Umbra: Task Management System

Back-End Project Documentation

04.06.2025

1. Project Overview:

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

* 1. Target Users: Admin users & Customers
  2. API Style: REST
  3. Version: v1.0.0
  4. Maintainers:
     1. 1306230116  Sohail Mohamed Elskhawy
     2. 1306220129 Ebrahim Alkridi
     3. 1306210119 Sidra Bkdash
  5. Release Cycle: Bi-weekly feature releases, hotfix patches as needed
  6. 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

1. Technology Stack:
   1. Programming Language: Javascript
   2. Web Framework: Node.js v18.17.1 + Express.js v1.7.14
   3. Database: MongoDB 14
   4. Authentication: JWT
   5. Deployment: AWS (via Docker Compose)
2. Setup & Development Environment
   1. Prerequisites:
      1. Node.js(v14+)
      2. Npm or yarn
      3. Git
   2. Local Setup;
3. Clone the repository:

git clone https://github.com/SohailElskhawy/Bringy.git

cd bringy

1. Switch to development branch

git checkout dev

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

\Bringy\backend\.env

4. Open two terminals for running

-Backend terminal

- Frontend terminal

cd frontend

npm install

npm run dev

cd backend

npm install

npm start

1. Folder Structure

bringy/  
├── backend/  
│ ├── controllers/  
│ ├── models/  
│ ├── routes/  
│ ├── services/  
│ ├── tests/  
│ ├── .env  
│ └── server.js  
├── frontend/  
│ └── src/  
└── README.md

1. API Documentation:

Authentication  
- POST /api/v1/auth/register  
- POST /api/v1/auth/login  
- GET /api/v1/auth/verify-email  
  
Products  
- GET /api/v1/products  
- GET /api/v1/products/search/:searchTerm  
- GET /api/v1/products/sort/:order  
- GET /api/v1/products/category/:category\_id  
- POST /api/v1/products  
- PUT /api/v1/products/:id  
- DELETE /api/v1/products/:id  
- PATCH /api/v1/products/restore/:id  
  
Orders  
- POST /api/v1/orders  
- GET /api/v1/orders  
- GET /api/v1/orders/:id  
- PUT /api/v1/orders/:id/status  
- DELETE /api/v1/orders/:id  
- GET /api/v1/orders/customer/:customerId  
  
Basket  
- POST /api/v1/basket/add  
- PUT /api/v1/basket/increase  
- PUT /api/v1/basket/decrease  
- DELETE /api/v1/basket/remove  
- DELETE /api/v1/basket/clear  
  
Categories  
- POST /api/v1/categories  
- GET /api/v1/categories  
- GET /api/v1/categories/:id  
- PUT /api/v1/categories/:id  
- DELETE /api/v1/categories/:id  
  
Suppliers  
- POST /api/v1/suppliers  
- GET /api/v1/suppliers

1. 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

1. Environment Variables:

.env

DATABASE\_URL:

CORS\_ORIGIN:

Database\_URL

mongodb+srv://sohailelskhawy:WdjEtLNHrwaRRYLl@bringy.bzol5.mongodb.net/?

1. Testing Strategy

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

1. Deployment Pipeline
   1. GitHub Actions:
   2. Rollback:
2. Error Handling & Logging:

- Logs: JSON  
- Central error handling middleware

1. API Security & Rate Limiting:
   1. Access Token: 15 mins
   2. Refresh Token: 7 days
   3. Passwords: Hashed with argon2id
   4. Rate limits: 100 req/min/user
   5. IP Whitelist: Configurable per environment
   6. CORS: Allowlist
2. Monitoring & Observability
   1. Prometheus /metrics
   2. Grafana dashboards:
      1. API latency (p50/p95)
      2. Error rate (4xx/5xx)
      3. Uptime %
      4. DB query duration
   3. Slack alerts
3. Service-Level Agreements (SLA)
   1. Uptime: %99.9
   2. Max response time: <500ms (p95)
   3. Critical bug fix time: <4h
   4. Data protection: KVKK & GDPR compliant
   5. Backups:
      1. Frequency: Hourly data backup
      2. Retention: 7 days
      3. Location: AWS S3
      4. Hazard scenario: phsical & geographical backups
4. Contribution Guidelines:
   1. Git Strategies: Feature branching (feature/xyz)
   2. Code Style: PEP8
   3. Checklist:
      1. Code formatted
      2. Tests included
      3. Docs updated
5. Onboarding & Appendix

- Git: feature/xyz, hotfix/xyz  
- Style: ESLint + Airbnb  
- Checklist:  
 [ ] Tests added  
 [ ] Swagger updated  
 [ ] No console.logs  
  
Steps:  
- Clone repo  
- Copy .env  
- make up, make test