ROSSMOYNE SENIOR HIGH SCHOOL

Computer Science ATAR

Year 12 ATAR: Unit 4 – Managing Data – Project

Develop and implement an integrated database programming solution using a software development framework. Include ER diagrams, a normalised schema, SQL queries (CREATE, INSERT, SELECT, UPDATE, DELETE), and appropriate security measures.

Weighting: 20%
Due date: 6th September, 2025

Project Title: "Community Connect" - A Volunteer Coordination Platform

This project requires students to design, develop, and evaluate a database-driven web application to connect local volunteers with community organisations needing assistance for various events.

Project Scenario & AI Usage Policy

Local community groups and non-profits often struggle to find and manage volunteers for their events (e.g., fundraisers, clean-up days, festivals). Volunteers also find it difficult to discover opportunities that match their skills and availability. Your task is to develop a prototype system called **"Community Connect."** This system will consist of a robust relational database and a web interface. It will allow organisations to post volunteering opportunities and allow volunteers to sign up, list their skills, and register for events.

The final product will be a **Python Flask web application** that interacts with a **SQLite database**, demonstrating a comprehensive understanding of data management, modelling, integrity, and development issues.

See next page

Note on AI Usage: You are permitted and encouraged to use AI development tools (like Gemini, GitHub Copilot, etc.) to generate the front-end **User Interface (UI)** code (i.e., the HTML, CSS, and basic Flask routing templates). The primary assessment focus is on the **database design, SQL implementation, and back-end logic**. The UI is simply a means to interact with your database. **All database design (ERD, normalisation), SQL queries, and Python logic for handling data must be your own original work.**

Weekly Breakdown

Week 1: Investigation and Database Design

This week focuses on understanding the problem, planning the database structure, and ensuring the design is robust and efficient.

- **Tasks:**
    1. **Investigation:** Research existing volunteer platforms. Find 1-2 articles (newspaper, journal, or online) discussing the challenges of volunteer management or the importance of community engagement. Summarise the key takeaways.
    2. **Deconstruction:** Identify the core entities, their attributes, and the relationships between them (e.g., Volunteers, Organisations, Events, Skills).
    3. **Entity Relationship Diagram (ERD):** Create a detailed ERD using **Crow's Foot notation** for the "Community Connect" system. It must contain a minimum of **five tables** and correctly resolve at least one **many-to-many (**M:N**) relationship** (e.g., a volunteer can sign up for many events, and an event can have many volunteers).
    4. **Normalisation:** Start with a single, unnormalised table or a poorly structured set of tables representing the data. Document the process of normalising the data to the **Third Normal Form (3NF)**, showing the relational notation for 1NF, 2NF, and 3NF.
    5. **Data Dictionary:** Create a comprehensive data dictionary for your normalised (3NF) database design.

See next page

6. **Ethical & Legal Research:** Begin researching the relevant Australian Privacy Principles (APPs) and other ethical/security concerns related to storing personal data (names, contact info, skills).

- **Week 1 Deliverables:**
  - A project management plan outlining timelines.
  - A research summary with referenced articles.
  - A complete ERD in Crow's Foot notation.
  - Full documentation of the normalisation process to 3NF.
  - A complete data dictionary.

Week 2: Database Implementation and Basic Application

This week focuses on translating the design into a functional database and building the core connectivity.

- **Tasks:**
  1. **SQL Database Creation:** Using DB Browser for SQLite, write and execute **SQL CREATE TABLE** statements to build the database from your 3NF design.
     - Enforce **data integrity** using constraints (PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK).
     - Implement **referential integrity** with FOREIGN KEY constraints, including the use of ON DELETE CASCADE where appropriate.
  2. **Data Population:** Write **SQL INSERT** statements to populate your database with at least 15-20 records of realistic sample data distributed across the tables.
  3. **Python Flask Setup:** Set up a basic Flask web application that can connect to your SQLite database (**Open Database Connectivity**).
  4. **Basic CRUD Functionality:** Implement simple web pages/forms to perform the following:
     - **Create:** A form to add a new volunteer to the database (INSERT).
     - **Read:** A page that displays a list of all organisations from the database (SELECT).
     - **Update:** A form to update a volunteer's contact number (UPDATE).

See next page

        - **Delete:** A button to remove a specific event (DELETE).
    5. **ACID Properties:** In your development journal, briefly explain how a simple transaction, like a volunteer signing up for an event, relates to the principles of **ACID** (Atomicity, Consistency, Isolation, Durability).
- **Week 2 Deliverables:**
    - A .sql file containing all CREATE TABLE and INSERT statements.
    - The SQLite database file (.db).
    - The Python Flask project files.
    - Screen captures of the basic web application performing the CRUD operations.
- **End of Week 2 In-Class Investigation (Device-Free):**
    - **Task:** Students are given a paper copy of their own ERD and data dictionary from Week 1 and must complete SQL Query questions for set tasks. You must use Aliases where appropriate.

Week 3: Advanced Queries, Evaluation, and Reporting

This week focuses on implementing complex data retrieval, evaluating the solution, and documenting the entire process.

