EASYGAMES
GROUP 1 REPORT

HIT 339 (Assessment 3)

Alex Phan	S325049
Veasna Kuoy	S317557
Vendel Gomes	S321266
Gislene FDLClancy	S361662

Contents

Topic 1. Introduction

1.1 Purpose

This report is the lecturer's guide to EasyGames. It explains what we built, why we made key design choices and how the final system satisfies the Software Requirements Specification. You'll find quick run instructions (.NET 8 + SQLite) with an Owner login for assessment, one page SRS summary (functional and non-functional requirements with MoSCoW priorities), and a traceability table linking each requirement to concrete evidence (screens, code excerpts, and test results). We document our development approach, team roles, timeline, and meeting notes to make individual contributions visible. The report also highlights key risks and mitigations, known limitations, and small high value enhancements for future work. Our goal is to enable fast, repeatable verification of the app and transparent evaluation of team process.

Transparency note: While the team's GitHub repository commits history shows heavier implementation by the Tech Lead (Assessment2 baseline owner) this only reflects time pressure and end of semester constraints rather than a lack of intent. The rest of the team remained engaged in meetings and on Teams; with more runway, contributions would have been broader. Please read the submission as a coordinated team effort delivered under tight deadlines.

1.2 Scope note.

This project delivers a working ASP.NET Core MVC web app for EasyGames that supports basic e-commerce for books, games, and toys.

In scope (implemented for Ass3):

- **Cart & Checkout:** add/update/remove items; demo checkout creates Order/OrderItems and clears cart.
- **Owner → Products:** stock management (create/edit/delete), Cost Price & Supplier, Margin/Margin% display.

- **Orders & Inventory:** checkout decrements stock transaction; orders store price/cost snapshots and compute Subtotal/Tax/Total/Profit.
- **Customer → My Orders:** order history with lifetime tier badge (Bronze/Silver/Gold/Platinum baseline).
- **Admin → Users:** basic user management (list/create/edit/delete; Owner-only access).
- **Admin → Email Groups:** Owner can send to All customers; simulated send by default (SMTP optional).

Technology target: .NET 8, EF Core with SQLite, cookie authentication, Bootstrap based styling.

Out of scope for A3 (documented as future work): real payment gateways, shipping carriers/rates, tax engines, multi-shop/tenant or multi-currency, image upload/CDN, advanced analytics/reporting, and automated tier-targeted campaigns. A manual Change Password exists; an admin reset-password flow is not implemented.

Reason for exclusions: protect stability and evidence quality; scope frozen T–4days before submission.

Topic 2. Installation & Running + User Guide

2.1 Prerequisites: Before you start (what you need)

To run the app, you'll need the **.**NET 8 SDK (it works safely alongside .NET 7 so nothing gets replaced or removed). You don't need to install a database separately because SQLite is built-in the app will create the file automatically the first time it runs.

You can use any of these tools to open or run the project:

- **Visual Studio 2022 Community** (with the "ASP.NET & web" workload),
- **VS Code** with the **C# Dev Kit**, or
- just the **Command Line** if you prefer using terminal commands.

You'll also need **Git** if you plan to clone the project from GitHub (it's optional if you just download the ZIP file). If so get the code via Git:

Git repository: alex-phan/easygames — default branch **main**.
Clone: *git clone https://github.com/alex-phan/easygames.git*

Why .NET 8?
The app was built for .NET 8. If your computer only has .NET 7, no problem in installing .NET 8 will sit side-by-side safely without breaking anything.

So, Step 1 — Get the Code

Option	What to Do	Example Path / Command
A — ZIP (easiest)	1 Download the ZIP file 2 Extract it to a simple folder	**Windows:** C:\Dev\EasyGames**macOS/Linux:** ~ /Dev/EasyGames
B — Git (for developers)	1 Open Terminal or Git Bash 2 Run these commands	git clone <YOUR_REPO_URL>cd EasyGames

***Note**: Emoji rendering system font **Segoe UI Emoji** (Windows 11)*

Note: If you want to manage the database using commands, install the optional **dotnet-ef** tool.

Step 2 — Build & Create the Database

Option	Steps	What You'll See
Visual Studio (Windows)	1 Double-click EasyGames.sln to open 2 Go to **Build → Rebuild Solution** 3 Open **Tools → NuGet Package Manager → Package Manager Console** and run: Update-Database 4 Press **F5** (or **Ctrl+F5**) to run	A browser opens at http://localhost:5xxx/ showing the Store home page.

VS Code / Command Line (Win/macOS/Linux)	1 In the folder with .sln file, run:
	`dotnet restore`
	`dotnet tool update --global dotnet-ef` *(if missing)*
	`dotnet ef database update`
	`dotnet run`
		Console shows a URL like http://localhost:5xxx/ — open it in your browser.

Note about the database:

A file called ***easygames.db*** is created automatically the first time you run the app. It's pre filled with sample data. Re-running the seed won't duplicate anything (it's *idempotent*).

Step 3 — Log In (Demo Account admin) or you can also create a user account

Role	Email	Password
Owner (Admin)	`owner.admin@easygames.local`	`GiveThisToLecturer123!`

Note: Owner bootstrap (env vars) was refined with ChatGPT advice (Prompt ID **BOOT-002**). See Topic 13 → AI References **(Figure R.1).**

Step 4 — Quick Feature Check (2–3 min)

Action	Path	What to Expect
🛒 **Browse & Buy**	`/Store → /Cart → /Checkout/Done`	See Subtotal, Tax, and Total.
📦 **View Your Order**	`/Orders/My`	Shows your order and "tier" badge.
📜 **Admin Products**	`Admin → Products`	Shows Price, Cost, and Margin%.
📬 **Admin Orders**	`Admin → Orders`	Shows totals/profit and lets you change status.
✉️ **Admin Email Groups**	`Admin → Email Groups`	Compose/send emails (simulated sending).
📉 **Stock Updates**	`Admin → Products` (after checkout)	Stock decreases for purchased items.

*Note: Emoji rendering system font **Segoe UI Emoji** (Windows 11)*

If there is a problem and you need to start over? 🔄 Reset / Reseed (Start Fresh)

Step	Command / Action
1️⃣ **Stop the app.**	—
2️⃣ **Delete `easygames.db`:** **Windows:** `EasyGames\easygames.db` **macOS/Linux:** `./easygames.db`	—
3️⃣ **Recreate the database:**	**Visual Studio:** `Update-Database` **Command line:** `dotnet ef database update`
4️⃣ **Run the app again.**	F5 or `dotnet run` — the Owner account and sample data will return.

Troubleshooting (Quick Fixes)

Problem	Fix (Visual Studio)	Fix (Command Line)
⚙️ Build fails	**Build → Clean Solution**, then **Rebuild**	`dotnet clean` → `dotnet build`
🗃️ Database error	`Update-Database`	`dotnet ef database update`
🔑 Login fails or data looks weird	Delete `easygames.db` and follow Reset steps	Same
🖼️ Images missing	Check folder `wwwroot/images/` exists	Same
🧩 dotnet-ef not found	`dotnet tool install --global dotnet-ef`	Restart terminal or use VS PMC

🍎 macOS/Linux Notes

All command-line steps are the same as Windows.

Just remember this : paths use forward slashes (/), e.g. ~/Dev/EasyGames.

Topic 3. Team Contract

3.1 Scope

We will deliver the HIT339 Group 1 project assessment to the agreed scope and marking criteria by extending the Assessment 2 EasyGames base into a clean, runnable ASP.NET MVC e-commerce demo for books/games/toys that covers both *Owner* and *Customer* flows. Priorities are working software, steady iteration, clear communication, and on-time delivery; rotation is encouraged so each member experiences at least two roles; the intent is a flexible, respectful, collaborative project where feedback stays about the work, not the person. The committed scope includes Owner stock CRUD with CostPrice/Supplier and calculated profit margin, User CRUD (admin-only), Email Groups (conditional SMTP/simulated), Checkout → Orders with stock decrement and price/cost snapshots surfaced in Admin Orders (list/details/status), and Customer registration, catalog browse/search, cart/checkout, and order history with a tier badge produced with clear evidence and documentation.

3.2 Communication & Responsiveness

- **Primary:** Microsoft Teams (chat + meeting notes).
- **Backup:** email if someone is offline.
- **Responsiveness SLA:** reply within *24 hours* on weekdays; weekends OK to be slower unless a *blocker* is flagged.
- Keep posts constructive, respectful and concise.

Meetings, Planning & Tracking

- **Weekly check-in:** Sundays 5:00–5:30 PM Darwin time unless otherwise rescheduled by consensus.
- **Mid-week async update:** each member can post a screenshot or short status in Teams.
- *If you can't attend* post an update before the meeting; decisions may still proceed.
- Single source *Microsoft Teams channel*.
- **Burndown:** at Sunday check-ins, we review open items and adjust scope if needed.

