Bringy: Grocery Delivery Web app

Back-End Project Documentation

04.06.2025

Project Overview:

Video Link: <https://drive.google.com/file/d/1sta14coSyAk_Y9gHR-Iy01wVlMVa56NV/view?usp=sharing>

GitHub Link: <https://github.com/SohailElskhawy/Bringy>

This is a RESTful API backend designed for an e-commerce application supporting online ordering, product and supplier management, and a customer basket system.

Target Users: Admin users & Customers

API Style: REST

Version: v1.0.0

Maintainers:

* + 1. 1306230116  Sohail Mohamed Elskhawy
    2. 1306220129 Ebrahim Alkridi
    3. 1306210119 Sidra Bkdash

Release Cycle: Bi-weekly feature releases, hotfix patches as needed

Scope:

- User authentication & registration  
- Product catalog & search  
- Shopping basket  
- Order processing  
- Supplier & category management

g. Out of Scope:

- Payment gateway integration (future scope)  
- File/media handling

Technology Stack:

Programming Language: Javascript

Web Framework: Node.js v18.17.1 + Express.js v1.7.14 + React 19

Database: MongoDB 14

Authentication: JWT

Setup & Development Environment

Prerequisites:

* + 1. Node.js(v14+)
    2. Npm or yarn
    3. Git

Local Setup;

1. Clone the repository:



1. Switch to development branch



3. Make Sure The Secret Keys .env File Is Inside The Backend Folder



4. Open two terminals for running

-Backend terminal

- Frontend terminal

Folder Structure 

Database Schema

Collections:  
- Users, Products, Orders, OrderItems, Categories, Suppliers, Baskets  
  
Relationships:  
- Order.customerId → Users.\_id  
- OrderItems.orderId → Orders.\_id  
- Product.category\_id → Categories.\_id  
- Product.supplier\_id → Suppliers.\_id  
- Basket.customerId → Users.\_id

Environment Variables:

.env

MONGO\_URI=mongodb+srv://sohailelskhawy:[WdjEtLNHrwaRRYLl@bringy.bzol5.mongodb.net](mailto:WdjEtLNHrwaRRYLl@bringy.bzol5.mongodb.net)/?retryWrites=true&w=majority&appName=bringy

OPENAI\_API\_KEY=sk-proj-R1AhOVjjaYZ7EfDOTiVBmf0Um7xvvNUPPqXD4lsrfeifqT7kFbwbJ4FoFFxzph-c0h7BHAIikWT3BlbkFJJ5\_Nj\_G5Vdx8Jxxv8hug9BwnTh1M07uviQJ63HSXuwCUR4\_pX3GA6pVJUqNuJgxbRNebfzxWQA

MJ\_APIKEY\_PUBLIC=400791df4f902b6e3601e14096659bcc

MJ\_APIKEY\_PRIVATE=902a4fb34f758f9e8d92d89c426f7190

CLIENT\_URL=http://localhost:5000

Testing Strategy

- Unit: Jest  
- E2E: Full workflow  
- Coverage: 85%+

Deployment Pipeline

GitHub Actions:  
 The deployment process is automated using GitHub Actions. On every push to the main branch, the CI/CD pipeline runs a series of steps:

* Run unit and integration tests
* Build the application
* Deploy to the staging environment
* If tests pass, proceed to production deployment

Secrets and environment variables are managed securely via GitHub Secrets.

Rollback:  
 In case of a failed deployment or critical bug in production, a rollback can be triggered by:

* Reverting the latest commit on main and pushing the change, which triggers the pipeline to redeploy the previous stable version.
* Alternatively, manually deploying a previous build using a version tag or release via GitHub Actions or the hosting provider’s dashboard (e.g., Vercel, Heroku, AWS, etc.).

Error Handling & Logging:

- Logs: JSON  
- Central error handling middleware

API Security & Rate Limiting:

Access Token: 15 mins

Refresh Token: 7 days

Passwords: Hashed with argon2id

Rate limits: 100 req/min/user

IP Whitelist: Configurable per environment

CORS: Allowlist

Monitoring & Observability

Prometheus /metrics

Grafana dashboards:

* + API latency (p50/p95)
  + Error rate (4xx/5xx)
  + Uptime %
  + DB query duration

Slack alerts

Service-Level Agreements (SLA)

Uptime: %99.9

Max response time: <500ms (p95)

Critical bug fix time: <4h

Data protection: KVKK & GDPR compliant

Backups:

* + Frequency: Hourly data backup
  + Retention: 7 days
  + Location: AWS S3
  + Hazard scenario: physical & geographical backups

Contribution Guidelines:

Git Strategies: All development was done directly on the dev branch. For future work, using feature branches (e.g., feature/xyz) or hotfix branches (hotfix/xyz) is recommended for better collaboration and review.

Code Style: JavaScript code is linted using ESLint with the Airbnb style guide.

Checklist:

* + Code formatted
  + Tests included
  + Docs updated

Onboarding & Appendix

- Git: feature/xyz, hotfix/xyz  
- Style: ESLint + Airbnb  
- Checklist:

* Code is properly formatted
* Relevant tests are written and passing
* Documentation is updated (README, Swagger, etc.)
* No console.log or debugging artifacts in production code