**GROCERY DELIVERY APPLICATION**

Here's a basic implementation of the key components of a grocery delivery app using:

**Front-End**: React.js

**Back-End**: Node.js with Express.js

**Database**: MongoDB

**Authentication**: JSON Web Tokens (JWT)

**Project Structure**

grocery-delivery-app/

│

├── backend/

│ ├── config/

│ │ └── db.js

│ ├── controllers/

│ │ ├── authController.js

│ │ ├── productController.js

│ │ ├── orderController.js

│ ├── models/

│ │ ├── User.js

│ │ ├── Product.js

│ │ ├── Order.js

│ ├── routes/

│ │ ├── authRoutes.js

│ │ ├── productRoutes.js

│ │ ├── orderRoutes.js

│ ├── middleware/

│ │ └── authMiddleware.js

│ ├── server.js

│

├── frontend/

│ ├── public/

│ └── src/

│ ├── components/

│ ├── pages/

│ ├── services/

│ └── App.js

│

└── package.json

**Backend (Node.js + Express)**

backend/server.js

javascript

const express = require('express');

const cors = require('cors');

const mongoose = require('mongoose');

const authRoutes = require('./routes/authRoutes');

const productRoutes = require('./routes/productRoutes');

const orderRoutes = require('./routes/orderRoutes');

const { requireAuth } = require('./middleware/authMiddleware');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

app.use(express.json());

app.use(cors());

// MongoDB connection

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/auth', authRoutes);

app.use('/api/products', productRoutes);

app.use('/api/orders', requireAuth, orderRoutes);

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

**backend/models/User.js**

javascript

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const userSchema = new mongoose.Schema({

name: { type: String, required: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

role: { type: String, default: 'customer' }, // 'customer' or 'admin'

});

**// Hash password before saving**

userSchema.pre('save', async function (next) {

if (!this.isModified('password')) return next();

this.password = await bcrypt.hash(this.password, 10);

next();

});

**// Validate password**

userSchema.methods.comparePassword = async function (password) {

return await bcrypt.compare(password, this.password);

};

module.exports = mongoose.model('User', userSchema);

**backend/models/Product.js**

javascript

const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({

name: { type: String, required: true },

description: { type: String, required: true },

price: { type: Number, required: true },

category: { type: String, required: true },

imageUrl: { type: String },

});

module.exports = mongoose.model('Product', productSchema);

**backend/models/Order.js**

javascript

const mongoose = require('mongoose');

const orderSchema = new mongoose.Schema({

customer: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },

items: [

{

product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product', required: true },

quantity: { type: Number, required: true }

}

],

totalAmount: { type: Number, required: true },

deliveryAddress: { type: String, required: true },

status: { type: String, default: 'pending' }, // pending, shipped, delivered

createdAt: { type: Date, default: Date.now },

});

module.exports = mongoose.model('Order', orderSchema);

**backend/controllers/authController.js**

javascript

const User = require('../models/User');

const jwt = require('jsonwebtoken');

// Create JWT token

const createToken = (user) => {

return jwt.sign({ id: user.\_id, role: user.role }, process.env.JWT\_SECRET, { expiresIn: '1d' });

};

***// User registration***

exports.register = async (req, res) => {

const { name, email, password } = req.body;

try {

const user = await User.create({ name, email, password });

const token = createToken(user);

res.status(201).json({ user, token });

} catch (err) {

res.status(400).json({ error: 'Registration failed' });

}

};

**// User login**

exports.login = async (req, res) => {

const { email, password } = req.body;

try {

const user = await User.findOne({ email });

if (!user || !(await user.comparePassword(password))) {

return res.status(400).json({ error: 'Invalid credentials' });

}

const token = createToken(user);

res.status(200).json({ user, token });

} catch (err) {

res.status(400).json({ error: 'Login failed' });

}

};

**backend/controllers/productController.js**

javascript

const Product = require('../models/Product');

**// Get all products**

exports.getProducts = async (req, res) => {

try {

const products = await Product.find();

res.status(200).json(products);

} catch (err) {

res.status(400).json({ error: 'Unable to fetch products' });

}

};

**// Create a product (Admin only)**

exports.createProduct = async (req, res) => {

const { name, description, price, category, imageUrl } = req.body;

try {

const product = await Product.create({ name, description, price, category, imageUrl });

res.status(201).json(product);

} catch (err) {

res.status(400).json({ error: 'Unable to create product' });

}

};

backend/controllers/orderController.js

javascript

const Order = require('../models/Order');

**// Create an order (Customer only)**

exports.createOrder = async (req, res) => {

const { items, totalAmount, deliveryAddress } = req.body;

try {

const order = await Order.create({

customer: req.user.id,

items,

totalAmount,

deliveryAddress,

});

res.status(201).json(order);

} catch (err) {

res.status(400).json({ error: 'Unable to place order' });

}

};

**// Get orders for a customer**

exports.getOrders = async (req, res) => {

try {

const orders = await Order.find({ customer: req.user.id }).populate('items.product');

res.status(200).json(orders);

} catch (err) {

res.status(400).json({ error: 'Unable to fetch orders' });

}

};

**backend/routes/authRoutes.js**

javascript

const express = require('express');

const { register, login } = require('../controllers/authController');

const router = express.Router();

router.post('/register', register);

router.post('/login', login);

module.exports = router;

**backend/routes/productRoutes.js**

javascript

const express = require('express');

const { getProducts, createProduct } = require('../controllers/productController');

const { requireAuth } = require('../middleware/authMiddleware');

const router = express.Router();

router.get('/', getProducts);

router.post('/', requireAuth, createProduct); // Only admin should be able to create products

module.exports = router;

**backend/routes/orderRoutes.js**

javascript

const express = require('express');

const { createOrder, getOrders } = require('../controllers/orderController');

const router = express.Router();

router.post('/', createOrder);

router.get('/', getOrders);

module.exports = router;

backend/middleware/authMiddleware.js

javascript

const jwt = require('jsonwebtoken');

exports.requireAuth = (req, res, next) => {

const token = req.headers.authorization.split(' ')[1];

if (!token) return res.status(401).json({ error: 'Unauthorized' });

jwt.verify(token, process.env.JWT\_SECRET, (err, decodedToken) => {

if (err) return res.status(401).json({ error: 'Unauthorized' });

req.user = decodedToken;

next();

});

};

**Frontend (React.js)**

**frontend/src/App.js**

javascript

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Login from './pages/Login';

import Register from './pages/Register';

import Products from './pages/Products';

import Cart from './pages/Cart';

import Orders from './pages/Orders';

function App() {

return (

<Router>

<Switch>

<Route path="/login" component={Login} />

<Route path