Table 1. Roles & Rotation Updated on 20 Oct

Role	Core responsibilities	Rotation	Backup / notes
Project Lead (GFDC)	Runs Sunday check-in; keeps Planner/board current; unblocks issues; records decisions & risks	Rotates weekly	*If absent, Tech Lead facilitates meeting*
Tech Lead — Gislene (GFDC)	Branching rules; EF/DB migrations (marshal); code quality & reviews; build health; architecture decisions; mentor	Fixed for Assessment 3	*Backup:* feature owner on duty
Orders & Checkout — GFDC	Cart → Order (demo payment) → transactional stock decrement; price/cost snapshots; /Checkout/Done; Admin Orders (list/details/status); baseline /Orders/My	Fixed per feature	Pairs with QA for acceptance checks
Email Groups — Veasna	Owner-only Email Groups; SMTP or simulated sender; compose/sent views; DI configuration	Fixed per feature	*Backup:* Alex after tiers
Customer History & Tiers — Alex	Extend /Orders/My; extract Tier service; optional filters (date/status); polish UI	Fixed per feature	*Backup:* GFDC (baseline exists)
User Admin / Reports — Vendel	Admin Users polish (e.g., role badge) or sales/profit CSV/reporting (if time permits)	Fixed per feature (reassigned)	*Backup:* GFDC; may be de-scoped if time-boxed
QA & Release Steward	Test runs; bug triage; release notes; demo rehearsal scheduling; screenshot evidence	Rotates Weeks 3–4	Can be combined with a Feature Owner role earlier

Table 2. Engineering Workflow

Practice	Standard
PR content	Include scope summary, UI screenshots (if relevant), tests affected, checklist ticks, and the Definition of Done (below).
Definition of Done (DoD)	• Compiles local, CI builds pass • Meets acceptance criteria; key edge cases tried • Tests updated/added where appropriate • UI matches house style, basic accessibility checks • Docs updated (README/notes/seeds/migrations) • ≥1 teammate has reviewed and approved
Migrations	One **"migration marshal"** to avoid EF migration conflicts; others sync before creating new migrations.

3.3 Conflict & Escalation

- Start with a direct, respectful conversation between involved members
- If unresolved within 48 hours, involve the Project Lead
- If still blocked, escalate to the *lecturer* for guidance

3.4 Deadlines & Scope Changes

- **Default:** quick discussion aiming for consensus**.**
- **No consensus in 20 minutes:** *Project Lead decides* and logs the rationale in meeting notes.
- **After Week 2:** any scope change needs a brief written *impact check* (time/risk) before approval.
- Hit internal milestones; raise risks *early* if you're slipping
- If you miss a deadline *without notice*, Lead may reassign work to protect the schedule
- Submit work early enough for *review and revision* before milestones

3.5 Milestones

Date	Milestone
Sept 22	Alex, Set up the repository. GitHub usernames/ emails added
Oct 08	Feature branches opened / integration started
Oct 19	**Feature freeze** – all PRs ready wait few days for final post
Oct 20	Final integration + report collation
Oct 23	**Submission**

Table 3. Responsible Use of AI & Academic Integrity

Topic	Agreement
Purpose	We may use AI tools to speed up drafting, formatting, grammar checks, idea validation, and non-photorealistic diagrams/placeholders. Team members remain fully responsible for accuracy, originality, citations and integrity of all submitted work.
Allowed (with review)	Grammar/style helpers (e.g., Grammarly); LLMs for brainstorming/outline/drafts (we verify/edit); Canva/Figma or similar for simple diagrams/placeholders; simple calculators or unit converters.
Not allowed	Uploading personal/sensitive data into third-party tools; letting AI produce *final* work without human verification and editing; submitting *uncited* AI-generated content or images as if entirely human-made.
Disclosure & citation	If AI-generated text/ideas are included, we *acknowledge* use in methods/notes. Any AI-generated *diagram/image* is captioned "AI-generated" with tool + year. **Example:* "Figure 2. Buyer flow (AI-generated using Canva, 2025)." Follow current APA-style guidance for AI citations (e.g., institutional library guides/APA blog).

**See References*

This project follows CDU's "Using AI Tools at CDU" guidance and the lecturer's requirement to reference AI use both in the report and in code**.**

A. Referencing AI

If any AI tool (e.g., ChatGPT) influenced design, code, tests or diagrams, include a short "AI References" subsection that lists:

- Tool citation (example):

  OpenAI. (2025). ChatGPT (GPT-5 Thinking) [Large language model]. https://openai.com
- The exact prompt IDs used (one per line), for example:

  prompt 001: build an item model

  prompt 002: build a controller to CRUD the item model

  prompt 003: generate seed data

  (continue the list as needed)

B. Referencing AI in the code (mandatory)

Where code was created or substantially modified using an AI prompt, wrap the affected region with start/end comments that include the prompt ID. Use this format in C# files:

```
// chatGPT Prompt 001 start
// Purpose: Item model generated from prompt 001, then edited for project needs.
public class Item
{
public int Id { get; set; }
public string Name { get; set; } = "";
public decimal Price { get; set; }
}
// chatGPT Prompt 001 end
```

> note: >prompt lines are short and factual; only used prompts are listed. *If all images in the project are AI-generated, add a single acknowledgement in the root `README.md` instead of repeating per image.*
Example: “All product/diagram images in this coursework are **AI-generated (Tool, 2025)**.”

Sign-off

Name	Signature	Date
Gislene FDLClancy	Gislene C	21/09/2025
Alex Phan		22/09/2025
Veasna Kuoy		21/09/2025
Vendel Gomes		

Topic 4. Requirements

4.1 Stakeholders & Goals

- **Owner (shop operator).** Manage stock; record ***costs & suppliers***; view ***margin/profit***; change ***order status; email*** customer groups.
  Success cue: Admin → Products shows ***Price/Cost/Margin%;*** Admin → Orders shows **Totals/Profit**; Email Groups works.
- **Customer (end-user).** Browse/search catalog; add to ***Cart***; complete ***Checkout***; view ***order history*** and ***tier badge***.
  Success cue: Store → Cart → Checkout/Done; ***/Orders/My*** lists orders with a badge.
- **Marker/Lecturer.** Run locally with minimal setup; verify each requirement quickly via routes + screenshots**.**
  Success cue: Follow Install steps; figures in EasyGames/Docs/evidence/ match the traceability table.
- **Developers/Team.** Keep main buildable; ship small, reviewable slices; maintain traceability from requirements to code and evidence.
  Success cue: Small PRs, early ***freeze (T–4),*** and an extended RTM in Topic 4.

4.2 MoSCoW Priorities

- **Must:** FR-01…FR-08; NFR-01…NFR-03
- **Should:** FR-09…FR-10; NFR-04…NFR-05
- **Could:** CSV/Reports export
- **Won't (now):** Multi-shop/POS, payment gateway.

4.3 Functional Requirements (FR)

- **FR-01 Catalog (M):** Browse products; filter by *Category*; search by *Title*.
- **FR-02 Cart (M):** Add/update/remove items; totals recalculated; session persists.
- **FR-03 Checkout → Order (M):** Submit checkout; land on /Checkout/Done showing **Subtotal/Tax/Total**; cart clears.
- **FR-04 Stock Decrement (M):** Reduce Product.StockQty atomically with order write.

- **FR-05 Profit Snapshots (M):** Snapshot UnitPrice and CostPriceSnapshot; compute Order Subtotal/Tax/Total/Profit.
- **FR-06 Admin Products (M):** CRUD with Cost and Supplier; show Margin/Margin%.
- **FR-07 Admin Orders (M):** List + Details + status flow Placed → Paid → Shipped (POST + antiforgery).
- **FR-08 Customer Order History (M): /Orders/My** table + lifetime tier badge (baseline).
- **FR-09 Email Groups (S):** Owner sends emails to All (optionally by tier); SMTP or Simulated sender.
- **FR-10 User Accounts (S):** Register, Login/Logout; Owner-only Admin.

4.4 Non-Functional Requirements (NFR)

- **NFR-01 Setup Simplicity (M):** .NET 8 + SQLite; Update-Database then run.
- **NFR-02 Reliability (M):** Checkout + stock updates in a single EF Core transaction.
- **NFR-03 Security Baseline (M):** Model validation, Razor encoding, anti-forgery tokens, Owner-only Admin.
- **NFR-04 Performance (S):** Typical pages render < 1s on dev machine.
- **NFR-05 Maintainability (S):** Conventional MVC; services via DI; idempotent seeding.

4.5 Assumptions & Out-of-Scope

- Demo checkout (no payment provider).
- Single store, single currency, no shipping calculator.
- Email defaults to Simulated unless SMTP is configured.

