CSCI 2040U - Software Design and Analysis
Project Lab 2 - Rapid Prototyping

Learning Outcomes

Upon completion of this lab, students will have demonstrated the following outcomes:

- Design and implement a high-level, functional prototype within a constrained timeframe.
- Utilize basic software design principles to create modular code that integrates a front end, back end, and database.
- Understand and apply the concept of rapid prototyping for validating ideas before delving into detailed requirements.
- Collaborate effectively in teams, dividing tasks and managing time to achieve a shared goal.

System Description

You have been tasked with rapidly prototyping a catalog management system. The system will allow users to manage a list of items stored in a database. While the exact domain of the catalog (e.g., books, tools, or products) is unknown, the prototype must include the following general functionalities:

1. **Database Interaction:**
    - Read an initial dataset from a simple database (such as a text or CSV file).
    - Save updates back to this database.
2. **Front-End Interface:**
    - Display a list of catalog items.
    - Allow users to view, add, and edit items.
3. **Back-End Logic:**
    - Validate user inputs for adding or editing items.
    - Handle the interaction between the database and the front-end interface.

The goal is to create a functional prototype demonstrating these core features, with the potential to expand or refine it in future iterations.

Expectations of This Lab

Phase 1: Planning (10 minutes)

1. **Team Roles:**
   Confirm roles based on team strengths (e.g., front end, back end, database manager, integrator, tester).
2. **System Overview:**
   Brainstorm and sketch the prototype's architecture. Decide how data will flow between components:
    - **Database (text/CSV file):** Source of data.
    - **Back end:** Logic for reading, editing, validating, and saving data.

    - **Front end:** Displays the catalog and enables user interaction (command-line menus are acceptable).

Phase 2: Prototyping (60 minutes)

Milestone 1 (10 minutes): Database Integration

- Create a flat file (e.g., CSV or text) containing sample catalog data (e.g., `ID, Name, Description`).
- Write a script to read data from this file and parse it into a usable format (e.g., a list of dictionaries or JSON objects).

Milestone 2 (15 minutes): Front-End Interaction

- Implement a basic CLI or menu system to display catalog items read from the database.
- Allow users to select an item to view details.

Milestone 3 (20 minutes): Adding Core Functionality

- Add back-end functionality to:
    1. Edit existing items (e.g., update fields like name or description).
    2. Add new items (prompt user for inputs, validate them).
    3. Validate inputs (e.g., check for non-empty fields).

Milestone 4 (10 minutes): Saving Changes

- Write logic to save any updates back to the original database file. Ensure all changes are reflected when the program is re-run.

Stretch Goal (Optional): User Login (5 minutes)

- Add a basic login system where users must enter a username and password (hard-coded or from a file).

Phase 3: Demo and Wrap-Up (10 minutes)

1. Teams present a short demo of their prototype to the TA.
    - Highlight the ability to view, edit, add, and save items.
2. Show the code structure.

Tips, Tricks, and Resources

Tips

1. **Focus on the Core:** Start with basic functionality (read, display, edit, add, save). Expand only if time allows.
2. **Define Clear Interfaces:** Agree on how each component will interact before implementation to avoid integration issues.
3. **Break It Down:** Divide the work into manageable tasks and set micro-deadlines.
4. **Test Incrementally:** Verify each component works before integrating it with others.

Tricks

- Use a simple text or CSV format for the database to avoid complexity.
- Modularize your code: keep database logic, back-end processing, and front-end interaction separate.
- For validation, enforce simple rules, such as requiring fields to be non-empty or numeric where appropriate.

General Resources

1. **Flat File Formats:** Text or CSV files are easy to manage and read/write programmatically.
2. **Prototyping Strategy:** Prioritize the "happy path" (normal, error-free user flow) first; handle edge cases later.
3. **Time Management:** Allocate time per task and stick to it. If a task overruns, simplify the approach.
4. **Division of Labor:** Assign each team member a dedicated responsibility but encourage constant communication to ensure smooth integration.