System Design Document

*Overview*

*The System Design Document describes the system requirements, operating environment, system and subsystem architecture, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and integrity controls.*

# INTRODUCTION

## Purpose and Scope

This document is intended to provide an in-depth overview of the Contact Manager Application. It will discuss system architecture, the human-machine interface, hardware and software design, and system integrity controls.

## Project Executive Summary

The Contact Manager application is designed to make employee jobs easier. Everyone in the organization will be able to access/modify/create records in this program to assist them in their work.

### System Overview

### The Contact Manager Program will be designed as a console application that provides a way to manage employee records throughout the client’s organization. Given the small number of total employees, the application will not require login credentials, so maintaining user IDs and access privileges will be out of this project’s scope. Users will be able to add, update, delete, search or display all contacts. Contact records will be managed by the ContactManager class and will hold a vector of Contact objects. Contact objects have data members of PersonalInfo and Employer objects. All manipulation of the data will be handled by the ContactManager class, and all i/o will be handled by the Menu class.

### Future Contingencies

It is possible that the organization may grow out of this application. Currently, the application is designed solely for a reference/management tool with minimal security features and functionality. If a more robust application is needed in the future, it is recommended to rebuild the new software from scratch.

# SYSTEM ARCHITECTURE

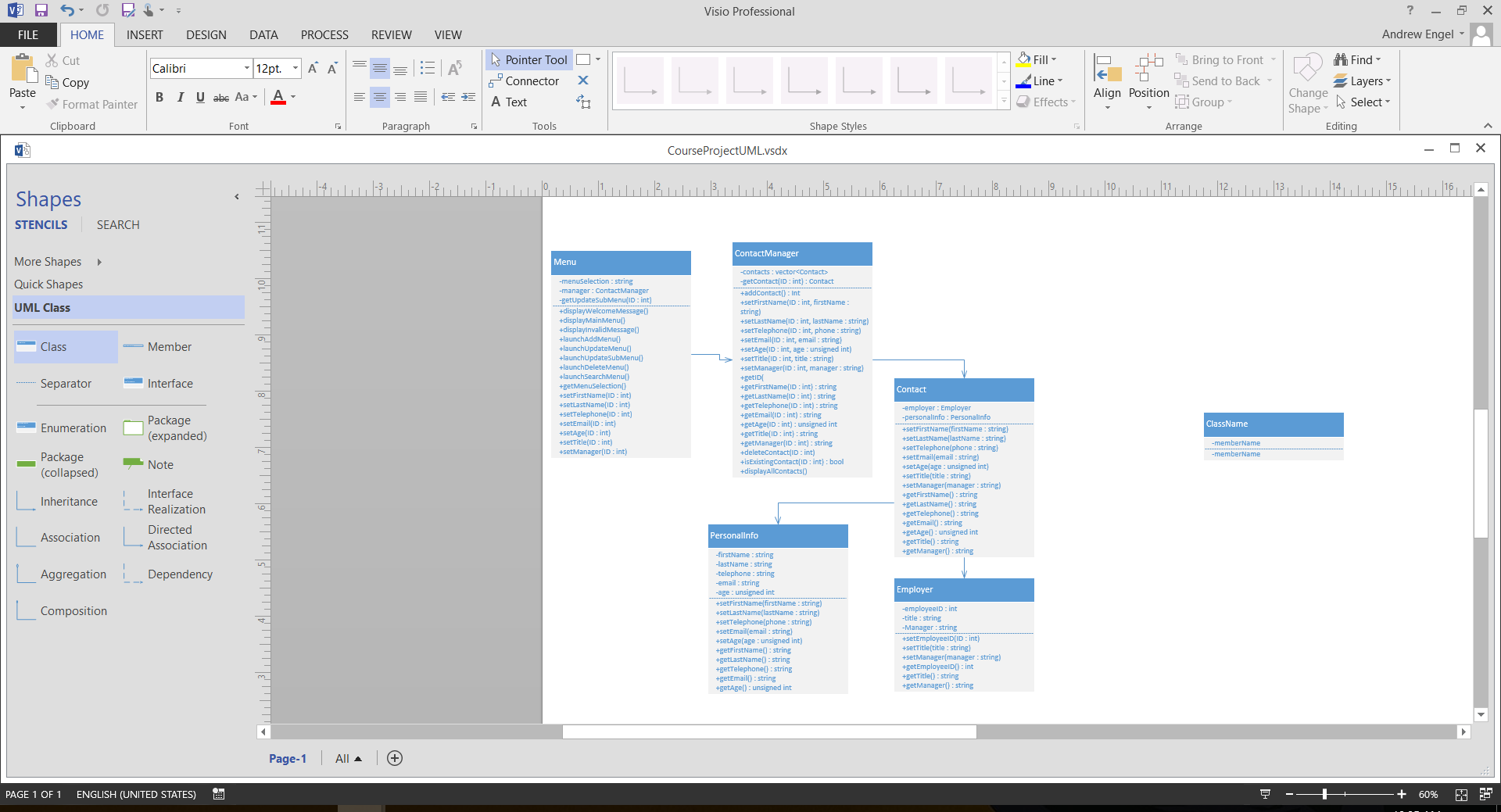
In this section, describe the system and/or subsystem(s) architecture for the project. It details the software’s class structure and the system’s hardware architecture.