4.6 Table 4. Traceability (FR → Evidence)

FR	Proof (page/route or artifact)	Screenshot/File	Evidence Location
FR-01 Catalog	/Store (search + category)	01_store.png	EasyGames/Docs/evidence/01_store.png
FR-02 Cart	/Cart (add/update/remove)	02_added_to_cart.png	EasyGames/Docs/evidence/02_added_to_cart.png
FR-3 Checkout/Done	/Checkout/Done shows Subtotal/Tax/Total	03_checkout_done.png	EasyGames/Docs/evidence/03_checkout_done.png
FR-04 Stock Decrement	Stock drops after order	04_placed_order.png	EasyGames/Docs/evidence/04_placed_order.png
FR-05 Profit Snapshots	Admin → Orders (Totals/Profit; details show snapshots)	05_my_order.png	EasyGames/Docs/evidence/05_my_order.png
FR-06 Admin Products	Admin → Products shows Price/Cost/Margin%	07_admin_products_margin.png	EasyGames/Docs/evidence/07_admin_products_margin.png
FR-07 Order Status Flow	Admin → Orders → status buttons (Placed→ Paid→ Shipped)	*08_admin_orders_status_Shipped.png*	EasyGames/Docs/evidence/

Note : All images are stored under EasyGames/Docs/evidence/ inside the application repository

4.6a Table 5. Extended Traceability Matrix

FR	Ass2 Baseline?	Ass3 Additions (what's new)	Owner	Proof (route / artifact)	Screenshot/File	Evidence Location	Key Code Paths (files/folders)
FR-01 Catalog	✅ Store + search/category	(no change)	GFDC	/Store	01_store.png	EasyGames/Docs/evidence/01_store.png	/Controllers/HomeController.cs, /Views/Home/Index.cshtml
FR-02 Cart	✅ Session cart	Minor polish only	GFDC	/Cart	02_added_to_cart.png	EasyGames/Docs/evidence/02_added_to_cart.png	/Controllers/CartController.cs, /Services/CartService.cs, /Views/Cart/*
FR-03 Checkout → Order	⛔ shell only	✅ Persist Order/OrderItems, show Subtotal/Tax/Total, clear cart	GFDC	/Checkout/Done	03_checkout_done.png	EasyGames/Docs/evidence/03_checkout_done.png	/Controllers/CheckoutController.cs, /Services/CheckoutService.cs, Migrations/*Order*
FR-04 Stock Decrement (atomic)	⛔	✅ Decrement Product.StockQty in same transaction	GFDC	After placing order	04_placed_order.png	EasyGames/Docs/evidence/04_placed_order.png	/Services/CheckoutService.cs (transaction), /Data/ApplicationDbContext.cs
FR-05 Profit Snapshots	⛔	✅ Snapshot UnitPrice + CostPriceSnapshot; compute Subtotal/Tax/Total/Profit	GFDC	Admin Orders list/details	*(added)* admin_orders_profit.png	EasyGames/Docs/evidence/admin_orders_profit.png	Models/Order.cs, Models/OrderItem.cs, Migrations/*AddOrderTotalsAndSnapshots*
FR-06 Admin Products (cost/supplier/margin)	⛔	✅ CostPrice, Supplier, Margin% in list + forms	GFDC	Admin → Products	*(added)* admin_products_margin.png	EasyGames/Docs/evidence/admin_products_margin.png	/Areas/Admin/Views/Products/*,/Areas/Admin/Controllers/ProductsController.cs, Models/Product.cs

FR-07 Admin Orders & Status	⛔	✅ List/Details + Placed→Paid→Shipped (POST + antiforgery)	GFDC	Admin → Orders	*(added)* admin_orders_status.png	EasyGames/Docs/evidence/admin_orders_status.png	/Areas/Admin/Controllers/OrdersController.cs, /Areas/Admin/Views/Orders/*
FR-08 Customer History + Tier	⛔	✅ /Orders/My table + tier badge (baseline)	GFDC (baseline)	/Orders/My	05_my_order.png	EasyGames/Docs/evidence/05_my_order.png	/Controllers/OrdersController.cs, /Views/Orders/My.cshtml, (optional) /Services/TierService.cs
FR-09 Email Groups	⛔	✅ Owner can send to All (SMTP or Simulated)	**Veasna**	Admin → Email Groups	*(added)* admin_email_groups.png	admin_email_groups_compose.png, admin_email_groups_sent.png	EasyGames/Docs/evidence/admin_email_groups_compose.png EasyGames/Docs/evidence/admin_email_groups_sent.png
FR-10 Accounts (Register/Login)	✅	(A2 baseline)	GFDC	/Account/Register, /Account/Login	*(optional)* account_register.png	EasyGames/Docs/evidence/account_register.png	/Controllers/AccountController.cs, Models/User.cs, Services/AuthService.cs

(FR → Evidence, Owner, Code Paths)

4.7 Assessment2 to A3 Additions Explanation

Evidence location for all delta images:

EasyGames/Docs/evidence/<file>.png

For Assessment 3 additions (summary)

Extended the Assessment2 base with four capabilities: (1) Owner stock details (CostPrice, Supplier) and Margin% in Admin Products; (2) real Orders & Inventory — checkout persists Order/OrderItems, decrements stock, and computes Subtotal/Tax/Total/Profit; (3) Customer history with a tier badge on /Orders/My; (4) Email Groups (Owner) with simulated send by default (SMTP if configured). Evidence: Admin Products (margin), Admin Orders (totals/profit + status), My Orders (tier), and Email Groups. Screenshots live in EasyGames/Docs/evidence/.

SRS coverage: This section is just to defines the Software Requirements Specification for EasyGames: FR-01…FR-10 and NFR-01…NFR-05, assumptions and scope limits, plus an extended traceability matrix mapping each FR to routes, code paths, and screenshots (all under EasyGames/Docs/evidence/). The SRS is multi page by design to include detailed tables and evidence links.

Topic 5 — Team Approach

5.1 Approach Summary (what we actually did)

We started with a walking skeleton strategy, prioritising checkout into real orders so the app worked end-to-end early. When team capacity dipped, the Tech Lead (GFDC) consolidated work and finished any blocking slices to keep the main branch runnable. We used feature branches with pragmatic code reviews small pull requests and brief async checks before merging. An early code freeze was set four days before submission, after which only low risk, reviewed changes were allowed. Finally, we kept tight scope discipline: ideas like novelty features or multi-shop support were moved to the “value-adding” backlog, while the last days focused on reliable evidence and the report.

Table 6. How work flowed

Step	Practice	Acceptance check
Plan	Pick the **smallest slice** tied to the SRS	Has a route/view + a screenshot to prove it
Build	Branch feature/<area>-<short>-<initials>; coordinate migrations	Solution compiles; smoke tests pass
Review	Post PR with 1–2 images + short notes	Reviewer launches the route; UI and totals verified
Merge	**Squash to main by Tech Lead** (migration marshal)	main still builds; demo path intact
Freeze	T–4 days; scope locked on 21 October	No schema churn; docs/screens updated instead

5.4 Roles & later Hand-offs

- **Tech Lead — GFDC:** migration marshal; delivered ***Stock/Costs + Margin***, ***Orders/Checkout (transactional)***, ***Admin Orders***, *and* ***/Orders/My baseline***; kept build green.
- **Email Groups — Veasna:** Owner-only compose/send with ***conditional DI*** (SMTP or Simulated). Merged cleanly ***(PR #6)***.
- **Customer History & Tiers — Alex:** planned polish; baseline tier badge implemented by Tech Lead; tiny follow ups requested post freeze.
- **User Admin / Reports — Vendel:** late pivot; deferred to *Value Adding* due to time.
- **Repo setup attribution:** early chat shows *Alex* organised GitHub repo/team onboarding; Ass3 code changes visible mainly from GFDC and Veasna.

5.5 Definition of Done (DoD)

- Compiles locally; acceptance criteria on the demo path pass.
- UI consistent enough for marking; labels/titles present.
- *Evidence files saved to* EasyGames/Docs/evidence/ and referenced in the doc.
- Any DB change coordinated via single migration marshal (Tech Lead).

5.6 Risk Management (what worked)

- **Capacity gaps:** centralised critical work under Tech Lead to avoid a broken demo.
- **Integration risk:** tiny PRs + early freeze prevented last-minute regressions.
- **Email variability:** default Simulated sender; SMTP only when configured.
- **Scope creep:** multi-shop and other novelty items explicitly de-scoped for A3; logged in Value Adding.

