INF1002 Programming Fundamentals

Team Project Specification

1. Introduction

A database is a structured collection of data organized in a way that allows for efficient storage, retrieval, and manipulation of that data. A database normally has many data tables and each data table includes data for a group of similar records. The table is made up of columns and rows and usually the first column is used as the unique "ID" to store each record (row). For example, the data table below is used to store the prices and vendor for each fruit ("ID").

Data Table: FruitPrice		
ID	***Price***	***Origin***
Apple	3.7	USA
Orange	3.6	China
Watermelon	5.6	Malaysia

A Database Management System (DBMS) is software that provides an interface for users' interacting with databases. It is a crucial component in modern information systems and plays a central role in storing, retrieving, and managing data. DBMS provides several fundamental operations for interacting with data stored in a database. These operations typically include: Insert (Add), Query, Update (Edit), and Delete.

2. Detailed Requirements

Your team is required to implement Class Management System (CMS) programme, a simple database management system in C with the requirements described as below. The programme will be using a command-line-like interface; **no graphical UI is required.**

2. 1. File Database

A file will be used as the database to store all the data records. Only **ONE** data table "**StudentRecords**" needs be implemented in your file database and there are 4 columns in the table as shown in the table below. A sample database file can be found in the "Sample-CMS.txt" file.

StudentRecords			
ID (data type: integer)	***Name (data type: string)***	***Programme (data type: string)***	***Mark (data type: float)***
2301234	Joshua Chen	Software Engineering	70.5
2201234	Isaac Teo	Computer Science	63.4
2304567	John Levoy	Digital Supply Chain	85.9

Please name your database file in the following convention:

- TeamName-CMS.txt. For example: P1_1-CMS.txt

2.2. Operations

Your programme should allow the users to conduct the following operations on the data records read from the file database.

- OPEN: to open the database file and read in all the records
- SHOW ALL: to show all the current records.
- INSERT: to add a new record to the database.
- QUERY: to search for a record in the database.
- UPDATE: to update the value for a given record.
- DELETE: to delete a record from the database.
- SAVE: to save all the records to the database file

Operation	Description
OPEN	Open the database file and read in all the records.
SHOW ALL	To display all the current records in the read-in data.
INSERT	To insert a new data record. • If there is already a record with the same student ID, an error message will be displayed and the insertion will be cancelled. • If there is no record with the same student ID, then ask the user to enter the data for each "Column" of the new record.
QUERY	To search if there is any existing record with a given student ID. • If a record with the same student ID exists, then display the data for the record. • If there is no existing record with the same student ID, display a warning message to indicate there is no record found.
UPDATE	To update the data for a record with a given student ID. • If there is no existing record with the same student ID, display a warning message to indicate there is no record found. • If a record with the same student ID exists, then ask for the new data for each "Column" of the record and update the record.
DELETE	To delete the record with a given student ID. • If is no existing record with the same student ID, display a warning message to indicate there is no record found. • If a record with the same student ID exists, double confirm with the user for deletion. If confirmed, then delete the record. Otherwise, no deletion will be made.
SAVE	Save all the current records into the database file.

Your programme should accept the operation commands from you (the "user") and make proper responses to your command. Please refer to Appendix A for the sample responses.

2.3. Enhancement Features

• Sorting Features

- Implement sorting of student records by:
  - Student ID (ascending or descending), and
  - Mark (ascending or descending).
- The sorted output should display correctly when users execute commands such as `SHOW ALL SORT BY ID` or `SHOW ALL SORT BY MARK`.

• Summary Statistics

- Implement summary commands, for example:
  - `SHOW SUMMARY` → Displays:

- Total number of students
- Average mark
- Highest and lowest mark (with student names)

- **Unique Feature**

- A unique feature derived by your team which demonstrates a number of the features of C and fits into your CMS system naturally.

2.4. Test Case Design and Validation

- Your team is required to come up with a comprehensive list of test cases to ensure your application works without errors, the following is a sample format for reference:

Test Case ID	Description	Input Commands(s)	Expected Output	Reason for Test	Actual Result
TC05	Update Record Successfully	UPDATE ID=250001 Mark=88.0	CSM: The record with ID=250001 is successfully updated	Verifies Update Logic	Pass

- When designing your test cases, you should identify any area in which your application may fail and test to ensure this case has been accounted for
- Test cases should be practically used while creating your application, but should also be listed in the report, and executed in your video presentation.

