

SULTAN QABOOS UNIVERSITY, COLLEGE OF SCIENCE  
DEPARTMENT OF COMPUTER SCIENCE, FALL 2025  
COMP4701: WEB APPLICATION DEVELOPMENT  
PROJECT – PHASE 2 – DATA ACCESS – 100 POINTS

**DUE DATE: MONDAY 1<sup>ST</sup> DECEMBER 2025 @ 11:59 PM**

- **NOTE:** You are not allowed to use the MVC Framework in this phase as MVC will be covered in phase 3.

**QUESTION 1 [20 POINTS]: DATABASE APPLICATION**

Create an SQL Server database with tables and constraints that reflect the requirements of your project. This phase replaces the C# classes used in Part 1 with database tables.

- 1) [6 Points] Create at least three database tables corresponding to the three C# classes used in Part 1, ensuring appropriate attribute names and data types.
- 2) [2 Points] Add an additional table to store user login information (e.g., username, password, email, phone number, etc.).
- 3) [4 Points] Define primary key constraints for all tables and foreign key constraints where applicable.
- 4) [3 Points] Apply other necessary constraints such as CHECK, DEFAULT, and NOT NULL when appropriate.
- 5) [3 Points] Create at least one database view that retrieves and processes data from multiple tables, and integrate its logic into your web application.
- 6) [2 Points] Populate all tables with complete and realistic data that supports the full functionality and transactions of your web application.

**QUESTION 2 [30 POINTS]: REDESIGN YOUR APPLICATION USING DIRECT DATA ACCESS**

- 1) [5 Points] Remove all classes, objects, and collections from Phase 1 and replace them with direct database access. At least three web pages must be redesigned using this approach (others may be handled under Question 3).
- 2) [6 Points] Implement at least three SELECT statements, each used to retrieve data directly from the database in three different web pages.
- 3) [8 Points] Use direct SQL commands to perform INSERT, UPDATE, and DELETE operations with proper logic.
- 4) [6 Points] Use appropriate ASP.NET Core components to render the retrieved data, and apply parameterized SQL commands with exception handling.
- 5) [5 Points] Ensure all HTML forms include proper exception handling and input validation controls.

**QUESTION 3 [40 POINTS]: ENTITY FRAMEWORK (For web pages that are not used in Question 2 above)**

- 1) [6 Points] Build an Entity Framework model to access all database tables.
- 2) **Adding Authentication pages with data sharing**
  - a) [6 Points] Develop three pages for login, logout, and user registration, each using Entity Framework to validate login credentials and register new users. User login info must be displayed on all pages.
  - b) [4 Points] Implement suitable state management techniques to share user information across pages.
  - c) [4 Points] Restrict access so that no page in the application can be accessed without logging in.
- 3) Use LINQ statements to access the database in at least 5 web pages (including login, logout and registration pages)
  - a) [5 Points] Use at least three different LINQ query styles with suitable ASP.NET Core elements and formatting.
  - b) e. [5 Points] Use at least three different types of lambda expression queries.
  - c) f. [6 Points] Use all major LINQ constructs, including where, order by, group by, and queries involving multiple database tables.
  - d) g. [4 Points] Perform full CRUD operations using LINQ, including adding, deleting, and updating data in the database.

**QUESTION 4 [10 POINTS] TEAMWORK AND SUBMISSION**

- 1) **Team Contribution [2 Points]:** The project tasks should be distributed evenly among team members. You need to specify the contribution for each team member in a tabular format.
- 2) **Video Demo [5 Points]:** Use **video demo** to demonstrating the main functionalities in your application, and how you have address all project requirements as stated above. Team members need to participate in the presentation evenly and will be evaluated accordingly.
- 3) **GitHub [3 Points]:** You need to upload your code and Demo in both Moodle and **GitHub** accounts. In addition, you need to use GitHub tools such as version control to document and improve your team collaboration and each team member needs to commit his/her contribution and changes.