BlackLogic Engineering Principles

Guiding all BlackLogic Technologists

1️⃣ Software Development Lifecycle (SDLC)

No system is built without a lifecycle. Follow all stages:

Problem identification

Requirement analysis

System design

Development

Testing

Deployment

Maintenance & iteration

Coding without lifecycle thinking is guessing, not engineering.

2️⃣ Problem-Solving Mindset

Engineers exist to solve problems, not write code.

Always ask:

What problem does this solve?

Who benefits from this solution?

What happens if it fails?

A well-defined problem is already half solved.

3️⃣ Backend Engineering Principle

One backend language only: **Python**.

Consistency, security, and maintainability are mandatory.

4️⃣ Front-End Engineering Principle

Approved stack:

HTML5

CSS / Tailwind CSS

Vanilla JavaScript

Strong fundamentals outlive frameworks.

5️⃣ Layout Guide

Structure guides understanding. Content must flow logically and avoid clutter.

Layout is visual logic.

6️⃣ Navigation Discipline

Use only approved navigation types: hamburger, top, bottom, side. Maximum 6 items per navigation.

Too many options is a usability bug.

7️⃣ Logo & Identity

Identity must be visible but not dominant. Minimum size: 250px, Maximum size: 350px. Placement must be consistent.

Brand supports the system – it does not compete with it.

8️⃣ Description & Action Buttons

Every screen must clearly indicate the next action:
One primary action per screen
Clear, direct descriptions
No decorative actions
Unclear actions slow systems and users.

9️⃣ Color Palette

Color communicates system state. Use limited palette for actions, warnings, and feedback.
Color misuse is a logic error, not an aesthetic one.

1🔟 Consistency

Actions and visuals must be consistent throughout the system.

Inconsistency breaks trust faster than bugs.

1️⃣1️⃣ Forms

Forms must prevent user errors. Use minimal inputs, clear validation, and helpful error messages.

Bad input leads to bad systems.

1️⃣2️⃣ Icons & Images

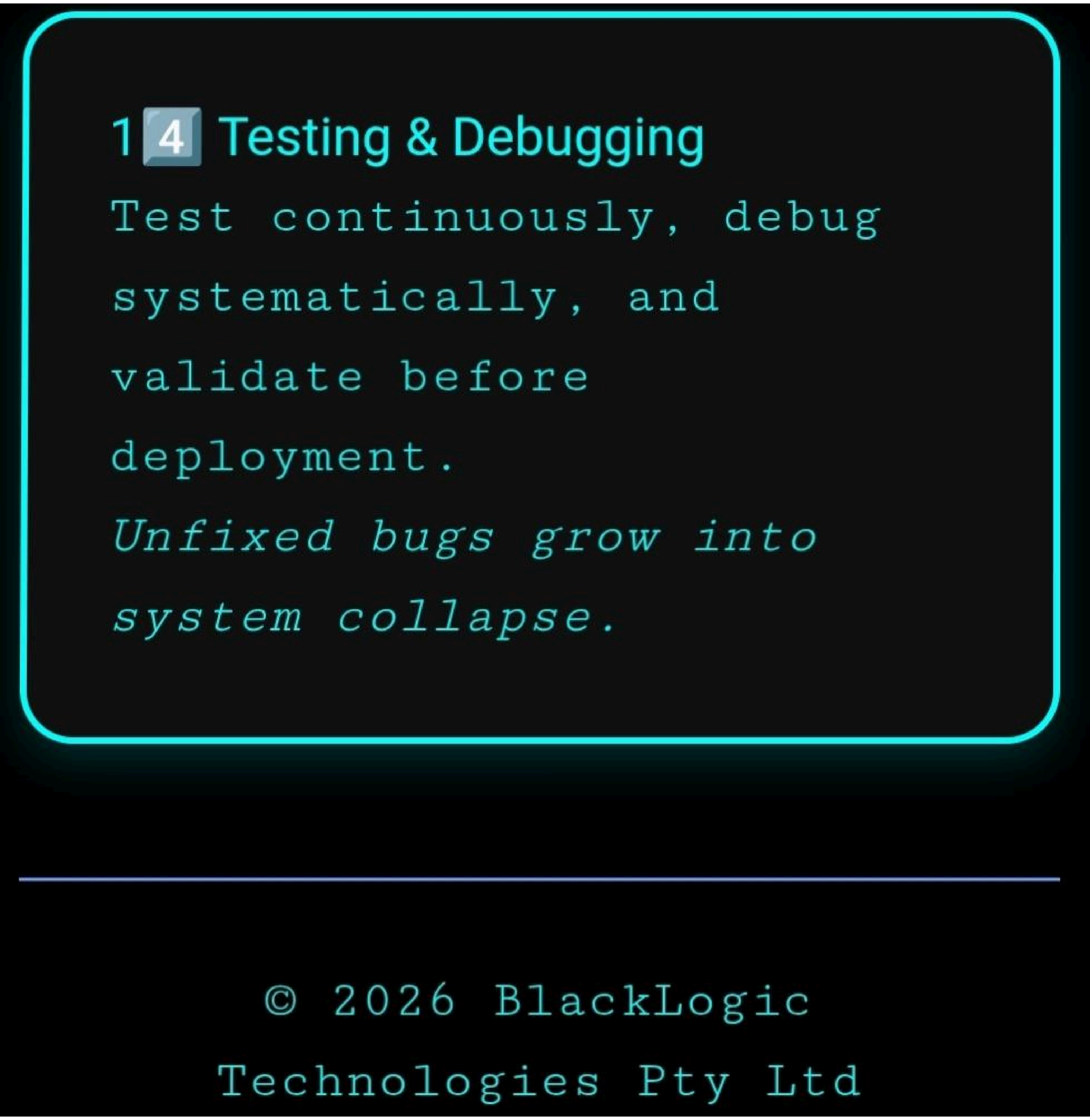
Visuals must support meaning. Icons must be intuitive. Images must add context.

If it doesn't help, remove it.

1️⃣3️⃣ UI / UX Precision

Alignment, spacing, and interactions must be precise and predictable.

Small UI mistakes signal deeper system problems.



1️⃣4️⃣ Testing & Debugging

Test continuously, debug systematically, and validate before deployment.
Unfixed bugs grow into system collapse.