## System Hardware Architecture

Throughout the organization, employees will be using desktop PCs and laptops with x86 processors. They are connected to a single network via ethernet and wi-fi, allowing them to share the Contact Manager application.

## System Software Architecture

The Contact Manger application will be programmed using Microsoft Visual Studio Community 2017 and the C++ programming language. The software will be a console application comprised of a main.cpp file, and five classes. The following is a UML class diagram displaying their members, functions and relationships.



# HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator.

## Inputs

The inputs from the user will consist of obtaining menu selections, user IDs, and values to update a contact. The following displays the text for each of the input messages

**Main menu**

Please select one of the following options:

1. Add a contact
2. Update a contact
3. Delete a contact
4. Search for a contact
5. Display all contacts

Q. Exit program

**ID input message for Update, Delete and Search**

Please enter an employee ID:

**Delete confirmation message**

Are you sure you want to delete this record? Y or N:

**Add and update input messages**

Please enter a first name:

Please enter a last name:

Please enter a telephone number:

Please enter an email:

Please enter an age:

Please enter a title:

Please enter a manger:

Validation for entering a phone number, email and age will be implemented. Phone numbers must be 10 digits, and can include or exclude hyphens. Phone numbers will be stored without hyphens. Email address will check to ensure an @ symbol is included. And ages will be required to be greater than or equal to zero.

## Outputs

The outputs for this application will include displaying a single record and displaying all records. Displaying a single record will be in the following format:

ID:

Name:

Phone:

Email:

Age:

Title:

Manager:

The output for displaying all contacts will be in the following tabular format:

ID Name Phone Email Age Title Manager

# DETAILED DESIGN

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product.