5.7 Tools & practices

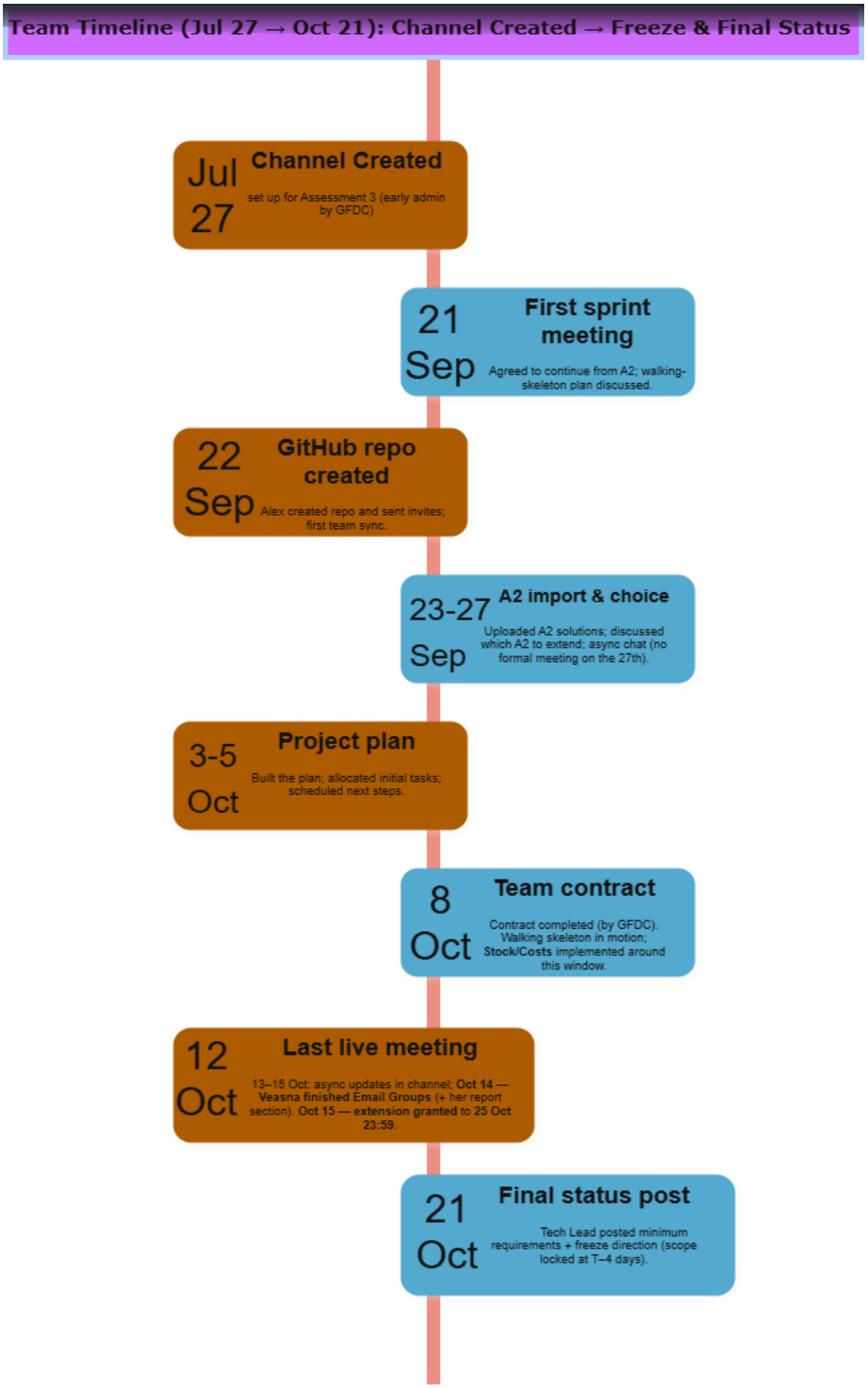
- GitHub (branches/PRs, squash merges), Visual Studio / VS Code, SQLite, ASP.NET MVC + EF Core.
- Screenshots stored under EasyGames/Docs/evidence/; figure captions reference exact paths.

5.8 Lessons learned (transparent and direct)

- **Freeze earlier** on group work; protect the demo path first.
- **One migration marshal** avoids EF conflicts.
- **Evidence-driven PRs** (screenshots + short notes) make marking and integration easier.
- **Clear "safe tasks" list** helps contributors add value late without risking stability.

Topic 6 — Timeline

Figure T1. Team timeline: channel creation to freeze (Jul 27 → Oct 21). Stored at EasyGames/Docs/evidence/team_timeline.png.



Created by (GFDLC) in diagrams.net, 2025.

6.1 Architecture Decisions (ADR highlights)

6.1.1 ADR-001 Routing (Conventional MVC + Admin Area)

- **Where in code:**
  - /Program.cs (endpoint mapping)
  - /Areas/Admin/* (Area folders, controllers, views)
- **How to verify:** browse /Admin → routes resolve to admin controllers/views.

6.1.2 ADR-002 Data (EF Core + SQLite)

- **Where in code:**
  - /Program.cs → AddDbContext<ApplicationDbContext>(UseSqlite(...))
  - /Data/ApplicationDbContext.cs (DbSets, configuration)
- **How to verify:** first run creates easygames.db in the project root.

6.1.3 ADR-003 Price/Cost Snapshots (historic profit)

- **Where in code:**
  - /Models/OrderItem.cs → UnitPrice, CostPriceSnapshot, LineTotal, LineProfit
  - Migrations/*AddOrderTotalsAndSnapshots*
- **How to verify: Admin → Orders → Details** shows Unit Price and Cost Snapshot columns.

6.1.4 ADR-004 Transactional Checkout (order + stock together)

- **Where in code:**
  - /Services/CheckoutService.cs → single EF transaction; writes Order/OrderItems and decrements Product.StockQty
- **How to verify:** place an order; stock reduces; /Checkout/Done shows totals.

6.1.5 ADR-005 Security Baseline (cookie auth, Owner-only Admin)

- **Where in code:**
  - /Program.cs → AddAuthentication().AddCookie(...)
  - /Controllers/AccountController.cs (login/logout)

- /Areas/Admin/* guarded by Owner authorization (filter/attribute under /Filters/* if used)

- **How to verify:** Admin pages require Owner login.

6.1.6 ADR-006 Email DI (SMTP or Simulated)

- **Where in code:**
  - /Services/IEmailService.cs, /Services/SmtpEmailService.cs, /Services/NullEmailService.cs, /Services/SmtpOptions.cs
  - /Program.cs → registers Smtp when configured, otherwise *NullEmailService*
- **How to verify:** Admin → Email Groups shows Simulated send unless SMTP is set.

6.2 Risks & Mitigations

- **Capacity variance:** Centralised blocking work under Tech Lead → kept main runnable.
- **Integration churn:** Early freeze (T–4) + squash merges → avoided last-minute breaks.
- **Scope creep (multi-shop):** Moved to Value Adding; not in A3 deliverable.
- **Email config variability:** Default Simulated sender; SMTP optional.

6.3 Project Management Links

GitHub (code & PRs).

Link: https://github.com/alex-phan/easygames.git

Access: read-only. Evidence screenshots also live in EasyGames/Docs/evidence/.

Microsoft Teams (channel & files).

Channel link: https://tinyurl.com/2vym3kx7

Topic 7 — Project Task Allocations (Plan vs Actual)

7.1 Short summary

During Assessment 3, capacity dipped near semester end, so the Tech Lead consolidated blocking slices (orders/checkout, admin orders, baseline history) to keep main runnable. Veasna delivered Email Groups via PR #6. Alex's planned polish on tiers remained minimal (baseline shipped by GFDC). Vendel attended meetings; no A3 code changes landed, so User Admin/Reports moved to Value Adding.

Table 7. Billable Tasks — allocations only (hours not tracked)

Area / Feature	Planned Owner	Actual Implementer	Status	Notes
Stock & Costs + Margin (Admin → Products)	Vendel	GFDC	✅ Done	CostPrice/Supplier fields, Margin%.
Orders & Checkout (transactional) + Stock decrement + Profit snapshots	GFDC	GFDC	✅ Done	CheckoutService, Order/OrderItem, Subtotal/Tax/Total/Profit.
Admin Orders (list/details/status)	GFDC	GFDC	✅ Done	Status flow Placed→Paid→Shipped.
Customer history /Orders/My + tier badge (baseline)	Alex	GFDC (baseline)	✅ Done	Tier polish invited for small PR.
Email Groups (Owner-only; SMTP/Simulated)	Veasna	Veasna	✅ Done (PR #6)	Conditional DI, compose/sent views.
User Admin polish / Reports/CSV	Vendel	—	⏸ Deferred	Documented under Value Adding.

Tasks are treated as discrete billable units; allocations and status are recorded. Individual hours were not tracked.

7.2 Contributor Feature Report —by Veasna Kuoy: Email Groups (Admin-only)

Executive summary

This feature adds an Owner-only "Email Groups" screen to the Admin area. It lets the Owner send an announcement to all users or to a specific tier (Bronze/Silver/Gold/Platinum). For marking, the app defaults to a simulated mode that performs all logic without contacting an SMTP server; enabling real SMTP only requires configuration. The UI confirms the recipient count and clearly labels whether the send was simulated or real.

Requirement

- Owner can compose and send an email to all users or to a selected customer tier.
- Feature is restricted to the Owner role (admin-only).
- Provide a simulated mode when SMTP is not configured so tutors can test safely.
- Show final status with recipient count and whether the send was simulated.

Design overview

Security

The Emails controller is area-scoped and protected via role-based authorization: [Authorize(Roles = "Owner")].

