🍰 BakeBuddy — Full Stack Bakery Management System

MERN + JWT Auth + Orders + Cart + Analytics + Emails

📁 Project Structure (Important)

```
Bakebuddy-System/
│
├── client/                 # React Frontend
│   ├── src/
│   │   ├── pages/
│   │   │   ├── Login.jsx
│   │   │   ├── Register.jsx
│   │   │   ├── CustomerHome.jsx
│   │   │   ├── Cart.jsx
│   │   │   ├── MyOrders.jsx
│   │   │   ├── AdminLogin.jsx
│   │   │   ├── AdminOrders.jsx
│   │   │   ├── AdminAnalytics.jsx
│   │   │   └── AdminUsers.jsx
│   │   ├── api.js
│   │   ├── App.js
│   └── public/
│
└── server/
    ├── src/
    │   ├── app.js
    │   ├── server.js
    │   ├── config/db.js
    │   ├── middleware/
    │   │      auth.middleware.js
    │   │      role.middleware.js
    │   ├── controllers/
    │   │      auth.controller.js
    │   │      product.controller.js
    │   │      cart.controller.js
    │   │      order.controller.js
    │   ├── models/
    │   │      user.model.js
    │   │      product.model.js
    │   │      cart.model.js
    │   │      order.model.js
    │   ├── utils/email.js
    │   └── routes/
    │
```

🚀 1. Backend Essentials

🗄️ db.js — MongoDB Connection

```
import mongoose from "mongoose";

export default async function connectDB() {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("📦 MongoDB Connected Successfully");
  } catch (err) {
    console.error("❌ MongoDB Connection Failed");
  }
}
```

🔐 auth.middleware.js — Token Validation

```
import jwt from "jsonwebtoken";

export const protect = (req, res, next) => {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.status(401).json("No token");

  const decoded = jwt.verify(token, process.env.JWT_SECRET);
  req.user = decoded;
  next();
};
```

👨‍💼 role.middleware.js — Admin Guard

```
export const adminOnly = (req, res, next) => {
  if (req.user.role !== "admin") {
    return res.status(403).json("Admins only!");
  }
  next();
};
```

👤 user.model.js

```
const userSchema = new mongoose.Schema({
  name: String,
  email: { type: String, unique: true },
  password: String,
  role: { type: String, default: "customer" }
});
```

📦 product.model.js

```
const productSchema = new mongoose.Schema({
  name: String,
  price: Number,
  image: String,
});
```

🛒 order.model.js

```
const orderSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  items: [],
  totalAmount: Number,
  status: { type: String, default: "pending" }
});
```

📬 email.js — Send Email

```
import nodemailer from "nodemailer";

export const sendWelcomeEmail = async (to, name) => {
  await transporter.sendMail({
    to,
    subject: "Welcome to BakeBuddy! 🎉",
    html: `<h2>Hi ${name}, your BakeBuddy account is ready! 🍰</h2>`
  });
};
```

🍞 2. Important Backend Controllers

🔑 auth.controller.js — User Login

```
export const login = async (req, res) => {
  const user = await User.findOne({ email: req.body.email });
  if (!user) return res.status(400).json("User not found");

  const token = jwt.sign(
    { id: user._id, role: user.role },
    process.env.JWT_SECRET
  );

  res.json({ token, role: user.role, name: user.name });
};
```

🧺 cart.controller.js — Add to Cart

```
export const addToCart = async (req, res) => {
  const cart = await Cart.findOneAndUpdate(
    { user: req.user.id },
    { $push: { items: req.body } },
    { new: true, upsert: true }
  );
  res.json(cart);
};
```

📦 order.controller.js — Place Order

```
export const createOrder = async (req, res) => {
  const order = await Order.create({
    user: req.user.id,
    items: req.body.items,
    totalAmount: req.body.totalAmount
  });

  sendOrderPlacedEmail(req.user.email, order);
  res.json(order);
};
```

📊 analytics.controller.js

```
export const getAnalytics = async (req, res) => {
  const completedOrders = await Order.find({ status: "completed" });
  const totalRevenue = completedOrders.reduce((a, b) => a +
b.totalAmount, 0);

  res.json({ totalRevenue, completedOrders: completedOrders.length });
};
```

🎨 3. Frontend Essentials

🌐 api.js — Axios Setup

```
const api = axios.create({
  baseURL: "http://localhost:10000/api",
});
```

🔐 Login.jsx

```
const res = await api.post("/auth/login", { email, password });
localStorage.setItem("token", res.data.token);
navigate(res.data.role === "admin" ? "/admin" : "/home");
```

🏠 CustomerHome.jsx

```
useEffect(() => {
  api.get("/products").then(res => setProducts(res.data));
}, []);
```

🛒 Cart.jsx

```
const handlePlaceOrder = async () => {
  await api.post("/orders", { items: cart, totalAmount });
  alert("Order Placed!");
};
```

📦 AdminOrders.jsx

```
const updateStatus = async (id, status) => {
  await api.put(`/orders/${id}`, { status });
  loadOrders();
};
```

📊 AdminAnalytics.jsx

```
useEffect(() => {
  api.get("/analytics").then(res => setStats(res.data));
}, []);
```

👥 AdminUsers.jsx

```
useEffect(() => {
  api.get("/users").then(res => setUsers(res.data));
}, []);
```

🎯 4. Complete System Flow (Explain in Presentation)

Customer Flow

1. Register → Email Sent → Login
2. View Menu
3. Add to Cart
4. Place Order (Email sent)
5. Track Order Status
6. Completed → Email Notification

Admin Flow

1. Login → Dashboard
2. CRUD Products
3. View Orders
4. Update Status
5. Revenue Updates Automatically
6. Check Users Page
7. Check Analytics Page

💡 5. Key Selling Points

⭐ Dual Role Architecture

Admin & Customer modes with isolated dashboards.

⭐ Beautiful Pastel UI

Perfect bakery-themed experience.

⭐ Professional MERN Architecture

MVC backend, modular React frontend.

⭐ Automated Email Engine

Welcome email, order confirmation, status updates.

⭐ Analytics Dashboard

Auto-calculated based on order completions.

⭐ Users Management System

Admin can view all registered users.

⭐ Real Business Simulation

Full e-commerce style functionality.

📦 6. How to Run

```
cd server
npm install
npm run dev

cd ../client
npm install
npm start
```

👨‍💻 Developer

Javeed Quadri Mohammad (YashDev-Design)

MS CS · AUM · Full Stack Developer