2.5. Declaration of non-plagiarism

It is required that the CMS programme will output the following content when it is started:

“

Declaration

SIT’s policy on copying does not allow the students to copy source code as well as assessment solutions from another person AI or other places. It is the students’ responsibility to guarantee that their assessment solutions are their own work. Meanwhile, the students must also ensure that their work is not accessible by others. Where such plagiarism is detected, both of the assessments involved will receive ZERO mark.

We hereby declare that:

- We fully understand and agree to the abovementioned plagiarism policy.
- We did not copy any code from others or from other places.
- We did not share our codes with others or upload to any other places for public access and will not do that in the future.
- We agree that our project will receive Zero mark if there is any plagiarism detected.
- We agree that we will not disclose any information or material of the group project to others or upload to any other places for public access.

- We agree that we did not copy any code directly from AI generated sources

Declared by: Group Name (please insert your group name)
Team members:

1. XXX
2. XXX
3. XXX
4. XXX
5. XXX

Date: (please insert the date when you submit your group project).

“

3. Report

In addition to your code, you must also submit a **brief** technical report that describes how the program works and how you went about constructing it. It should contain:

- A technical description of how the programme is structured.
- Technical descriptions explaining what data structures (if any) were used and why these were chosen.
- A breakdown of the contribution made by each team member to the project.
- A list of all test case and validation, a sample one is provided in 2.4 above.

There should be of sufficient detail in your report that someone else who has also studied INF1002 would be able to implement a similar program without needing to refer to your code.

The report **does not** need to contain a user guide for the program.

4. Timeline and Deliverables

By 23:59 on 25th November 2025, Tuesday	Final Submission	Submit a zip file to your group’s xSiTe Dropbox, including: • All C source code • The compiled and runnable .exe file • Report / Reflection • Presentation slides The zip file should be named in format of: “INF1002C-TeamX”.zip. For example: “INF1002C-P1_1.zip”. For the video presentation, one video uploaded to YouTube or similar platform (can be marked as unlinked to keep it private to this module) and attach the link in your report and your zip file (URL link). Please ensure your instructor has access before submitting the link.

By 23:59 26th November 2025, Wednesday	Peer Evaluation	Complete your peer evaluation on TEAMMATES. Otherwise, • There will be a **10% penalty** for your own peer evaluation mark. • The contribution credits will be evenly distributed among the other team members.

5. Late Submission

A penalty of 20% per day for each deliverable will be imposed for late submission unless extension has been granted prior to the submission date. Request for extension will be granted on a case-by-case basis. Any work submitted more than 4 days after the submission date will not be accepted and no mark will be awarded.

6. Assessment criteria

Your assignment will be assessed according to the criteria listed in the mark scheme in Table 1.

Table 1 Assessment Criteria

Each requirement/function will be evaluated based on the following criteria.

Code (80%)
• Code completeness and correctness (45%)
  ○ All requirements are implemented.
  ○ Produce the correct results and exhibit the expected behaviours.
  ○ Data structure is properly chosen and is used efficiently.
• Code clarity (5%)
  ○ Code is well structured and organized with good modularity.
  ○ Code is easy to read with clear and sufficient comments.
  ○ Variables and functions are properly named with meaningful names.
• Code reliability (10%)
  ○ No bugs or errors.
  ○ Sufficient data validation with error/boundary cases handled properly.
  ○ Proper error messages are displayed for users.
• Enhancement Features (10%)
  ○ To achieve the highest grade, all enhancement features must be implemented successfully
• Test Cases and Validation (10%)
  ○ Design and document comprehensive test cases that verify each function and edge condition of the CMS.
  ○ Clearly demonstrate coverage, correctness, and the ability to detect potential issues.
  ○ Well-documented test case tables with expected and actual results are required in the report.