Abstractions

IEmailService defines SendAsync, SendBulkAsync, and RenderViewToStringAsync. SmtpEmailService implements real SMTP sending and Razor view rendering with TLS and explicit credentials. NullEmailService provides a safe, simulated send for demo/marking.

Dependency injection

Program.cs registers SmtpEmailService only when configuration contains Smtp:Host, Smtp:Username, Smtp:Password, and Smtp:FromEmail; otherwise it registers NullEmailService for simulated sends.

Templating

The system attempts to render /Views/Shared/EmailTemplates/BulkMessage.cshtml; if not present, it falls back to a safe HTML body constructed from the provided Subject and Message.

Files and locations

Areas/Admin/Controllers/EmailsController.cs

Areas/Admin/ViewModels/EmailGroupVM.cs

Areas/Admin/Views/Emails/Compose.cshtml

Areas/Admin/Views/Emails/Sent.cshtml

Services/IEmailService.cs

Services/SmtpEmailService.cs

Services/NullEmailService.cs

Services/SmtpOptions.cs

Program.cs (conditional DI)

UI / UX

The feature is visible to the Owner role only and is accessible via the main navbar.

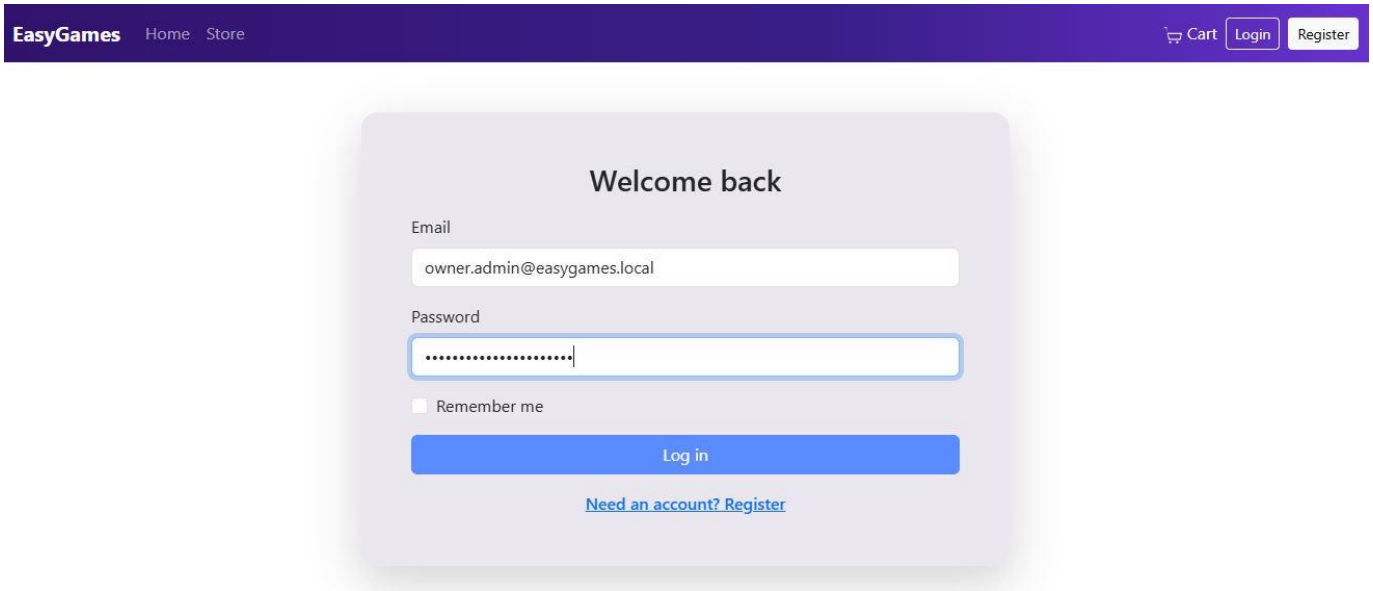


Figure 1. Admin login (Owner only).

The navbar surfaces a dedicated Email Groups entry for discoverability

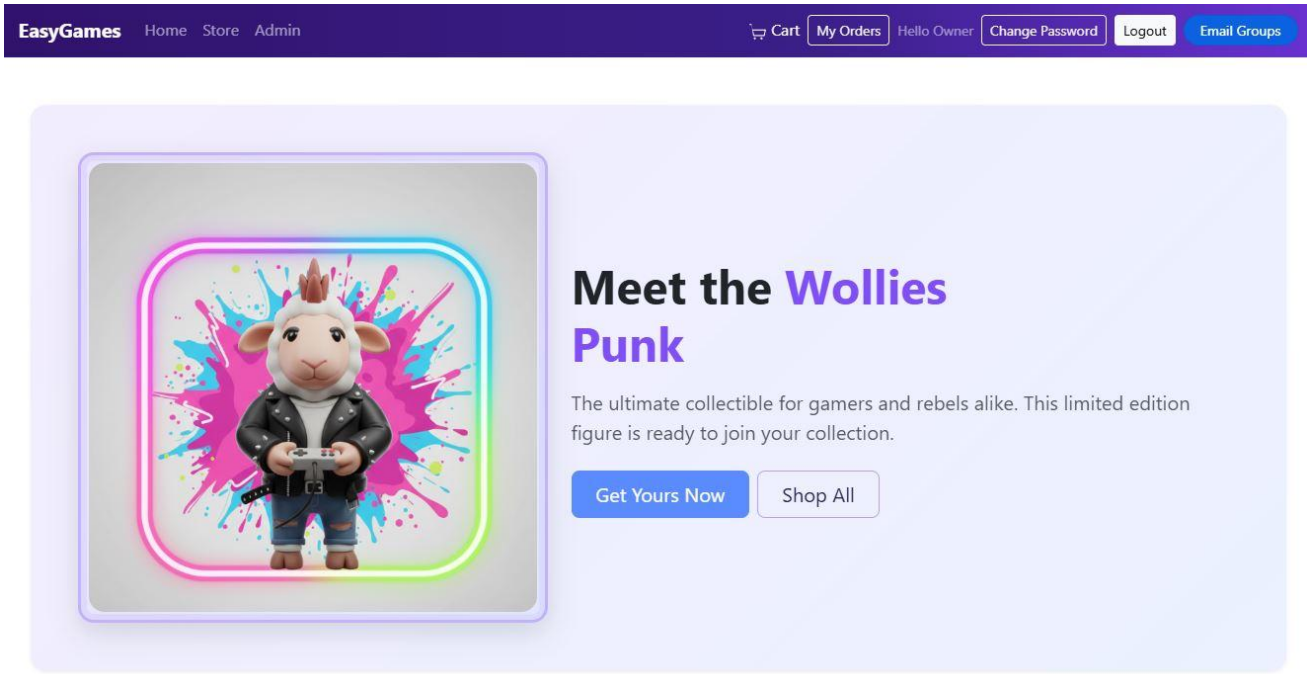


Figure 2. Navbar showing "Email Groups" (Owner only).

The compose page uses a Bootstrap card with fields for Subject and Message, a 'Send to all users' checkbox, a Tier dropdown (Bronze/Silver/Gold/Platinum), and Send/Clear actions.

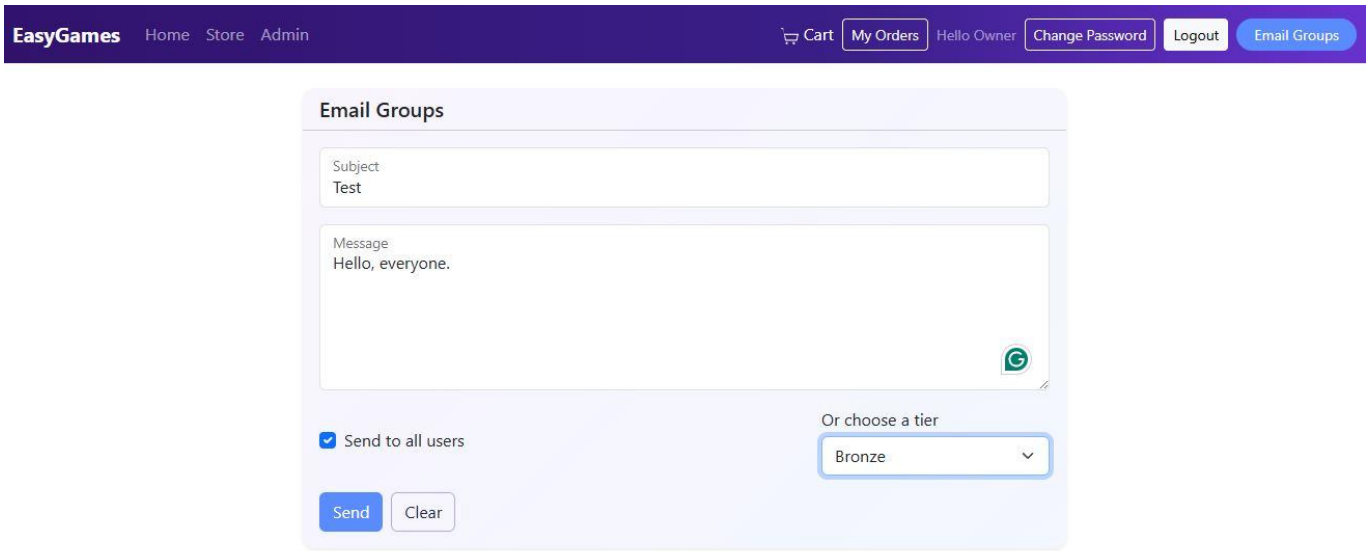


Figure 3. Email Groups — compose form.

