**Project Title:** Virtual Assistant

**Student Name:** Juan Carvajal Lara  
**Course:** COMPSCI-2-40364  
**Instructor:** Oke Onwuka  
**Date:** April 16, 2025

**Project Overview**

This project is a virtual assistant built using C++. The assistant interacts with the user through simple text commands and responds with real-time information like date and time, workout routines based on the day and even takes a customized coffee order. It includes interactive behavior on exit and ensures error handling for reliable input management.

The assistant uses object-oriented programming principles, such as classes, memory allocation, and structures, while also demonstrating basic input validation and interaction logic. The design is user friendly.

**Features Implemented**

|  |  |
| --- | --- |
| datetime | Shows the current date and time in 12-hour format |
| workout | Displays an exercise schedule based on the day of the week |
| order coffee | Accepts coffee type and size with input validation |
| weather | Mock weather information display |
| exit | Interactive exit with thank you messages and response check |

**Design Decisions**

**1. Class Design**

The assistant is created using a class Virtual Assistant to capture logic and state. A dynamically allocated structure Coffee Order is used to store and process a user order.

**2. Efficiency and Simplicity**

All memory allocations are cleaned using a destructor to avoid memory leaks. Time and date retrieval uses standard C++ libraries (<ctime>), keeping system resource usage to a minimal.

**3. Error Handling**

Coffee order inputs are validated for accepted values. Invalid commands prompt users to try again. Edge cases like incorrect command casing or whitespace are handled using string techniques.

**C++ Concepts Used**

|  |  |
| --- | --- |
| Classes & Objects | Virtual Assistant class with encapsulated methods |
| Structures | CoffeeOrder struct for user-defined data |
| Pointers | Dynamic memory handling for the coffee order |
| Input/Output | Console input/output using cin and getline |
| Conditionals & Loops | User command handling with error checking |
| Time and Date Handling | Used <ctime> to access and format time |
| String Manipulation | Input validation, case-insensitive comparison |

**Challenges & Solutions**

|  |  |
| --- | --- |
| Input Validation for Orders | Used transform () to normalize and compare lowercase inputs |
| Handling Exit Interaction | Added custom interactive Exit () function with keyword match |
| Cross-Platform Audio Output | Disabled voice output to maintain compatibility across systems |
| Avoiding Memory Leaks | Used destructor to free dynamically allocated memory |

**Conclusion**

This project demonstrates the use of core C++ programming concepts in building an interactive, user-friendly assistant. It shows a clean structure, error handling, and efficiency. The project can be extended further with file I/O, persistent storage of orders, or integration with APIs for live weather or voice output.

**Future Improvements**

* Integrate voice-to-text commands more advanced
* Add user authentication or multi-user session features