Report (10%) • A logic flow with all required information (Section 3) included. • The code structure and the used data structure (if any) are clearly elaborated. • Well organized and easy to read with clear and concise writing style. • Free from grammatical errors and typos • A list of all test cases designed by the team for testing and validation on their application • No more than 15 pages, Times new Roman, font size 11 or greater (standard margines and borders). This does not include title page, table of content, appendix. For the appendix, you

can include items for reference, but your report should be standalone without the appendix (if the examiner MUST look at something in the appendix, it will be counted towards the page count).

Individual Reflection (5%)

- A clear and honest reflection of your own contributions
- A critical evaluation of the project, what went right and what went wrong
- An honest reflection of how AI and AI-Teammate was used in this project

Video Project presentation and demonstration (5%)

- Presentation (max 5 minutes)
  - Functions/features, code structure and data structures are clearly presented.
  - Presentation time is properly managed.
- Video demo of the developed CMS (max 10 minutes)
  - All required functions are demonstrated successfully.
  - Properly handle comprehensive test-cases developed by your team
  - Demo time is properly managed.

The grade for each individual member will be weighted based on the contribution and peer evaluation.

Appendix A. Sample responses

Assume you give a name "CMS" for your programme and the user is named as "P1_1". The database file is named "P1_1-CMS.txt".
Below shows sample responses.

OPEN	P1_1: OPEN CMS: The database file "P1_1-CMS.txt" is successfully opened.
SHOW ALL	P1_1: SHOW ALL CMS: Here are all the records found in the table "StudentRecords". ID Name Programme Mark 2301234 Joshua Chen Software Engineering 70.5 2201234 Isaac Teo Computer Science 63.4 2304567 John Levoy Digital Supply Chain 85.9
INSERT	P1_1: INSERT ID=2301234 CMS: The record with ID=2301234 already exists. P1_1: INSERT ID=2401234 Name= Michelle Lee Programme=Information Security Mark=73.2 CMS: A new record with ID=2401234 is successfully inserted. P1_1: SHOW ALL CMS: Here are all the records found in the table "StudentRecords". ID Name Programme Mark 2301234 Joshua Chen Software Engineering 70.5 2201234 Isaac Teo Computer Science 63.4 2304567 John Levoy Digital Supply Chain 85.9 2401234 Michelle Lee Information Security 73.2
QUERY	P1_1: QUERY ID=2501234 CMS: The record with ID=2501234 does not exist. P1_1: QUERY ID=2401234 CMS: The record with ID=2401234 is found in the data table. ID Name Programme Mark 2401234 Michelle Lee Information Security 73.2
UPDATE	P1_1: UPDATE ID=2501234 Mark=69.8 CMS: The record with ID=2501234 does not exist. P1_1: UPDATE ID=2401234 Mark=69.8 CMS: The record with ID=2401234 is successfully updated. P1_1: UPDATE ID=2401234 Programme=Applied AI CMS: The record with ID=2401234 is successfully updated. P1_1: SHOW ALL CMS: Here are all the records found in the table "StudentRecords". ID Name Programme Mark 2301234 Joshua Chen Software Engineering 70.5 2201234 Isaac Teo Computer Science 63.4 2304567 John Levoy Digital Supply Chain 85.9 2401234 Michelle Lee Applied AI 69.8

<table>
<tr><td>DELETE</td><td>P1_1: DELETE ID=2501234
CMS: The record with ID=2501234 does not exist.
P1_1: DELETE ID=2401234
CMS: Are you sure you want to delete record with ID=2401234? Type “Y” to Confirm or type “N” to cancel.
P1_1: N
CMS: The deletion is cancelled.
P1_1: DELETE ID=2301234
CMS: Are you sure you want to delete record with ID=2301234? Type “Y” to Confirm or type “N” to cancel.
P1_1: Y
CMS: The record with ID=2301234 is successfully deleted.
P1_1: SHOW ALL
CMS: Sure. Here are all the records found in the table “StudentRecords”.
ID Name Programme Mark
2201234 Isaac Teo Computer Science 63.4
2304567 John Levoy Digital Supply Chain 85.9
2401234 Michelle Lee Applied AI 69.8</td></tr>
<tr><td>SAVE</td><td>P1_1: SAVE
CMS: The database file “P1_1-CMS.txt” is successfully saved.</td></tr>
</table>