On submit, the app redirects to an Email status page that displays the recipient count and indicates whether the send was Real (SMTP configured) or Simulated (no SMTP configured).

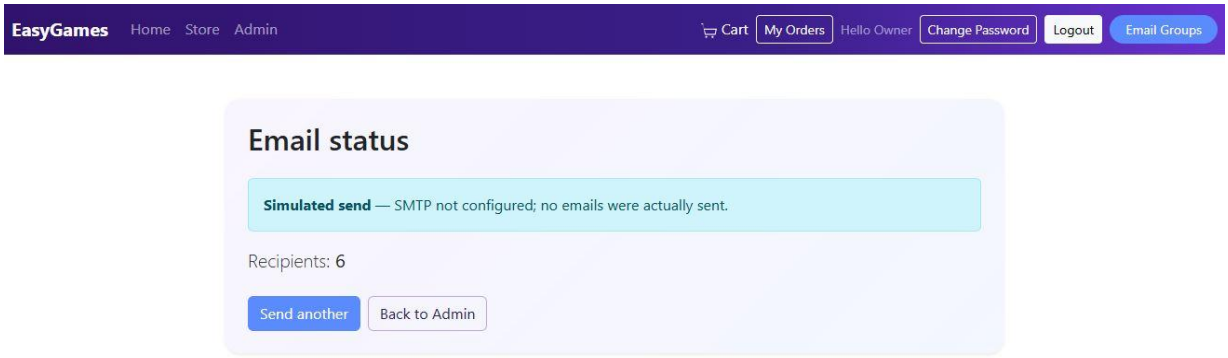


Figure 4. Email status — simulated mode (no SMTP configured).

Controller flow

- GET Compose → returns View(new EmailGroupVM()).
- POST Compose → validates Subject and Message.
- Queries Users for distinct, non-empty emails.
- If NOT sending to all and a tier is selected → applies tier filter (EF.Property<int?>("Tier")).
- Renders email HTML via Razor template with encoded fallback.
- Tries SendBulkAsync: SMTP present → Success; SMTP missing → treated as Simulated success.
- Redirects to Sent(count, simulated).

Configuration (optional for real sending)

Default behaviour: Simulated mode (no Smtp section).

To enable real sending, add the following to User Secrets or appsettings.Development.json:

```
{
  "Smtp": {
    "Host": "sandbox.smtp.mailtrap.io",
    "Port": 2525,
    "UseSsl": true,
    "FromName": "EasyGames",
    "FromEmail": "no-reply@easygames.test",
    "Username": "<MAILTRAP_USERNAME>",
    "Password": "<MAILTRAP_PASSWORD>"
```

```
 }
}
```

Testing & acceptance

Test	Steps	Expected	Result
Happy path (simulated)	Owner fills Subject + Message, clicks Send	Status shows Simulated send and recipient count	Pass
Send to tier only	Uncheck 'Send to all' → choose Bronze → Send	Status shows only Bronze recipients counted	Pass
Validation	Submit with empty Subject or Message	Validation errors display under fields	Pass
AuthZ	Access feature as non-Owner	Access denied / link hidden	Pass

Limitations & future work

- Replace EF.Property-based tier filter with a strong-typed User.Tier column when available.
- Batch/queue large sends (e.g., Chunk(200) or background jobs).
- Production deliverability: SPF/DKIM/DMARC and bounce/complaint handling.
- Optional WYSIWYG editor and template picker for admins.

Contribution summary

Implemented Email Groups (Admin-only): controller, view model, views, email services (real + simulated), conditional DI, and configuration guidance. Ensured safe simulated mode for marking and provided a clear status page.

Topic 8 — Meeting Minutes (highlights only + log from MS Project)

8.1 Early Meeting summary

05 Oct 2025 (≈22 min)

Attendees: Gislene · Alex · Vendel *(Viesna absent; recording available)*

Decisions

- Base code: continue from Gislene's A2 EasyGames (most complete).
- Brand: keep purple glass; allow minor blend-ins.
- Repo: use alex-phan/easygames; branch/PR workflow.
- Workstyle: vertical slices per person; feature branches.
- Scope baseline (from A2): Auth; Store (search/category); Cart + checkout shell; Admin scaffold; themed images.

A3 features required

- Owner stock details: CostPrice, Supplier, Margin%.
- Orders & inventory: persist Order/OrderItems, decrement stock, compute totals/profit.
- Customer history & tiers + email groups.
- User management (Owner): CRUD, roles, reset.
- Docs/report: contract, architecture notes, testing, final PDF.

Provisional task split

- Gislene: stock fields & migration foundation.
- Alex: customer history, tiers, email groups.
- Vendel: user management or orders/inventory.
- Viesna: assign after sync.

Availability note

- Gislene busy until Tue; push stock foundation Tue night.

Action items

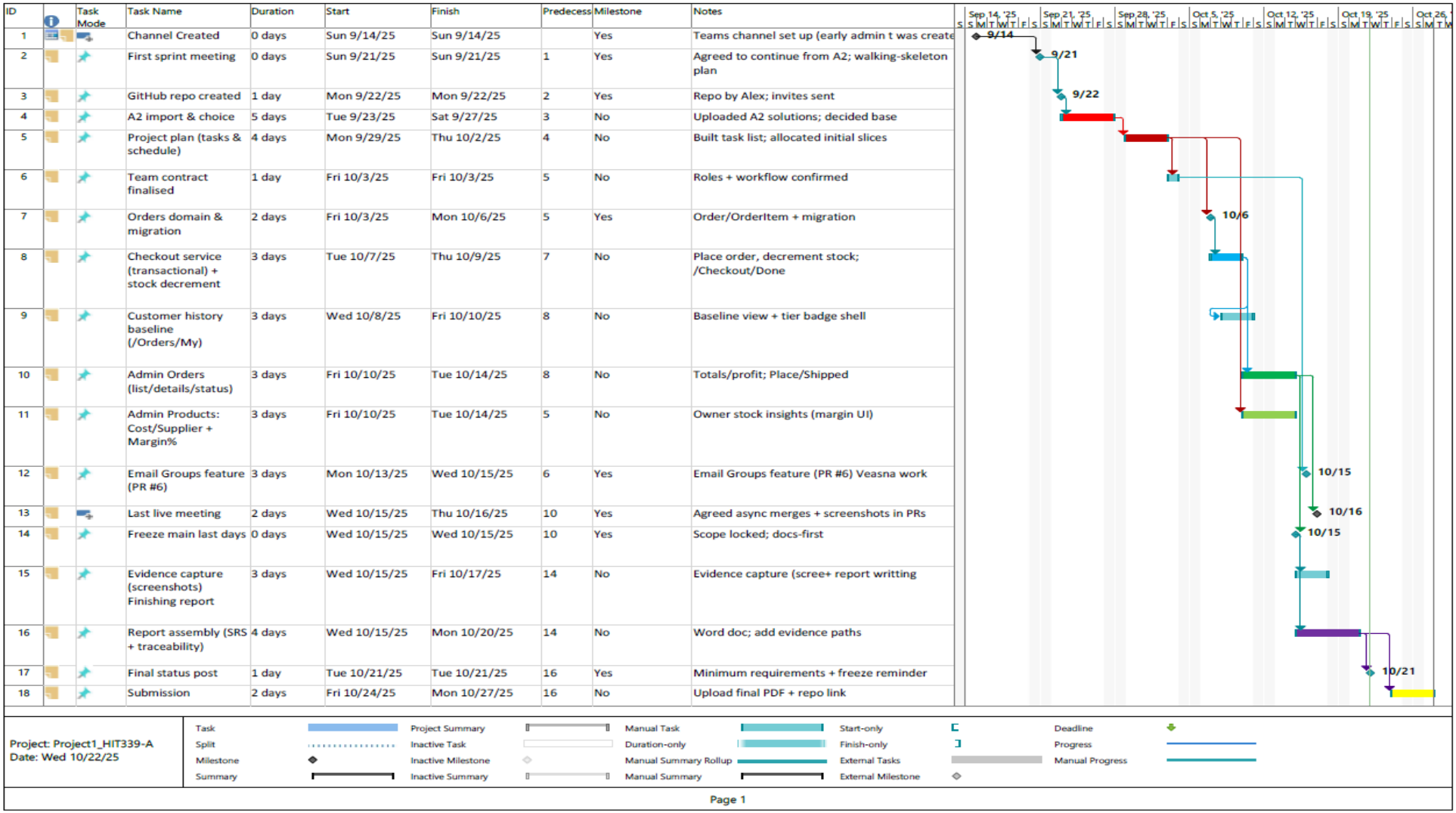
- **Repo push:** base project to alex-phan/easygames.
- **Branch created:** feature/stock-crud.
- Add .gitignore, clean bin/obj.
- Create issues/branches for each slice (names below).
- Short README for team setup + run steps.

