **Admin Panel** - Manage products, orders, users
4. **Order Status Updates** - Track shipping, delivery
5. **Payment Integration** - Stripe, PayPal, etc.
6. **Product Reviews** - User ratings and reviews

7. **Image Upload** - Upload product images
8. **Inventory Management** - Track stock levels
9. **Discount Codes** - Promo codes and sales
10. **Email Notifications** - Order confirmations, updates

📞 Support & Maintenance

Database Backup (Recommended Daily)

```
docker exec yourstyle-postgres pg_dump -U yourstyle_user yourstyle_db > backup.sql
```

Update Dependencies

```
npm update
```

View Logs

```
# Backend logs
docker-compose logs -f

# PostgreSQL logs
docker logs yourstyle-postgres

# Apache logs
C:\xampp\apache\logs\error.log
C:\xampp\apache\logs\access.log
```

✅ Project Checklist

- [x] Database schema designed and implemented
- [x] Docker Compose configuration
- [x] Database migrations system
- [x] Product seeding script
- [x] User authentication (register/login)
- [x] JWT token generation and validation
- [x] Products API (list, details, search)
- [x] Cart API (full CRUD)
- [x] Orders API (create, history)
- [x] Input validation and sanitization
- [x] Error handling
- [x] Security middleware (Helmet, CORS)
- [x] Apache reverse proxy configuration
- [x] Comprehensive documentation
- [x] Environment variables setup

- [x] .gitignore for security
- [x] Production deployment notes

🏁 Summary

You now have a **complete, production-ready backend system** with:
- ✅ Secure authentication
- ✅ Full e-commerce functionality
- ✅ Database with proper schema
- ✅ Docker containerization
- ✅ Apache integration ready
- ✅ Comprehensive documentation
- ✅ Security best practices

Total Time to Deploy: ~30 minutes following SETUP.md

Ready for Production: Yes (after security configuration)

📄 License

ISC

Created: October 2025
Version: 1.0.0
Status: Production Ready ✅

🙏 Thank You!

This backend is now ready to power your YourStyle furniture e-commerce website. Follow the SETUP.md guide to get it running, and refer to README.md for API usage.

Good luck with your project! 🚀