

# Retro Bytes - Build and Setup Guide

This document describes how to set up, build, run, and verify the Retro Bytes application on a clean developer machine.

## 1. Prerequisites

- Go 1.22+ (<https://go.dev/dl>)
- Git (<https://git-scm.com/downloads>)
- Windows PowerShell or a terminal (macOS/Linux work too)
- Optional: SQLite CLI for manual inspection
- Ports: ensure port 8081 is free (the app currently listens on 8081)

## 2. Project Structure (Key folders)

- cmd/retrobytes – application entrypoint (main.go)
- internal/config – configuration loader (PORT, DB\_DSN, MEDIA\_DIR, LOG\_FILE)
- internal/http/handlers – HTTP route handlers (pages + APIs)
- internal/repos – database access, schema/seed logic
- internal/services – business logic (catalog, cart, inventory, orders, wishlist)
- internal/validate – input validators (ZIP, email, name, password, IDs, search query)
- web/templates – Go HTML templates
- web/static – CSS and static assets
- web/media – product images served at /media
- retrobytes.db – SQLite DB (auto-created/seeded on first run)
- Tests: internal/http/\*\_test.go for security/validation/logging authz and limits

## 3. Clone the Repository

```
git clone https://code.umd.edu/ksp/enpm680fall25project-ksp.git
```

```
cd enpm680fall25project-ksp
```

## 4. Configuration

Defaults (override via environment variables):

- PORT: default 8080 (note: main listens on 8081 currently)
- DB\_DSN: default retrobytes.db (SQLite file in project root)
- MEDIA\_DIR: default ./web/media
- LOG\_FILE: default ./retrobytes.log (logs to file + stdout)

Example (PowerShell):

```
$env:PORT='8081'
```

```
$env:DB_DSN='retrobytes.db'
```

```
$env:MEDIA_DIR='./web/media'
```

```
$env:LOG_FILE='./retrobytes.log'
```

## 5. Install Dependencies

```
go mod tidy
```

## 6. Build & Run

```
go run .\cmd\retrobytes
```

(Starts on port 8081 with current code.)

## 7. Database Setup

Automatic. On first run, schema is created and seeded with:

- Categories/products/inventory (idempotent)
- Users: alice, bob, luke, yoda (USER) and admin (ADMIN), all with password Passw0rd!
- To reset the database: stop the app and delete retrobytes.db, then start the app again.

## 8. Verification (Smoke Test)

- Home: <http://localhost:8081/> shows categories.
- Category: </category/retro-consoles> shows products.
- Product: </product/gbc-001> shows details and availability check with ZIP 20742.
- Cart/Order: add item → </cart> → </checkout> → place order → redirected to </order/{id}>.
- Wishlist: </wishlist> add/remove items.
- Search: </search?q=game> with filters.
- Admin (log in as [admin@retrobytes.test](mailto:admin@retrobytes.test) / Passw0rd!): see “Admin” link, then:
  - </admin/orders> view/cancel orders
  - </admin/inventory> edit stock, recent orders block
  - </admin/users> delete non-admin users (orders set to CANCELED but retained)

## 9. Security/Behavior Highlights

- Input validation: ZIP (5 digits), search query whitelist, name/email/password length & format, ID/qty limits, cart max total qty=10.
- Auth/Session: bcrypt hashes, login throttling (5/10 min global app, tighter in tests), logout expires cookie and order ownership enforced, wishlist scoped to session.
- Data protection: media path traversal blocked (raw and encoded). Static media root fixed and logs go to stdout + LOG\_FILE.
- Error handling: friendly messages, global error handler logs details server-side.
- Logging: auth success/fail, access denials, validation failures, rate-limit hits, order totals (server/client/mismatch), admin inventory/user deletes, CSRF failures, server errors.
- Rate limits: </search> and </api/v1/availability> throttled and </login> throttled.

- Body size: 1 MiB max request body (cart/orders).

## 10. Testing / Verification

■ To run the full suite at once: **go test ./internal/http**

■ Security-focused tests (run individually if needed):

```
go test ./internal/http/input_validation_test.go    # SR-VAL-01/02
```

```
go test ./internal/http/auth_test.go              # SR-AUTH-01/02
```

```
go test ./internal/http/authz_order_totals_test.go # SR-AUTHZ-01/02
```

```
go test ./internal/http/authz_admin_test.go       # SR-AUTHZ-04
```

```
go test ./internal/http/log_auth_test.go          # SR-LOG-01
```

```
go test ./internal/http/log_access_test.go        # SR-LOG-02
```

```
go test ./internal/http/log_admin_inventory_test.go # SR-LOG-04
```

```
go test ./internal/http/error_handler_test.go     # SR-ERR-01
```

```
go test ./internal/http/rate_size_test.go         # SR-RATE-01 / SR-SIZE-01
```

```
go test ./internal/http                        # run all of the above together
```

If build cache permissions block tests on your machine, set a writable GOCACHE (e.g., C:\Users\prane\workspace\go-cache) before running.

## 11. Troubleshooting

- Port in use: set PORT/update listen port and restart.
- Templates not found in tests: ensure relative path ../../web/templates when running from package directories.
- Media 404: ensure MEDIA\_DIR points to ./web/media and verify files exist under web/media/products/<id>/main.jpg.
- CSRF 403: ensure hidden csrf field matches csrf\_cookie (refresh /login if stale).
- Rate limit 429: wait for window reset (search/availability/login).
- Body too large: keep cart/order POST bodies <1 MiB (default server limit).

## 12. Security Notes

- Use Secure cookies in production behind HTTPS.
- Session IDs are HttpOnly, SameSite=Lax, not rotated automatically and logout unbinds and expires cookie.
- Server recomputes order totals/prices and stock enforced at checkout.