LESSON 83

Overview of ALT 1: Introduction to Interactive Information Systems

80 minutes (2 periods) | Applied Learning Tasks (ALTs) | Outcome 3.1 | Outcome 3.2 | Outcome 3.3 | Outcome 1.19 | Outcome 1.20 | Outcome 1.23 | Outcome 1.15

WHAT THIS LESSON DOES

In this lesson, you'll explore the basics of interactive information systems. Follow steps to understand the task, review key skills, brainstorm ideas, form teams, and plan your project.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.1	understand and list user needs/requirements before defining a solution
3.2	create a basic relational database to store and retrieve a variety of forms of data types
3.3	use appropriate programming languages to develop an interactive website that can display information from a database that meets a set of users' needs
1.19	identify features of both staged and iterative design and development processes
1.20	collaborate and assign roles and responsibilities within a team to tackle a computing task
1.23	reflect and communicate on the design and development process
1.15	consider the quality of the user experience when interacting with computers and list the principles of universal design, including the role of a user interface and the factors that contribute to its usability

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the fundamental concepts and purpose of interactive information systems.
- Apply prior knowledge of frontend and backend development to design a functional web-based application.
- Collaborate effectively in teams to brainstorm, plan, and execute a project.
- Identify real-world problems that can be solved through database-driven solutions.
- Develop a structured approach to project planning and task management.

BEFORE THE LESSON

- This is a planning and discussion lesson, so no environment or database needs to run today. Decide in advance how you want teams formed, whether by student choice or your own grouping, so team formation does not eat the class.
- Have a scope in mind for what an achievable artefact looks like given your class time, so you can steer brainstorming away from ideas that are too large to finish.
- Remind yourself of the CRUD requirement and any search, filter or authentication features you will accept, so you can answer feasibility questions on the spot.

Overview of ALT 1: Introduction to Interactive Information Systems

HOW THE LESSON RUNS

- Introduction
- Reviewing Key Concepts
- Understanding the Task Requirements
- Forming Your Team
- Brainstorming Artefact Ideas
- Planning Your Project
- Reflection Time

TEACHING MOVES

- 1. Introduction. Frame ALT 1 as an in-class structured task, not the externally assessed Coursework Project. Use the concrete example of a library catalogue or grade tracker so students picture a system where users add, view, update and delete data.
- 2. Reviewing Key Concepts. Draw out from students what HTML, CSS and backend logic each contribute, rather than lecturing it. Ask who feels confident with forms, who with styling, who with data handling. Their answers seed the team roles two steps later.
- 3. Understanding the Task Requirements. Spell out that the artefact must have a database plus a user interface covering all four CRUD operations. Name search, filtering and authentication as extensions, not the core, so weaker teams do not over-scope.
- 4. Forming Your Team. Hold teams to two to four students and insist each person names a distinct role: database, frontend, backend, testing. Pair complementary strengths deliberately rather than letting friend groups cluster all one skill together.
- 5. Brainstorming Artefact Ideas. Push teams to start from a real problem in their school or community, not from a technology they want to use. Timebox to twenty minutes and ask each team to defend why their idea genuinely needs a database.
- 6. Planning Your Project. Require a written project goal and a task breakdown before the class ends. Ask each team to say who owns which piece, so the plan divides work rather than listing features nobody has claimed.
- 7. Reflection Time. Keep this spoken. Ask each student for one challenge they foresee and one prior skill that will help. This surfaces gaps you can plan around before coding starts.

WHAT TO WATCH FOR

- Students think any website with a form counts as a database-driven interactive information system, when a static page that stores nothing does not meet the task. Ask each team to point to where their data lives and how it survives after the browser closes. If they cannot, the design has no database yet.
- Students confuse this ALT with the Coursework Project and assume it is graded and individual. State plainly that ALTs are team-based, formative and not externally assessed. The individual, externally graded project is a separate piece of work later in the course.
- Teams scope an idea far too large to finish in the available time, such as a full social network. Have them list only the CRUD operations on a single type of record. If the plan needs more than one core data table to work at all, cut it back.

DIFFERENTIATION

- Emerging: Give a student who cannot pick an idea a shortlist from the lesson: library book review, grade tracker, recipe store. Let them adapt one rather than invent from nothing. Assign a clear, bounded role such as frontend layout so the student contributes without needing to hold the whole system in their head.
- Developing: Have the team walk you through their plan and explain how each CRUD operation will actually work before they move on, so they reason about the design and not just list features.
- Proficient: Challenge a finished team to add a Higher Level feature such as user authentication or filtering, and to justify why it improves the system. Ask them to sketch the database structure they will need, naming the fields, as early evidence they can carry into the Applied Learning Task write-up.