Course Title	Programming Fundamentals
Publication Number	CA-PRFND v1-0 Project
Project Title	Inventory Management
Number of Pages	
Released	
Last Revised	

Programming Fundamentals Project

Project 1: Inventory Management System

INTRODUCTION

Today is your first day on the job as a junior programmer. Your immediate supervisor gives you the responsibility to develop an application prototype that simulates the capturing of basic information.

Your supervisor is expecting you to deliver a functioning prototype in six days. This is your opportunity to showcase your analysis, design, coding, and testing abilities in C# using the Visual Studio IDE

OBJECTIVES

The main objectives of this project are to:
- Interpret specifications and analysis performed
- Design a solution based on the requirements and specifications
- Design the logic required for functional solution
- Translate design documents and algorithms into source code
- Use debugging tools, and error-handling techniques
- Validate the solution with test data
- Integrate the knowledge acquired thus far
- Use the features of Visual Studio IDE
- Demonstrate the use of the C# programming language.
- Apply the various program flow constructs.

TIME REQUIRED

You will require 30 hours to complete this project. These hours include 5 in-class sessions plus homework time to complete the application

REQUIRED MATERIAL

You will need the following material to complete this project:
- Microsoft Visual Studio .NET (specifically C# .NET)
- Microsoft Visual Studio .NET documentation, you course textbook or any other reference material suggested or provided by your instructor

SPECIFICATIONS

This inventory management is all about the functionality and not the appearance. This simulation will require the application to display a basic menu of 4 options allowing the user to:

1. Insert a new item in inventory
2. Display a item

3. Display all items
4. Exit the application

Key Functions

The main functions of the simulator may be summarized as follows:

- The menu of options should be displayed to the user at all times after a selection is completed.
- The menu screen must include a prompt to the user to select a menu item and allow the user to enter his or her selection.
- Each option should perform the key functions as describe below.
- The data does not need to be saved (This is just a simulator), but must be accessible as long as the user does not select the option to exit the application.
- Item information must belong to the same when displayed. For example, if 1 is Chocolate and total final amount is $42 when the user selects, 1 to display, all the data for 1 is displayed accurately.
- The user must be able to enter a variable number of s. Regardless of how you decide to store your data (structure, array, multidimensional array, classes, etc) you must allow the user to enter a variable number of items. Alternatively, you can prompt the user to enter the number of s to be entered and create your storage accordingly.
- The important piece is the functionality and not the interface design. (Do not spend too much time worrying about the aesthetics of the project. We are looking for your ability to translate logic into code that accomplishes the necessary tasks.

Detailed Functions

The main functions of the simulator may be summarized as follows
The :
will have four pieces of information as follows:

- A ID (string,) stores a 3 digit numerical string of characters
- A first name (string) stores the short name
- A last name (string) stores the long description
- Price amount (double) stores the price amount of the item

The Menu:
The menu must offer the user 4 options as follows:

- Create a new Item
- Display a item
- Display all item
- Exit

Below the menu, the application must provide the user with a prompt to select one of the options in the menu. When the user enters a valid option, the application must execute the option that the user selected. If the user enters an invalid option, the program must notify the user with an appropriate message and return the user to the menu to make another selection.

Insert a new item :

When the user selects this option, the application must prompt the user for all four elements of data attributed to each as described previously. The data entry must be validated and only accept the proper data type.
If the user enters an incorrect value for the date element requested, the application must notify the user of the error and re-prompt for the data.
When the user is in the function to create a new item, before returning to main menu, the program must ask the user if he or she wishes to enter another . If the user wants to add another new , the program should resume prompting the user to enter the next data. Otherwise, the program should return the user to the main menu.

Display a item :

When the user selects the option to display a , the function must prompt the user for the ID. All IDs are a string of 3 numerical digits (ex: 001, or 123 etc) Hint: keep this simple, do not use complex numbers for the ID.
When the user enters a valid ID, the program must display each data element on a separate line. Each line must have a label for the data element followed by a separator character (hyphen, a semi-colon, three or four periods, or any other separator that you decide) followed by the correct data value. The price amount should be preceded by the $ symbol.
An example of the output could resemble the following
ID: 001
Short name: choc
Long description: Milk chocolates
Price Amount : $49.2

Display all items :

This feature allows the user to display all s that have been entered. Very similar to the above but with the added feature that it will cycle through all the entries and display them in sequence. The display will show each 's data on a single line as the example below. The first line of the display should be a line of data element headings. Followed by the actual data.

ID	short Name	long description	Price Amount
001	choc Milk	Chocolates	$49.2

This feature must also offer an option to return to the main menu when the user selects the option to return to the main menu.

The Exit option:

When the user selects this option the program terminates. Data is not saved and does not need to be restored.