Suggested branches/issues

feature/stock-crud · feature/orders-inventory · feature/customer-history-tiers · feature/user-management · docs/project-docs

Note: Below is **EasyGames Project Plan** Gantt chart showing tasks from late Sep to Oct: orders domain, transactional checkout, customer history baseline, Admin Orders, Admin Products (Margin), Email Groups PR #6, freeze at T–4, evidence capture, report, submission.

Figure P1. EasyGames Project Plan (Gantt)



ID	Task Mode	Task Name	Duration	Start	Finish	Predecess	Milestone	Notes
1		Channel Created	0 days	Sun 9/14/25	Sun 9/14/25		Yes	Teams channel set up (early admin t was create
2		First sprint meeting	0 days	Sun 9/21/25	Sun 9/21/25	1	Yes	Agreed to continue from A2; walking-skeleton plan
3		GitHub repo created	1 day	Mon 9/22/25	Mon 9/22/25	2	Yes	Repo by Alex; invites sent
4		A2 import & choice	5 days	Tue 9/23/25	Sat 9/27/25	3	No	Uploaded A2 solutions; decided base
5		Project plan (tasks & schedule)	4 days	Mon 9/29/25	Thu 10/2/25	4	No	Built task list; allocated initial slices
6		Team contract finalised	1 day	Fri 10/3/25	Fri 10/3/25	5	No	Roles + workflow confirmed
7		Orders domain & migration	2 days	Fri 10/3/25	Mon 10/6/25	5	Yes	Order/OrderItem + migration
8		Checkout service (transactional) + stock decrement	3 days	Tue 10/7/25	Thu 10/9/25	7	No	Place order, decrement stock; /Checkout/Done
9		Customer history baseline (/Orders/My)	3 days	Wed 10/8/25	Fri 10/10/25	8	No	Baseline view + tier badge shell
10		Admin Orders (list/details/status)	3 days	Fri 10/10/25	Tue 10/14/25	8	No	Totals/profit; Place/Shipped
11		Admin Products: Cost/Supplier + Margin%	3 days	Fri 10/10/25	Tue 10/14/25	5	No	Owner stock insights (margin UI)
12		Email Groups feature (PR #6)	3 days	Mon 10/13/25	Wed 10/15/25	6	Yes	Email Groups feature (PR #6) Veasna work
13		Last live meeting	2 days	Wed 10/15/25	Thu 10/16/25	10	Yes	Agreed async merges + screenshots in PRs
14		Freeze main last days	0 days	Wed 10/15/25	Wed 10/15/25	10	Yes	Scope locked; docs-first
15		Evidence capture (screenshots) Finishing report	3 days	Wed 10/15/25	Fri 10/17/25	14	No	Evidence capture (scree+ report writting
16		Report assembly (SRS + traceability)	4 days	Wed 10/15/25	Mon 10/20/25	14	No	Word doc; add evidence paths
17		Final status post	1 day	Tue 10/21/25	Tue 10/21/25	16	Yes	Minimum requirements + freeze reminder
18		Submission	2 days	Fri 10/24/25	Mon 10/27/25	16	No	Upload final PDF + repo link

Gantt for A3 sequencing: showing orders-first work, admin features, Email Groups PR #6, early freeze, then evidence/report and submission

Topic 9 — Strengths & Weaknesses

9.1 Strengths

The app reached end to end stability early and stayed reliable to submission. Checkout is transactional (stock drops, price/cost snapshots persist), Admin has practical views (Orders totals/profit/status; Products cost/supplier/margin%), and Email Groups works safely in simulated mode. Evidence is linked under EasyGames/Docs/evidence/.

9.2 Weaknesses

Contribution was uneven toward the end of semester. The Tech Lead completed several blocking items, and some polish was deferred (user-admin, CSV/reporting, multi-shop). Automated tests are light and UI polish is mixed.

9.3 Teamwork context

While the team repository's commit history shows heavier implementation by the Tech Lead (Assessment 2 baseline owner), this result reflects time pressure and end of semester constraints rather than a lack of effort or intent from the other team mates. Throughout the project the team remained active on Teams and attended meetings, often discussing progress late at night while balancing other units, work, and personal commitments. The workload distribution naturally shifted toward whoever could push final builds and testing, but coordination, communication, and shared decision-making were consistent. With more time, contributions across all members would have been broader and more visible in commits. Please consider the final submission as a genuine team effort delivered cooperatively under challenging time and study conditions.

9.4 What we'd change next time

Freeze even earlier, publish a small list of "safe tasks" for late contributions, and aim for one tiny PR per person per day. Add basic tests for tier thresholds and order maths, and time-box novelty behind feature flags so anything not ready by a week out moves to Value Adding.

Topic 10 — Software Meets Requirements

10.1 Statement.

Our Assessment 3 build meets functional requirements **FR-01 → FR-10** and the key non functional items from the SRS. I've kept the proof simple: each requirement is mapped in the Traceability Table (Topic 4.6) and backed by clean screenshots in EasyGames/Docs/evidence/. If you prefer to verify it live, the app follows conventional MVC routes and each flow can be checked in under a minute. For more information see Ass2 Readme File inside the application.

Quick guide path for (check).

- **Store** → /Store (Fig. 1)
- **Cart** → /Cart (Fig. 2)
- **Checkout** → place order, then Done at /Checkout/Done showing Subtotal/Tax/Total (Fig. 3)
- **Inventory update** → stock is decremented after purchase (Fig. 4)
- **My Orders + Tier badge** → /Orders/My (Fig. 5)
- **Admin Products** (Cost, Supplier, Margin%) → /Admin/Products (Fig. 6)
- **Admin Orders** (list/details with totals & profit) → /Admin/Orders (Figs. 7–8)
- **Email Groups** (compose/send to All or by Tier) → /Admin/EmailGroups (Figs. 9–10)

Topic 11 — Quality of Source Code

The codebase follows a conventional ASP.NET MVC layout with clear separation between Controllers, Views, Models, and small Services. Admin lives under /Areas/Admin, which keeps owner-only features grouped and easy to secure. Data access uses EF Core with a scoped **ApplicationDbContext**; migrations are coordinated by a single "marshal" to avoid conflicts, and seeding is idempotent so first run is predictable.

Safety is handled by server side model validation, Razor HTML encoding by default, and anti-forgery tokens on POST actions. Owner-only pages require authentication and role checks, and email sending is wired through DI so the default is a simulated sender unless SMTP is configured this prevents accidents during marking. Checkout executes inside one database transaction so orders, totals and stock updates commit together or not at all.

Readability is kept by small controllers and thin views; calculation logic sits in services (e.g., checkout, email), and comments use a simple “note to self” style rather than noisy blocks. Names are consistent, routes are conventional, and files stay where a marker would expect to find them. Images and evidence are organised under EasyGames/Docs/evidence/ and referenced in the report with exact paths.

Known technical debt is modest and documented: user-admin polish and CSV/reporting are deferred; tier logic is service-ready but kept simple for submission; automated tests cover the happy path more than edge cases. These are listed in Value Adding so future work is obvious and low risk.

Topic 12 — Value Adding (proposed) & Conclusion

No extra features beyond the already documented requirements were added after the ***T–4 freeze***. The items below are proposed future work only (documented for the marker; not shipped in Ass3).

Proposed :

CSV/reporting on Admin Orders (export + simple daily totals chart).

Identity migration (built-in reset/lockout/confirm).

Tier-driven email targeting with small Razor templates.

User-admin polish (role badge, reset flow, filters).

Quality ops (basic unit tests for order maths/tier thresholds; alt-text/focus states; publish profile).

Declared out of scope: multi-shop/POS.

Conclusion

This build met the core functional goals and delivered a stable end-to-end flow store, cart, checkout, orders, admin stock with cost/margin, profit snapshots, and email groups, backed by clear evidence. Given more time, we’d polish the UX, deepen reporting and add automated tests to lock in reliability. Overall, we’re satisfied with the outcome and confident it meets the assessment requirements while leaving a clear path for future improvements.

