programming-exercises-repository-tool (pert)

James Vernon

Version 1.0.0

Date Created: 26-Mar-25

Last Edit: 26-Mar-25

# Overview

### Description:

Programming Exercise Repository Tool (pert) is a tool designed to store information about programming assignments in a structured format that can be created, read, updated or deleted (CRUD functionality).

The structured data should be saved to disk and searched for easily. In addition, the data should be able to print to specified format as well as print for git hub markdown files.

### pert minimum functions

1. **Version 1.0.0 (semantic versioning)** 
   1. **CLI menu – for CRUD**
   2. **Data Store**
   3. **Print Formatted Text**

### Environment and Language

pert will be written in C# using the .NET framework and configured for the Windows environment.

# Analysis

|  |  |
| --- | --- |
| Problem Statement(s) | As a person responsible for assigning programming problems, it would be useful to have a program that easily creates and reads programming problems. The information is text based and needs to be formatted in various ways to easily publish. |
| Domain Analysis/User Scope | * The domain in which this tool will be used is the author’s work machine. Design and creation should be considered for other environments. For now though the program will be configured for Windows => 10/11 * The data store feature will need to be abstracted so that either the data can be saved to a flat file or RDBMS system. * Large amounts of text will be passing through the command line. Considerations should be made to print into a readable format * Each problem should consist of the following data properties * Common to all Problems:   + Id - must be unique   + Title – Title   + Type – {programming, hands on,etc,…} * Common To Programming Problems   + Language   + Expected Input – Input sources or types of data to be ingested   + Expected Output – what the program should do   + Hints – any types of related classes or paradigms   + Concepts – a store of searchable terms related to programming   + Requirements – any mandatory concepts required for assignment completion * Description data should be searchable * Concepts, Language, Level, Requirements, and Course should be fields that can be filtered on |
| Must Do | Read, Write, and Search structured data to and to and from a disk store |
| Must Not Do | No compliance or policy requirements as of Ver-1.0.0 |

# (Whiteboarding)Documentation

|  |  |  |
| --- | --- | --- |
| **Entities** | | |
|  | **Interfaces** | IProblem which is an interface for the abstraction of a problem. Or properties that are common to all problems  -id  -Name  -Type |
| **Abstract Classes** | Problem – abstract class that represents a Problem by implementing IProblem |
| **Concrete Classes** | A ProgrammingProblem is a type of Problem |
| **Data Storage** | | Host env |
| **External Dependencies** | | none |
| **Program State** | |  |
| **Testing** | | PertTests |

# Diagraming

A diagram of a problem

AI-generated content may be incorrect.

1. Menus
   1. Main
   2. Should indicate if the data store is loaded
   3. Should only attempt to Load or Create Data Store at this point
   4. If Data Store is loaded
      1. Display Programs by scrollable screen(R)
      2. Index by a number(A)
         1. Print(R) - formatted
         2. Edit
            1. Save(W)/Cancel
         3. Delete
            1. Confirm(W)/Cancel
         4. Format to File(R)
         5. Create New
            1. Save/Cancel
         6. Change Data Store – Create New
         7. Exit
2. Program States
   1. DATA\_STORE\_KNOWN
   2. DATE\_STORE\_UNKOWN
   3. DATA\_LOADED
   4. DATA\_NOT\_LOADED
   5. READY\_TO\_READ
   6. READY\_TO\_WRITE
   7. READING
   8. WRITING
   9. ERROR

MENU