## Hardware Detailed Design

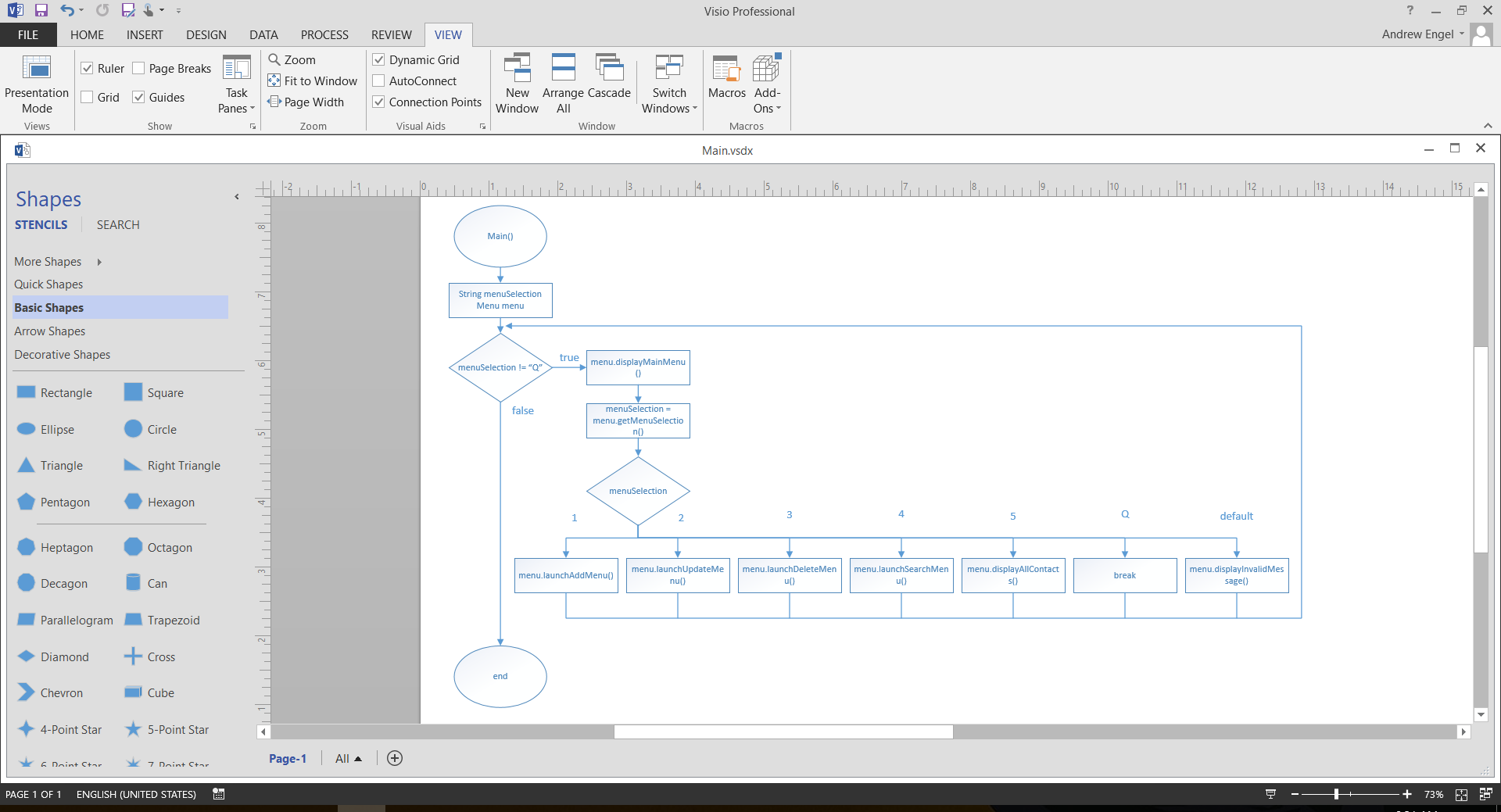
The hardware required for this program is a PC or laptop with an x86 processor. A monitor and keyboard are required to operate this program. Minimum requirements are 800 X 600 resolution monitor, a 450 watt power supply, 3 GB available storage, and 4 GB RAM.

## Software Detailed Design

The following displays the application’s methods, algorithms, and logic flow.

