Course Project Overview

Objective: The primary focus of this course is the implementation phase of a team-based software project using software engineering techniques presented in both this course and the prerequisite course. The project scope is large enough to require multiple programmers collaborating to produce a single software product. This **real-world team approach** is the second half of the capstone project experience for McM CS & IT majors and builds upon the prerequisite course COIS-4350.

Project Overview: This course project requires implementation of an object-oriented solution for Chocoholics Anonymous (ChocAn) based on the design produced in the prerequisite course. It must run as a "stand alone" desktop application on a Windows laptop computer such as the HP laptops used by McM students. No external computing resources are allowed. If the solution requires a database, it must be MySQL running on an XAMPP server on the laptop as was used in COIS-3311. Students should already have an in-depth understanding of ChocAn's requirements along with an initial design from the prerequisite course. However, the course instructor may provide feedback that requires design updates during implementation.

The instructor will also provide implementation details and grading criteria (this document). Student teams are expected to develop a test plan and necessary test data to demonstrate complete functionality of the implemented system. The course instructor serves as the sole "customer representative" for resolving any questions concerning system requirements, functionality, and operation. All requests for clarification of system details or changes to requirements must be documented via the course's Moodle forum. The instructor will also serve as the customer's "acceptance testing authority" for determining success of the project and compliance with stated requirements.

Required Deliverables:

1) Zip file containing all source code including non-standard libraries, installation scripts, testing data files, and testing instructions and results such as reports generated during system testing.
2) Java systems must provide an executable JAR file containing all solution components. Systems developed in other languages must provide an executable exe and necessary installation script files. If solution includes a MySQL database, SQL scripts to create and populate the database must be included in as part of the installation scripts.
3) Updated system document with changes to previous sections highlighted and new sections containing implementation workflow and testing details and results.

Grading: The following guidelines will be used in assigning a grade to the project:

- Grade of **C** requires delivery of the source code zip file that is free of syntax errors and warnings and produces a system that runs without execution errors or exceptions and successfully implements all of the **critical** features identified in next section of this document.
- Grade of **B** requires, in addition to above items, successful implementation of all **important** features identified in next section of this document along with an updated system document.
- Grade of **A** requires, in addition to above items, the following items:
  - Detailed installation and operation instructions with scripts.
  - Full suite of test cases with step-by-step testing instructions and expected results.
  - Detailed report of your team's testing results including identification of known deficiencies.
  - List of recommended updates or system enhancements, software can ALWAYS be better!

Due Date: May 3, 2021 with team presentation and demo on May 5th at 10:30am

Course Project Requirements:

- The following items are **CRITICAL** features:
  - All teams must use the GitHub repository to manage the collaboration for this project. The course instructor will create the project and invite students to participate. Basic use of GitHub will be demonstrated in class and additional guidance may be provided via discussion forums.
  - DOS-style console user interface (or GUI in next section).
    - Upon startup, program must display a "splash" screen with the system name, course number and semester, list of implementation team members and a prompt to allow the user to either continue with processing or exit the system.
    - When the user continues with system processing, program must prompt for and obtain the working directory that provides necessary data files for a test run and collects associated output files for test result analysis (must default to the execution directory).
    - Program then displays the main menu screen which must contain an exit option.
    - Subsequent screens would then provide interfaces to selected functions as needed.
  - Program must implement all application features described in the system document produced during the prerequisite course with any changes or clarifications "approved" by the user representative during the current semester (via discussion forum posts).
  - Program must terminate normally without any exceptions when acceptable input data is provided and must display a closing screen message indicating success of the application.
  - Program must terminate normally when unacceptable input data is provided and must display a closing screen message indicating failure of the application (details of issue not required).
  - Program must use programmer-defined objects to store and work with system's data and perform all system processing via programmer-defined methods (forces use of OOP).
- The following items are **IMPORTANT** features:
  - Full graphical user interface (GUI) instead of DOS-style console user interface.
    - Upon startup, program must display a window with your team's system name that is moveable and resizable with standard window controls. It must display system name, course number, and semester as above but the team info gets moved to an "About" screen that is available through a "Help" menu item. The startup screen must also provide buttons and a "File" menu with commands and hot keys to either continue with processing or exit.
    - Upon option to continue processing, GUI must use appropriate file open and file save dialog screens with meaningful titles to obtain necessary directory and file names. Readability or writeability of each must be checked and resolved before continuing.
    - After all directories and files have been selected, display a summary dialog containing names and locations of all selected files and present the user with options to confirm and continue or cancel processing.
    - Upon option to continue, screen must display main system menu with a "status" area to indicate success or failure of previous processing action upon return to main menu. Subsequent screens must be presented as appropriate for selected function. Main menu screen must have an option to exit the system.
    - Screens for displaying reports must be non-editable, scrollable windows. Reports generated as pdf files should open in Adobe Reader windows.
  - Program must perform data validation checks for any input provided by user and react appropriately to prevent subsequent exceptions or failures during processing.