PART 1

CONTENT AND ASSETS LOG –GUIDE

What the activity requires

You must:

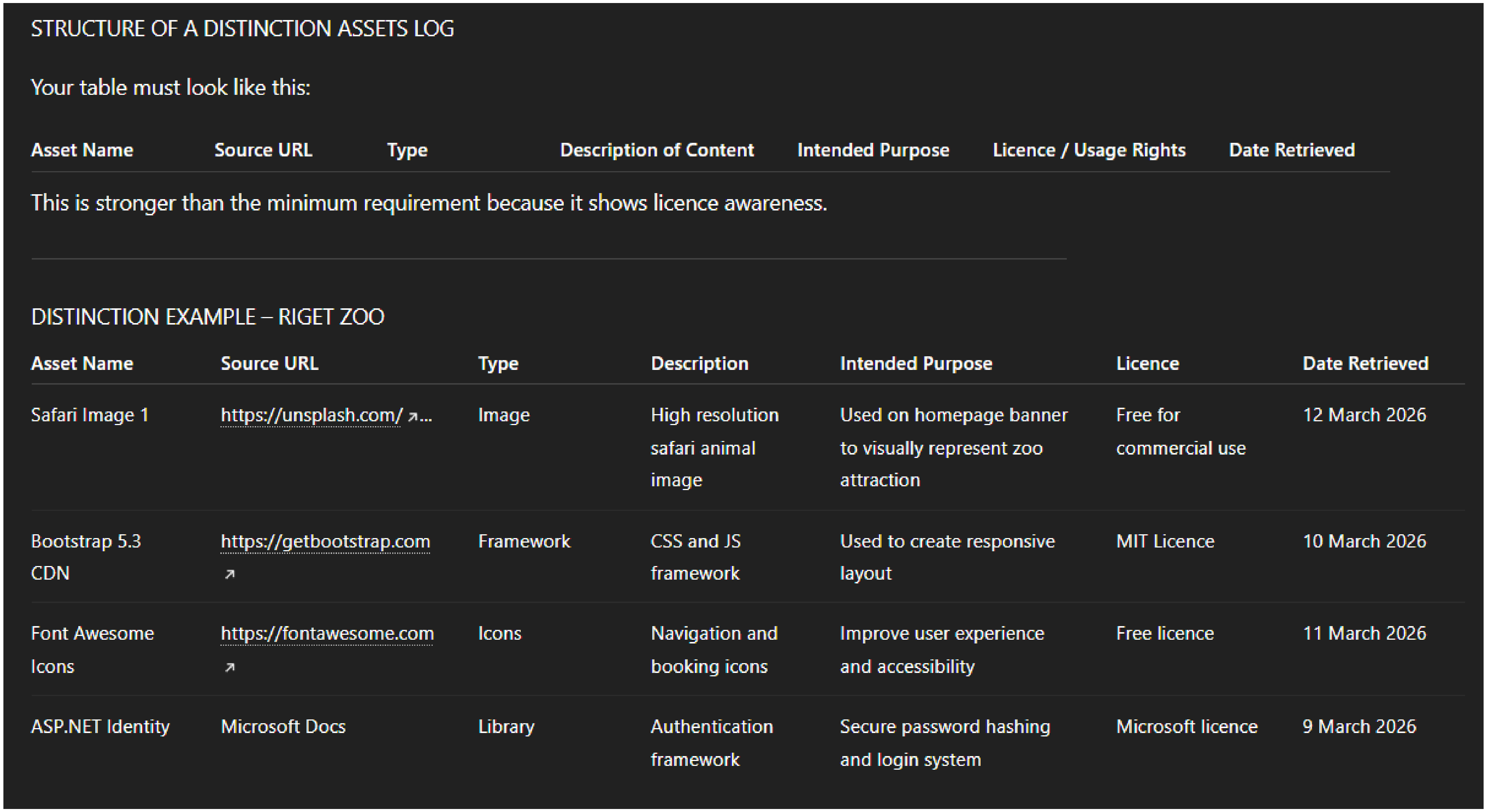
- Record ALL sources used
- Describe the content
- Explain its intended purpose
- Log the retrieval date

WHAT COUNTS AS AN ASSET

You must log:

- Images
- Icons
- Logos
- Stock photos
- External datasets
- Code snippets
- Fonts
- APIs
- Any third party template
- Any AI generated image
- Bootstrap CDN
- jQuery CDN
- Any NuGet package
- Any documentation referenced

If it did not come directly from you, log it.



