ENSF 380 WS 2025 Individual Project Instructions

Features (New and changed requirements)

Overview

Our initial concept was based on a preliminary understanding of the project requirements. Through consultation with potential users, we identified a number of shortcomings of the original design. Some of these are technical, while others demonstrate the need for engagement with diverse potential users in order to develop software which addresses a variety of needs.

Refer to the rubric to see the relative weighting of each feature. The rubric is designed to prioritize complete and correct implementation of a feature over partial implementation of multiple features. The more difficult the feature, the more weight it carries. However, the user interface is a special case as all other features depend upon it. **A user interface must be implemented in order for us to run and grade your code.** If we cannot see evidence of a feature by running the program, we will assume the feature has not been implemented.

We are interested in the overall functioning of the program as described in GP1 and the features described here, not the specific implementation details which were provided as example solutions to the group project. Example solutions were limited by the concepts we had studied at that time. For example, if you decide that the ASSIGNED_SOCIAL_ID field in DisasterVictim is essentially identical to an auto-incrementing database key and therefore the counter field is no longer needed, you are free to remove it. Likewise, if you determine that medical records are better stored as a List than an Array, you are free to make that change. Renaming classes, methods and parameters is completely acceptable.

Features

1) User-friendly interface

A typical user does not expect a program to throw an exception. They would like for the program to be tolerant of typos, and to require minimal typing.

Task: Create a user-friendly interface to the program.

- You may opt to develop a GUI or command-line (CLI) user interface. No extra marks are given for choosing a GUI interface; we would like you to prioritize the functionality of your program over a specific type of interface.
- The user interface to the application must allows relief workers to enter information about DisasterVictims, including their relationships, medical records, and location. It must allow inquiries to be logged, and supplies to be allocated to individuals or

added to a location. Relief workers must be able to view and modify existing data (deletion does not need to be supported). The user interface must allow the relief worker to select between the different types of tasks.

- Enforce data validity (do not accept data which is outside of range or of the wrong type).
- In cases where the user has made a typo (e.g., entering a value of Z when acceptable options are 1-4), the program should inform the user of the error and give them an opportunity to correct it.
- In cases where it is impossible for the program to recover from the error (e.g., cannot connect to database), the program should append to a file errorlog.txt in the data directory (creating the file if necessary) and provide the user with a message before exiting cleanly.
- The error log should contain the timestamp of the error and the exception.
- The user interface should not require the user to enter unnecessary text. A CLI should provide pre-populated options when significant typing would be required (e.g., allow the user to select a location by number from a list, instead of asking the user to type the location name).
- The program should not end until the user chooses to exit (i.e., the program should return to a menu when a specific task is completed). The program should provide a clean exit when the user decides to end (i.e., the user should not need to interrupt the process).

2) Database data storage

Storing information between runs of the program is essential for any application. The program will need to connect to a database in order to retrieve, add, and update information.

Task: When starting the program, it should retrieve data from the database, perform any necessary updates (delete expired/obsolete data) and make the data available to the user. The user's additions, deletions and modifications of the data should be reflected in the database.

- Use the provided project.sql file which defines the expected tables and fields and provides some example data.
- You should not add, modify or remove any tables or fields in the database. You may alter the example data.
- When your program is tested, we will use a different file with different test data. The tables and fields will be the same as in the provided file.

3) Gender options

Gender options in the original system were "female", "male" and "other" which does not take into consideration how this usage might impact individuals. In 2018, the Canadian government issued a policy statement on modernizing sex and gender information

practices. In this document, gender is defined as referring to "an individual's personal and social identity as a man, woman or non-binary person (a person who is not exclusively a man or a woman)" and is distinguished from sex assigned at birth. In the 2021 census, the variable "gender of a person" had the following options: man, woman, and non-binary person.

Task: Update the gender options provided by the program to align with current government standards.

4) Language support

Canada has two official languages, as well as a number of other languages which are in widespread use in certain communities or among certain populations. Software which is meant to be deployed in a wide variety of circumstances must be able to be adapted to the language or languages where it is used.

Task: Provide a command-line option when starting the program to set the language displayed to the user through a language and country code. Ensure that all text displayed to the user (aside from data derived from the database) could be translated (i.e., is derived from the language data file, rather than hardcoded in the program).

- The language options which are supported must be dynamically determined by the presence of data files in the format of aa-BB.xml where aa is a two-letter language code (such as 'en' for English and 'fr' for French) and BB is a two-letter country code (such as 'CA' for Canada).
- The user is expected to provide the argument without the file extension (e.g., 'en-CA' if Canadian English is desired).
- If the user does not supply an argument, the provided argument does not match the expected format, a matching data file cannot be found, or the data file cannot be parsed, the user should be provided with an informative error message (in English) and the program should default to using en-CA, which must be provided within the data directory in the final submission.
- Use the same format as the example en-CA.xml file. Your data file may contain any actual data, provided the XML tags shown in the example are used.

5) Family groups

The original method for defining family relationships makes it difficult to retain consistency. There is no guarantee that each side of a relationship will have the same FamilyRelation object, nor any logic as to which person should be personOne and which should be personTwo. Rather than worry about complex family relationships, we have decided to implement family groups which allow us to know that two individuals are related, even if we do not know the exact relationship between them.

Task: Replace the FamilyRelation concept with a family group concept.

- Each person can be a member of only one family group.
- Each family group can have one or more members.

6) Inventory improvement

Supplies consist of individual items of different types. We need to be able to store information related to both the type of the item, as well as about a specific item, and to effectively keep track of if items still exist and where they are located.

Task: Allow for information to be stored about item types and individual items.

- Information stored about individual items consists of a description which is used to define items a person brought with them, such as "a green leather suitcase."
- The following item types need to be supported: personal belonging, blanket, cot, and water.
    - Personal belonging is used for items a person brought with them. It is distinguished by having a description field, with example data like "a green leather suitcase."
    - Cot is distinguished by having a room and grid location, such as "115" (room) and "B6" (grid).
    - Water gets used. When water is allocated to a person, the date it was allocated is stored. One day after allocation, the water is assumed to be used and is no longer shown as existing/available in the system. Water does not expire when it is allocated to a location.
    - Blanket has no special functionality.
- An item can be allocated to a location or an individual, except for a personal belonging, which can only be allocated to an individual.
- When an item is allocated to an individual, it is no longer available to a location.
- Only inventory at the same location as the individual can be allocated to that individual.

7) Anyone can make an inquiry

In the present implementation, only people calling to inquire about disaster victims can log inquiries. The system should be expanded so that it is also possible for disaster victims to provide information about family members they are looking for.

Task: Allow both external inquirers and disaster victims to log inquires.

- No redundant information about a person should be stored.