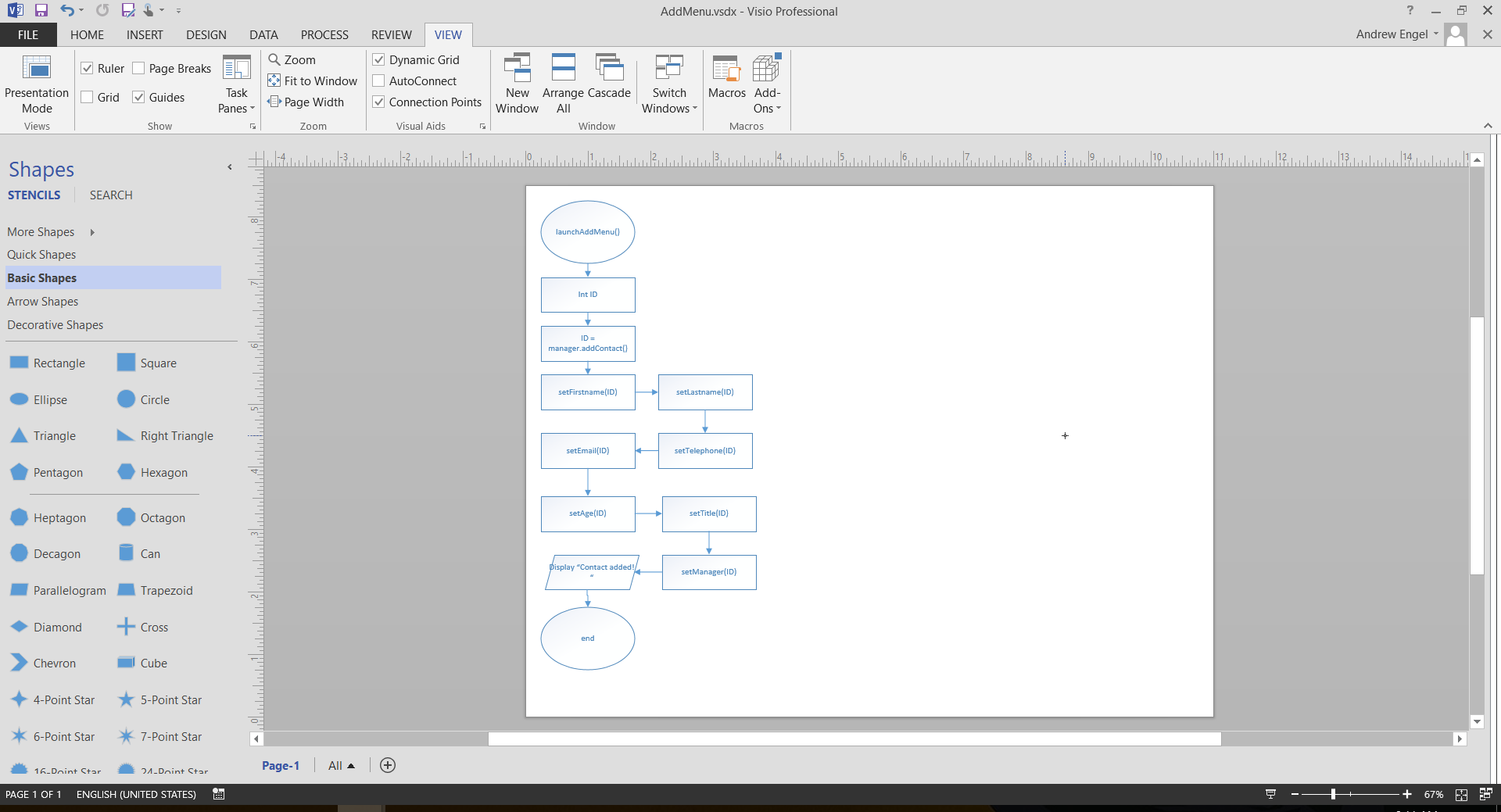
**Main Method**

While the get/set functions of each class will be straight forward, the following flow charts display the functionality of more complex functions. First will be the main method. The main method will declare a string variable and construct a menu object. It will then begin a while loop to maintain a continuously running application until the user selects the option to quit. In this loop, the menu object’s main menu function is called to display options to the user. The main menu function gets the user’s menu selection, which is the passed to the main method by calling the menu object’s getMenuSelection() function. The selection value is then passed to a switch statement that will call a submenu based on the user’s selection. One will launch the add contact menu, two will launch the update contact menu, three will launch the delete contact menu, four will launch the search for a contact menu, five will display all contacts, Q will quit the application, and any other value will display an error message. If Q is selected, the loop ends and the program exits.



