LESSON 85

Building the Database and Backend Integration

240 minutes (6 periods) | Applied Learning Tasks (ALTs) | Outcome 3.1 | Outcome 3.2 | Outcome 3.3 | Outcome 2.16 | Outcome 2.18 | Outcome 1.22 | Outcome 2.20

WHAT THIS LESSON DOES

In this lesson, you'll create a database for your interactive information system and integrate it with a backend. Follow steps to review data types, set up your database, insert data, build backend routes, and test everything effectively.

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
2.16	use data types that are common to procedural high-level languages
2.18	collect, store and sort both continuous and discrete data
1.22	read, write, test, and modify computer programs
2.20	identify and fix/debug warnings and errors in computer code and modify as required

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the principles of data types and relational database structures for effective data organisation.
- Design and create a functional database schema tailored to user needs.
- Implement data insertion and querying techniques to manage database content.
- Develop backend routes to facilitate interaction between the database and application.
- Apply abstraction to organise data efficiently while meeting system requirements.

BEFORE THE LESSON

- Confirm each machine can create a Python virtual environment and install Flask behind the school filter. Test the exact pip install command yourself first, as a blocked install will stall the whole class.
- Have a working create_db.py, insert_test_data.py and a minimal Flask route saved somewhere the class can reach, so students who fall behind can catch up from a known-good copy.
- Check VS Code and Python are installed and that students know how to open a terminal in their project folder.

Building the Database and Backend Integration

HOW THE LESSON RUNS

- Introduction
- Review of Data Types and Relational Databases
- Creating the Database
- Inserting Data
- Querying the Data
- Creating Backend Routes
- Wrap Up

TEACHING MOVES

- 2. Review of Data Types and Relational Databases. Tie each data type to a real field in their own schema: an ID as integer, a price as real, availability as boolean. Ask why storing a price as text would cause problems later.
- 3. Creating the Database. Insist the virtual environment is active before installing Flask. Have students run create_db.py once and check a .db file appears in the folder. No file means the connect path is wrong.
- 4. Inserting Data. Warn that running the insert script twice will error on duplicate primary keys. Explain this is expected, not a bug, and shows the key is doing its job.
- 5. Querying the Data. Make them run a SELECT and eyeball the output against what they inserted. A looks-fine table is not verified until the returned rows match. Only reach for ALTER TABLE when the structure is genuinely wrong.
- 6. Creating Backend Routes. Start with one Read route tested in the browser before adding Create, Update or Delete. Get one route returning real data first, then build outward from a working point.
- 7. Wrap Up. Have each student demonstrate one working route to you before they finish. Point them to name exactly what the frontend teammate will need: route URLs and the data each returns.

WHAT TO WATCH FOR

- Students think a page or query that runs without an error message is therefore correct. Insist they compare returned rows against the data they inserted by hand. No error is not the same as right data.
- Students confuse what runs in the browser with what runs in the Flask backend, expecting the database to be reachable from browser code. Draw the split: the browser requests a route, Flask talks to the database, Flask sends data back. The database never talks to the browser directly.
- Re-running the insert script and reading the duplicate-key error as broken code. Explain the primary key is rejecting a repeat. Show how to clear or guard the table instead of blaming the script.

DIFFERENTIATION

- Emerging: If the environment or install is fighting them, give the known-good starter files so they build database understanding rather than losing the session to setup. Have them describe in words what one table holds before writing any CREATE TABLE statement.
- Developing: Ask them to explain why they chose each data type in their schema before moving on to routes. Pair them to test each other's SELECT output against expected data.
- Proficient: Stretch toward the full CRUD set with an Update and Delete route, and have them handle a query that returns no rows gracefully. Ask them to write a short note documenting their routes as evidence for the group Applied Learning Task.