# Final Project : “Freelance Upgrade Sprint”

## 1. Goal (What you’ll do)

Pick **one of your existing projects** from Block 1 or 2 that clearly has room to improve (the “uglier,” the better):

- a simple **HTML/CSS** page, or
- a simple **backend** (API/database) with weak/no front-end.

Pretend this project arrived at your **freelance agency**. Your “client” asked for a fast improvement pass. You will:

1. **Analyze** what’s wrong and what matters most for users.
2. **Plan** a small, realistic scope (4–6 impactful upgrades).
3. **Implement** the changes.
4. **Document** what you did in a short **journal** (process + time/cost + mini team simulation).

> **Timebox:** Aim for **8–12 hours total** work. Choose depth over breadth.
> **Deadline**: Presentation and submit - December 19th, 2025

## 2. Scope Suggestions (pick the relevant ones)

Choose **4–6** items that meaningfully improve the project.

### Front-End / UX

- Rebuild layout with semantic HTML; fix headings hierarchy (H1→H2→H3).
- Add responsive grid (CSS Grid/Flex), correct spacing/line-length, improve contrast.

- Replace generic fonts with a readable pair (e.g., system UI + display) and set proper scale (e.g., 1.25–1.333).
- Refactor color palette; add variables (`:root { --brand: … }`).
- Replace low-quality images; add alt text; optimize sizes (WebP).
- Add a simple component: sticky header, mobile nav/burger, accordion/FAQ, tabs, modal, tooltip.
- Add micro-interactions (transition/transform, reduced-motion support).
- Implement a11y quick wins: focus states, skip link, labels, keyboard navigation, ARIA where needed.
- Performance tweaks: lazy-load media, defer non-critical JS, minify assets.
- You can think of different ideas and suggest to the teacher

## Content & Structure

- Rewrite headings and CTAs for clarity; restructure long blocks into scannable sections.
- Add a site map block / breadcrumb / footer info clarity.

## JavaScript

- Validate forms (client-side); inline error messages.
- Light gallery/slider (small lib or vanilla).
- Fetch dynamic data (e.g., testimonials JSON) instead of hard-coding.

## Backend (if applicable)

- Add one **useful** endpoint (e.g., `/api/contacts`, `/api/products`).
- Connect front-end to your API; display filtered/sorted data.
- Add simple auth wall for an admin page (mock or real).

- Input sanitization; meaningful status codes; error handling JSON.

### Data & SEO

- Add basic metadata (title/description), social preview, canonical.
- Structured data (JSON-LD) for article/org/breadcrumb (simple version).

### Tooling (small wins only)

- Basic repo hygiene: README, file structure, `.gitignore`.
- Add a lightweight build step (optional): CSS minify or PostCSS autoprefixer.

> **Important:** Use techniques **covered in class** or within your level. Keep the stack simple and justified.

## 3. Deliverables (submit all on Moodle)

1. **Before/After Evidence**
    - 3–6 annotated screenshots (before vs after), or a short 30–60s screen capture.
2. **Working Files**
    - Repo link (preferred) **or** ZIP.
3. **Upgrade Journal (2–3 pages, PDF)** — see template below.
4. **(If backend)** brief API note: endpoints, sample request/response.

## Upgrade Journal — Mini Template (2–3 pages)

You must write a short **individual** journal. You don’t need to answer in full sentences for each point, but all topics must be covered.

You can use these prompts as headings or bullet points:

1. **Project & “Client” Context**
    - What project did you choose?
    - Who is the “client” in your simulation (type of business / target users)?
    - What was wrong or limited in the original version?
2. **Main Improvements**
    - What were the **3–6 main changes** you made?
    - For each change: what did you improve and why (design, UX, performance, code, accessibility, etc.)?
3. **Time & Effort**
    - Roughly how much time did you spend on this project?
    - Which tasks took longer than you expected? Why?
4. **Team Simulation**
    - If this was done in a real agency, which roles would be involved? (e.g., PM, designer, front-end dev, back-end dev, QA)
    - Which role were *you* mostly acting as in this project?
5. **Quality Check (QA)**
    - How did you test your changes? (e.g., mobile view, different browsers, keyboard navigation, broken links)
    - What still isn’t perfect or could break?
6. **Learning & Next Steps**
    - What did you learn from upgrading an “ugly” or incomplete project instead of starting from zero?
    - If you had **10 more hours**, what would you do next?

# Rubric (100 pts)

| Criterion | Excellent | Good | Developing | Pts |
|---|---|---|---|---|
| **Problem Analysis & Scope** (15) | Sharp diagnosis; picks high-impact items aligned to user/client needs. | Mostly relevant; small gaps. | Vague; low-impact picks. | /15 |
| **Quality of Improvements** (30) | Changes clearly improve UX/visuals/perf/a11y; clean, semantic code. | Visible improvements; minor issues. | Limited or cosmetic only. | /30 |
| **Technical Implementation** (25) | Solid HTML/CSS/JS (and API if used); organized structure; no obvious bugs. | Works with small flaws. | Fragile or inconsistently implemented. | /25 |
| **UX & Accessibility** (10) | Meaningful a11y wins (focus, labels, alt, contrast); better readability and flow. | Some a11y/UX gains. | Minimal or incorrect. | /10 |
| **Upgrade Journal** (15) | Concise, professional; clear rationale; time log; team simulation; QA evidence. | Complete but light on rationale/QA. | Missing pieces or unclear. | /15 |
| **Professionalism & Submission** (5) | Proper repo/ZIP, clear structure, screenshots, naming. | Minor format issues. | Disorganized / missing items. | /5 |
| **Bonus** (up to +5) | Smart extras (tiny perf boost, JSON-LD, simple test, or micro-copy improvements). | – | – | +/5 |

**Total: /100**