- **Tasks:**
    1. **Advanced SQL Integration:** Enhance the Flask application by implementing pages that display the results of complex queries, such as:
        - A page to search for volunteers based on a specific skill (e.g., 'First Aid Certified').
        - A page for an organisation to view all volunteers signed up for one of their events (requiring an **inner join across 3 or more tables**).
        - A summary page showing statistics using **aggregate functions** (e.g., COUNT of volunteers per event, AVG duration of events). Use **GROUP BY**.
        - Display a formatted list of volunteers, showing their full name as a single field (**concatenated field**) and their age calculated from their date of birth (**calculated field**). Use **aliases** for these new fields.

See next page

2. **Data Quality & Cleaning:** Write a brief paragraph discussing the factors influencing the quality of the data in your system (currency, authenticity, relevance, accuracy) and how you might clean for outliers (e.g., an invalid date of birth).
3. **Final Report:** Compile a comprehensive final report that includes:
    - Introduction and problem definition.
    - All design documents from Week 1 (ERD, normalisation, data dictionary).
    - All SQL scripts from Week 2.
    - Screen captures of the final application, demonstrating all basic and advanced features.
    - **Evaluation:** A critical evaluation of the final product against the initial requirements. Discuss limitations and potential future improvements.
    - **Development Issues:** A detailed discussion on the **ethical, legal, and security issues** pertaining to this project.
        - **Ethical:** Privacy concerns, appropriate use of data, data mining potential.
        - **Legal:** Specifically address how your design complies with **APP5 (Notification of Collection), APP10 (Quality of Personal Information), APP11 (Security), and APP12 (Access to Personal Information)**.
        - **Security:** Discuss measures like restricting access, the importance of backups, and data ownership.

- **Week 3 Deliverables:**
    - The final, fully functional Python Flask application and SQLite database.
    - A comprehensive final report in digital format.
- **End of Week 3 In-Class Investigation (Device-Free):**
    - **Task:** Students are presented with a short case study on paper.
    - **Scenario:** "The 'Community Connect' platform suffers a data breach, and a news outlet reports that volunteer data was sold to marketing companies. A volunteer who signed up last year formally requests that all their personal data be permanently removed from your system."

See next page

Marking Key

Criteria	A Grade (Excellent)	B Grade (Good)	C Grade (Satisfactory)	D Grade (Limited)
ERD & Normalisation (30% weighting on total)	Flawlessly models requirements in a 5+ table ERD with correct Crow's Foot notation. Justifies design choices. Consistently and accurately normalises a complex model to 3NF.	Accurately models most requirements in an ERD. Applies normalisation to 3NF correctly.	Creates a functional ERD, but with some inconsistencies in notation or relationships. Applies normalisation to 3NF for a simpler model.	Creates an incomplete ERD with inaccurate keys/relationships. Attempts normalisation but does not achieve 3NF correctly.
SQL: Database Creation (15% weighting on total)	Consistently writes SQL to create a functional multi-table database that accurately reflects the 3NF design, using a wide range of constraints to ensure high data validity and integrity.	Uses SQL to create a functional multi-table database reflecting requirements, with appropriate keys and data types.	Uses SQL to create a simple database, but with some errors or missing constraints.	Creates a database with limited functionality that only partially reflects the design.
SQL: Advanced Queries (30% weighting on total)	Consistently and accurately uses complex SQL (multi-table joins, aggregates, GROUP BY, calculated/concatenated fields with aliases) to produce meaningful and efficient outputs integrated into the application.	Writes functional queries across multiple tables using aggregates and calculated/concatenated fields to extract meaningful data.	Writes simple queries across multiple tables, with inconsistent use of aggregates or calculated/concatenated fields.	Unsuccessfully creates queries across multiple tables.
Application Integration (5% weighting on total)	Consistently and accurately uses Python/Flask to develop an efficient and effective solution that fully meets all specified CRUD and advanced query requirements, demonstrating excellent DB connectivity.	Uses Python/Flask to develop a working solution that meets most requirements, including DB connectivity for CRUD and some advanced queries.	Develops a solution that partially meets requirements. Basic CRUD may work, but advanced queries are limited or buggy.	Develops an incomplete or non-functional solution with limited database interaction.
Development Issues Report (Ethical, Legal, Security) (10% weighting on total)	Provides a justified and insightful analysis of ethical, legal, and security issues. Accurately and specifically relates the project to the named Australian Privacy Principles with justification.	Explains ethical, legal, and security issues. Relates the project to the relevant APPs.	Describes ethical and legal issues related to the project but with limited depth or direct application.	Demonstrates a limited understanding of ethical and legal issues with minimal relevance to the project.
Development Process & Documentation (10% weighting on total)	Consistently applies and justifies a structured development process. All documentation (data dictionary, research, screen captures) is accurate, comprehensive, and professional.	Consistently applies a structured process. All required documentation is present and accurate.	Applies a development process. Documentation is mostly complete but may have minor errors or omissions.	Inconsistently applies a development process. Documentation is incomplete or inaccurate.

See next page