STRUCTURE OF A DISTINCTION ASSETS LOG

Your table must look like this:

Asset Name	Source URL	Type	Description of Content	Intended Purpose	Licence / Usage Rights	Date Retrieved

This is stronger than the minimum requirement because it shows licence awareness.

DISTINCTION EXAMPLE – RIGET ZOO

Asset Name	Source URL	Type	Description	Intended Purpose	Licence	Date Retrieved
Safari Image 1	https://unsplash.com/ ↗...	Image	High resolution safari animal image	Used on homepage banner to visually represent zoo attraction	Free for commercial use	12 March 2026
Bootstrap 5.3 CDN	https://getbootstrap.com ↗	Framework	CSS and JS framework	Used to create responsive layout	MIT Licence	10 March 2026
Font Awesome Icons	https://fontawesome.com ↗	Icons	Navigation and booking icons	Improve user experience and accessibility	Free licence	11 March 2026
ASP.NET Identity	Microsoft Docs	Library	Authentication framework	Secure password hashing and login system	Microsoft licence	9 March 2026

LEGAL AND ETHICAL SECTION YOU SHOULD ADD

At the bottom of your assets log include:

Statement of compliance:

All third-party assets used in this prototype were reviewed to ensure compliance with copyright, licensing and GDPR requirements. Only royalty free or open-source resources were used. No copyrighted content was used without permission.

PART 2

TEST LOG – GUIDE

Examiners want evidence of:

• Normal data
• Invalid data
• Boundary data
• Iteration
• Fixes
• Regression

STRUCTURE OF A DISTINCTION TEST LOG

Use the official template but structure your entries like this:

| Description of Test | Test Data | Expected Outcome | Actual Outcome | Comments and Intended Actions |

DISTINCTION TEST LOG EXAMPLE

Description of Test	Test Data	Expected Outcome	Actual Outcome	Comments and Intended Actions
Register valid user	Valid email: test@email.comStrong password: Pass123!	Account successfully created and stored in database	Account created successfully	No changes required. Function working as intended.
Register duplicate email	Existing email already in database	Error message: “Email	Account created incorrectly	Added unique constraint in SQL database and server-side validation in AccountController.

		already exists”		Re-tested successfully. Error now displays correctly.
Register weak password	Password: 123	Error message indicating password too weak	Account created	Implemented password policy validation using ASP.NET Identity. Re-tested. Weak passwords now rejected correctly.
Book 2 tickets (Normal Test)	TicketQuantity = 2	Booking created and total cost calculated correctly	Booking created correctly	Calculation logic confirmed accurate. No changes required.
Book 0 tickets (Lower	TicketQuantity = 0	Validation error displayed	Booking created incorrectly	Added Range attribute validation [Range(1,10)] in

Boundary Test)				Booking model. Re-tested successfully.
Book 11 tickets (Upper Boundary Test)	TicketQuantity = 11	Validation error displayed	Validation error displayed	Boundary validation working correctly.
Enter text in ticket field (Invalid Data Test)	TicketQuantity = "abc"	Model validation prevents submission	Form submission blocked	Server-side validation correctly prevents invalid data type.
Overlapping hotel booking	Same RoomId and same date range	Booking prevented	Double booking occurred	Added availability check before database insert. Implemented date overlap logic. Regression tested successfully.

Loyalty points update	Book 3 tickets	Loyalty points increased correctly	Points not updated	Moved loyalty calculation into separate service class. Fixed logic. Re-tested successfully.
Regression test after validation update	Book 2 tickets after model changes	Booking still functions correctly	Booking successful	Confirmed new validation rules did not break existing functionality.