Topic 13 — AI References & Evidence

13.1 Reflection, Tool & Prompt IDs

Reflection: *state/alignment* problem

I spent ~2 hours on this because the failure looked like a simple login issue but was actually a configuration/state tangle. Symptoms were misleading (Owner not promoted, Admin link missing), while the real causes were *stale DB data*, a *cached user session*, and env-vars attached to the wrong VS launch profile. The one time seeder also made it harder to reproduce after the first run. I worked through it by aligning the correct EasyGames profile, logging the four bootstrap env-vars to confirm they were present, deleting easygames.db and re-applying migrations, normalising email compares **(Trim().ToLowerInvariant()),** and signing in again in a fresh browser session. Once I saw the state clearly, the fix took minutes lesson learned: treat this as a ***state/alignment*** problem, not a coding bug.

Tool
OpenAI. (2025). *ChatGPT (GPT-5 Thinking)* [Large language model]. https://openai.com

Prompt IDs used Summary
BOOT-002 — Visual Studio Launch Profiles: one-time Owner bootstrap via environment variables (promote/rename/set password)

Scope note:

This solution was reached after *several iterative prompts*. Early responses produced incorrect or incomplete configurations, and the final working setup emerged only after refining the instructions through multiple follow-ups.

BOOT-002a — one-time Owner bootstrap via VS *Launch Profiles → Environment variables*

BOOT-002b — Set the exact env var names: EG_OWNER_EMAIL_PRIMARY, EG_BOOTSTRAP_OWNER_FROM, EG_BOOTSTRAP_OWNER_TO, EG_BOOTSTRAP_SET_OWNER_PASSWORD

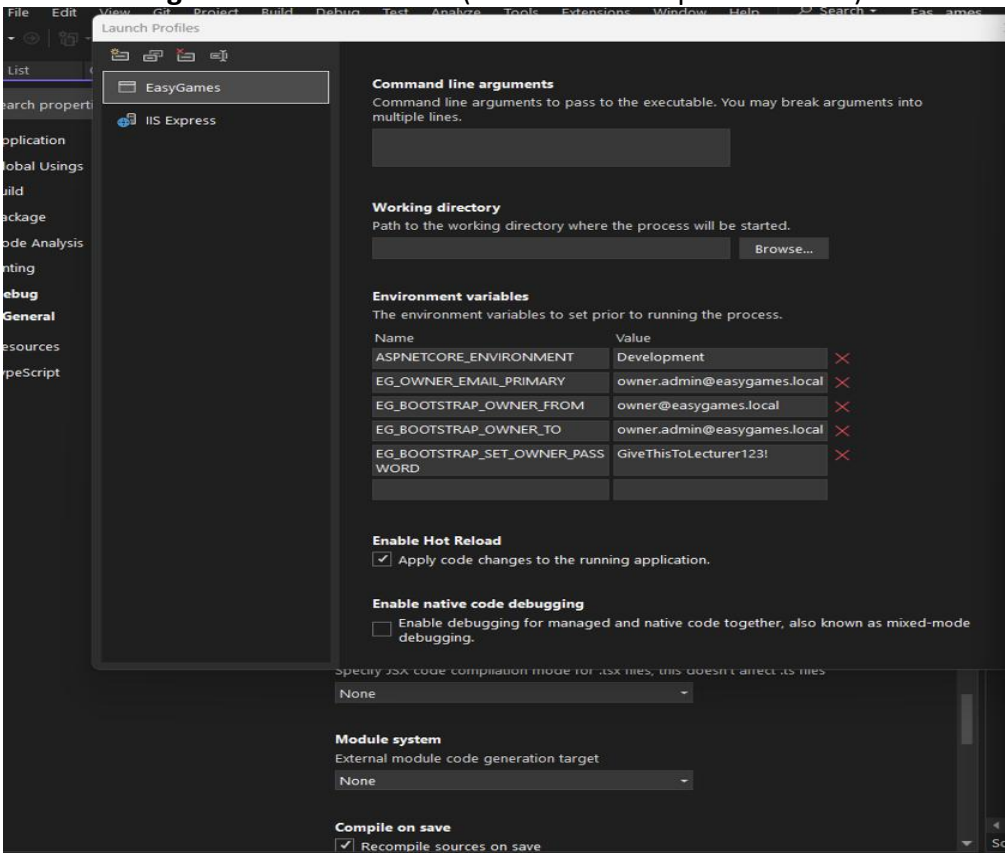
BOOT-002c — "Run once then clear" behaviour so bootstrap doesn't repeat

BOOT-002d — Troubleshoot wrong profile (IIS Express vs EasyGames)

BOOT-002e — Troubleshoot email conflict (FROM/TO rename logic)

BOOT-002f — Final verify: login as Owner, see Admin

Figure R.1 — BOOT-002 (Owner bootstrap via env vars)



Code location

`/EasyGames/Data/DataSeeder.cs` → method `ApplyOwnerFixupsAsync()`

// chatGPT Prompt BOOT-002 start

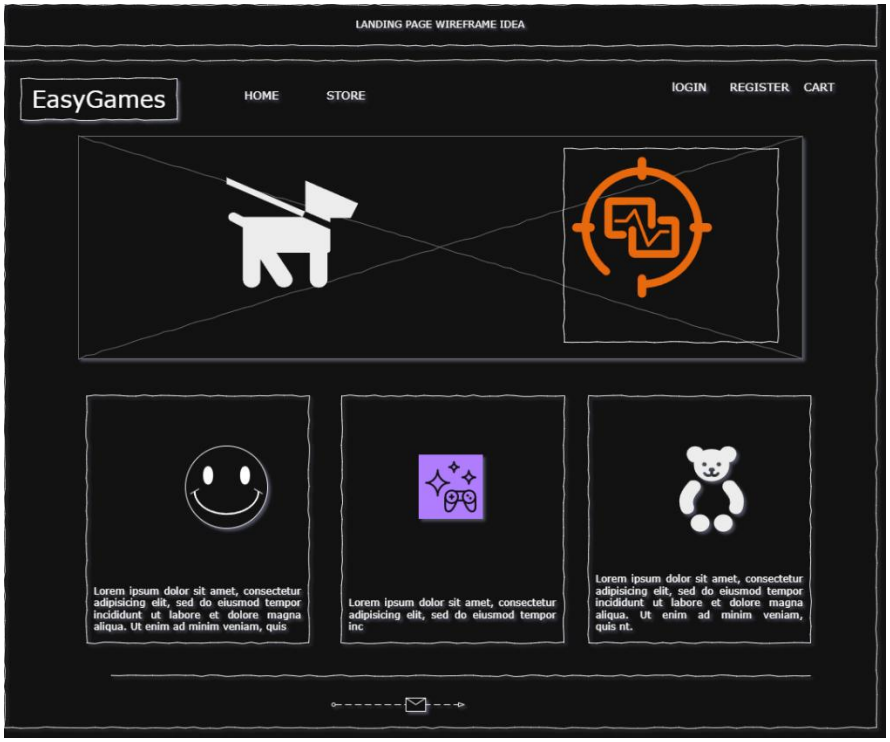
```
var ownerPrimary = Environment.GetEnvironmentVariable("EG_OWNER_EMAIL_PRIMARY");

var fromEmail    = Environment.GetEnvironmentVariable("EG_BOOTSTRAP_OWNER_FROM");

var toEmail      = Environment.GetEnvironmentVariable("EG_BOOTSTRAP_OWNER_TO");

var newPassword  =
Environment.GetEnvironmentVariable("EG_BOOTSTRAP_SET_OWNER_PASSWORD");
```

// chatGPT Prompt BOOT-002 end

Image sources. Unless otherwise noted, all product and diagram images used in the EasyGames application and in this report were created by me with AI generative tools (Google Nano/Banana, 2025) from original prompts for academic use in HIT339.

Wireframes. Website structure was planned from wireframes drawn in **app.diagrams.net**; an early landing-page concept is included in the evidence folder and guided the final Home/Store/card layout: /docs/evidence/Wireframe_Ass2.png

Figure R.2 Landing page wireframe (planning artifact)



Created by (GFDLC) in diagrams.net, 2025.

Early layout concept used to structure the Landing/Store/cards and navbar before implementation

References

APA citation style · Citing · Help & how-to · Concordia University Library. (2025, October 2). https://library.concordia.ca/help/citing/apa.php

Dotnet-Bot. (n.d.). *.NET documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/fundamentals/

Ellis, M. (2023, July 6). *How to cite ChatGPT and AI in APA Format*. How to Cite ChatGPT and AI in APA Format | Grammarly Blog. https://www.grammarly.com/blog/citations/ai-citations-apa/?utm_source=chatgpt.com

MartinDevs. (2024, August 18). *ASP.NET Core MVC eCommerce made easy: Build a Full-Stack online store | NET 8* [Video]. YouTube. https://www.youtube.com/watch?v=rTQgaQ5N9ZA

Nagel, C. (2021). *Professional C# and .NET* (8th edition). Wiley.

Turtschi, A., Werry, J., Hack, G., & Albahari, J. (2002). *C#. Net Developer's Guide* (1st ed.). Elsevier Science & Technology Books.

Wadepickett. (n.d.). *Get started with ASP.NET Core MVC*. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-9.0&preserve-view=true&tabs=visual-studio