REQUIREMENTS:

Part 1 – The logic:

The first part of this project will require you to apply what you have learned in the Programming Logic and Design course. You will need to analyze the functionality of the various elements of this project and produce your program design documents (flowcharts, pseudocode, class diagrams etc) as required.
You will create the necessary, flowcharts and/or pseudocode to model the logic for the simulator.

When you complete this first part of the project, you must submit it to your instructor for validation. Once your design is approved by the instructor, you may begin the coding of your project.

Note: the part 1 documentation is an essential part of your project, failure to complete the first part and obtaining approval from your instructor (signed off and dated by the instructor) will result in the loss of the marks for part 1.

Part 2 -Interpret the Program Logic to Write Source Code
1. Refer to your analysis and design work, from part 1 and create the code that represents the logic you created. Your code must show the same flow as that which you designed in part 1. For example, if you design an algorithm to use a while loop to cycle through an array to extract a specific data element, your code should include the while loop as well with the same logic.

Error Management: Test the Prototype application

1. You should test as you build to avoid unnecessary delay.
2. Make sure you add appropriate data input validation, and error-handling (HINT: every time you capture a piece of data, make sure it is valid and matches the type of data expected)

SCHEDULE

Note, this schedule is a guideline only. You may require more or less time during each session.

Session 11: Part 1 Analysis and Design of the work to be done
- Analyze the project specifications, and the feature requirements.
- Create your application design. You must use at least two types of documents for the logic component. Suggestion, you may want to use a flowchart to design your program's overall flow, and then use class diagram, and or pseudocode for each individual feature (depending how you decide to create the program).
- Validate your logic with your instructor. Make sure you are able to explain the logic to your instructor and he or she approves it before moving on to the coding phase.
- Typically, you should be able to complete this phase in one session. Otherwise, you should complete it as homework and show it to the instructor at the beginning of the next session.

Session 12: Logic Required by the Processes
- Upon approval of your initial design, you can begin detailing each process. At this point each process should have sufficient detail to match the requirements. You can review the details of each and determine if there are any elements that you need to address based on your instructor feedback.
- Once everything is finalized, you may begin coding your program. Create your project structure in Visual Studio and ensure that all your required files are properly setup.
- Begin coding your application according to your logic documentation. It is essential that you follow your plan.

Sessions 13 to 15: Apply Analysis and Design to Coding the Application
- Apply your problem solving skills
- Code the various processes of your program according to specs .
- Add appropriate data input validation and error-handling
- When coding your application, you may need to revise the logic (iterative development). Make sure you update your design documentation as required
- Document any changes to your logic and design explaining why you decided to modify your approach. This must be submitted as part of your project submission

- Validate the solution using test cases.
- Add any missing input validation or error-handling.
- Update the documentation.
- Submit the project.

TEST DATA

The following is test data to use so that you can verify the basic functionality of your application. The idea is to test then program with data that should work, but also with data that should generate some exceptions so that we can adjust the exception handling processes accordingly. ID	Firstname	Lastname	Purchase	Comment
101	Pulses	Pulses pack	381.65	This should work without issues
200	Lemon	Lemon Box	587.17	The application should reject this and issue a message that the ID is not valid (specs require the ID to be 3 characters long)
234	Mango	Mango Box	587.17	This should work correctly now
984	Apple	Apple Box	Two hundred	This entry should generate a error that the entry for the price amount is of the incorrect type and reject this
Abc	Test	Test description	45	This should be reject because the id is not numerical
1Ac	Test	Test description	20	This should be reject because the id is not numerical

You should create additional test data to test your program. Submit your test data with your program so that your instructor can test it accordingly.
Make sure that any errors that are generated by your application as a result of data validation are properly handled.

IF YOU HAVE TIME (Bonus Marks)

This project is very basic and does not include any complex logic. However, you may find that you did not require all the time allocated to creating the solution. If you have any time left over after having fulfilled all the basic requirements, you may want to attempt to add some additional features to this program. Some examples are listed below.

? Instead of the user entering the 3 digit Item ID, you may design your structure to assign a sequential ID automatically. Hint: Every time the user creates a new item, the program looks at the current item determines the last itemID and issues the next sequential ID to the new item.
? When the user selects the option to display a specific item, the application displays a sequential list of all available item ID's to select from.
? Create an additional menu option to display the item with the highest price amount and display that item's details when selected.
? Add a sort feature to the Display all option so that before it displays the data, it sorts the data in (ascending or descending order based on the purchase amount.

MARKING SCHEME Evaluation Elements	**% of mark**
Analysis and Design Part 1 with complete documentation showing appropriate logic required by the processes/methods (pseudocode and flowcharts) approved by instructor	30
Functioning client simulator with source code based on analysis, design, logic (do all the processes function according to requirements.	25
Proper use of C# syntax and coding techniques (use of program flow structures as required)	20
Documentation preparation and update (if you modify your logic, it must be documented	5
Input validation and error handling using controls and code	15
Error free and bug free application (Application properly debugged)	5
TOTAL	100