WebShop Project - README

WebShop Project

A full-stack responsive e-commerce web application with a content management system (CMS). Built with **React** on the client side, **Node.js** + **Express** on the server side, and **MongoDB** for data persistence.

Client - WebShopClient

Tech Stack
- **React**
- **React Router DOM**
- **Axios**
- **CSS / SCSS**
- **Font Awesome / React Icons**

Main Features
- Responsive design compatible with all screen sizes
- Homepage with:
  - Title, subtitle, banner image
  - Search field
  - At least 3 product cards
- Product listing and details pages
- Dynamic navigation menu and footer
- User registration and login with validation:
  - Passwords must contain 1 uppercase, 1 lowercase, 4+ letters, and 1 special character
- JWT-based authentication
- Save favorite items to user's list
- Forms with client-side validation
- About page with app explanation
- Visual feedback for user actions (success/fail)
- Filter & view mode switching options for content

Structure
- `/src/components/` Reusable UI components
- `/src/pages/` Main views (Home, Product, Register, etc.)
- `/src/context/` Global state using React Context
- `/src/utils/` Utility functions and constants

Server - WebShopServer

Tech Stack
- **Node.js**
- **Express.js**
- **MongoDB + Mongoose**
- **JWT** for token-based auth
- **dotenv** for environment variables
- **Morgan** for HTTP logging
- **Nodemon** for development

Main Features
- REST API for CRUD operations

WebShop Project - README

- Authentication & authorization middleware
  - Only `admin` can create/edit/delete data
- JWT issued upon successful login
- Rate limiting middleware to prevent abuse
- Models: User, Product, Favorites, CustomerRequest
- Validation via **Joi** + **Mongoose**
- Clean architecture: routes, controllers, models, middleware
- Logs and error handling
- Protected routes for sensitive actions
- Admin user management features

Structure
- `/routes/` All route handlers
- `/models/` Mongoose models
- `/middlewares/` Auth, logger, rate limiter
- `/utils/` Helper functions
- `.env` Environment configuration (NOT shared)
- `index.js` Server entry point

Getting Started

Prerequisites
- Node.js
- MongoDB (local or cloud via Atlas)

Run Server
```bash
cd WebShopServer
npm install
npm run dev
```

Run Client
```bash
cd WebShopClient
npm install
npm start
```

Notes

- Do not upload `node_modules` to git
- Add `.env` file with relevant keys (e.g., Mongo URI, JWT secret)
- All images must include `alt` attribute for accessibility
- Code must follow industry best practices: modular, semantic, documented

License

This project was developed as a final project for a web development course.