**Add a Contact Method**

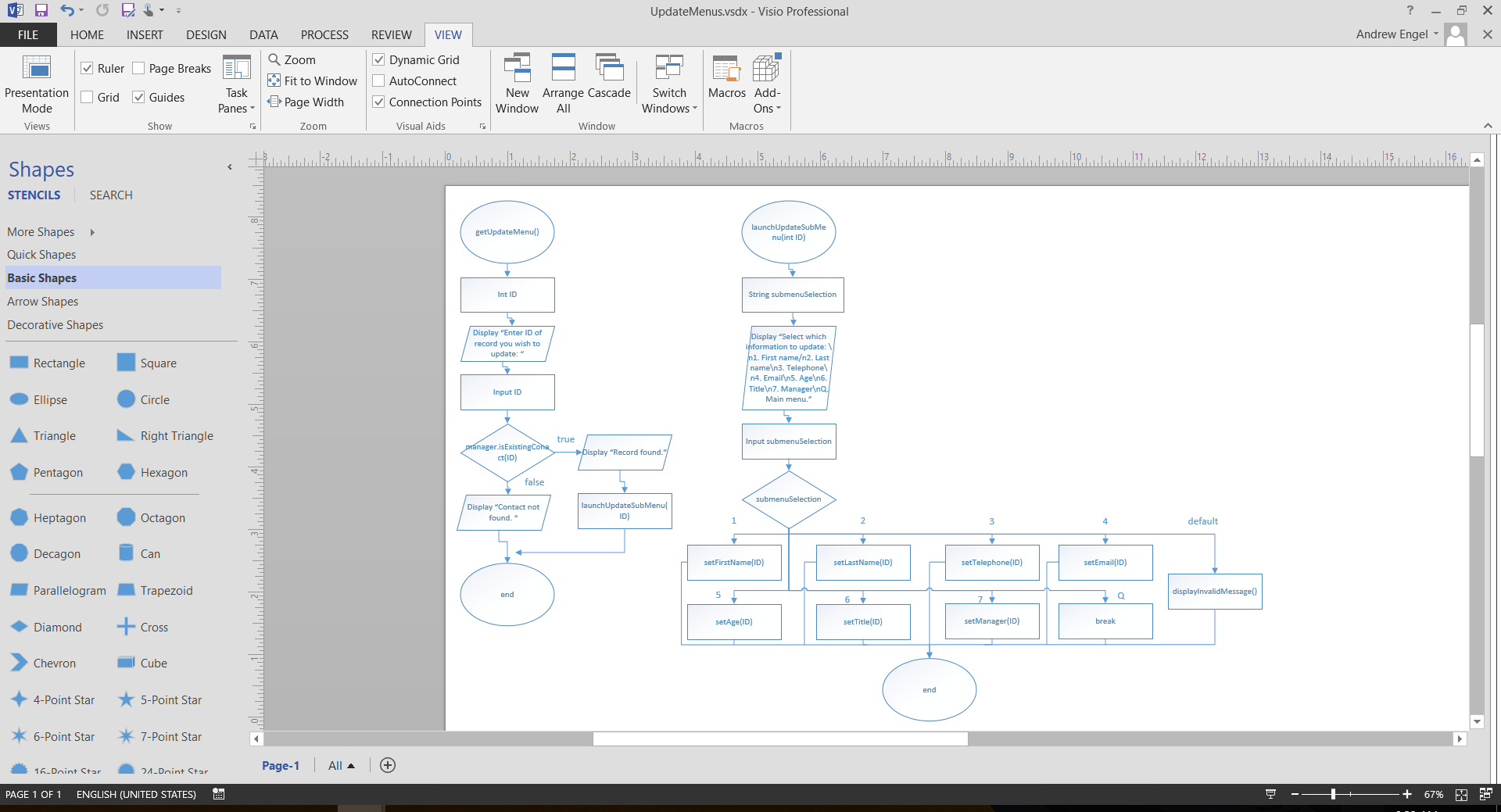
If the user selects 1 in the main menu, the program will call the launchAddMenu() function. This function begins by declaring an integer and assigning a value to it by calling the contactManager’s addContact() function (which returns an employee ID, an integer). Once an ID is obtained, it can be used as an argument to call all of the set functions (which prompt input messages to the user) to populate the contact’s data. After all member data has been set, a confirmation message is displayed and the function exits, returning the user to the main menu.



**Update a Contact Methods**

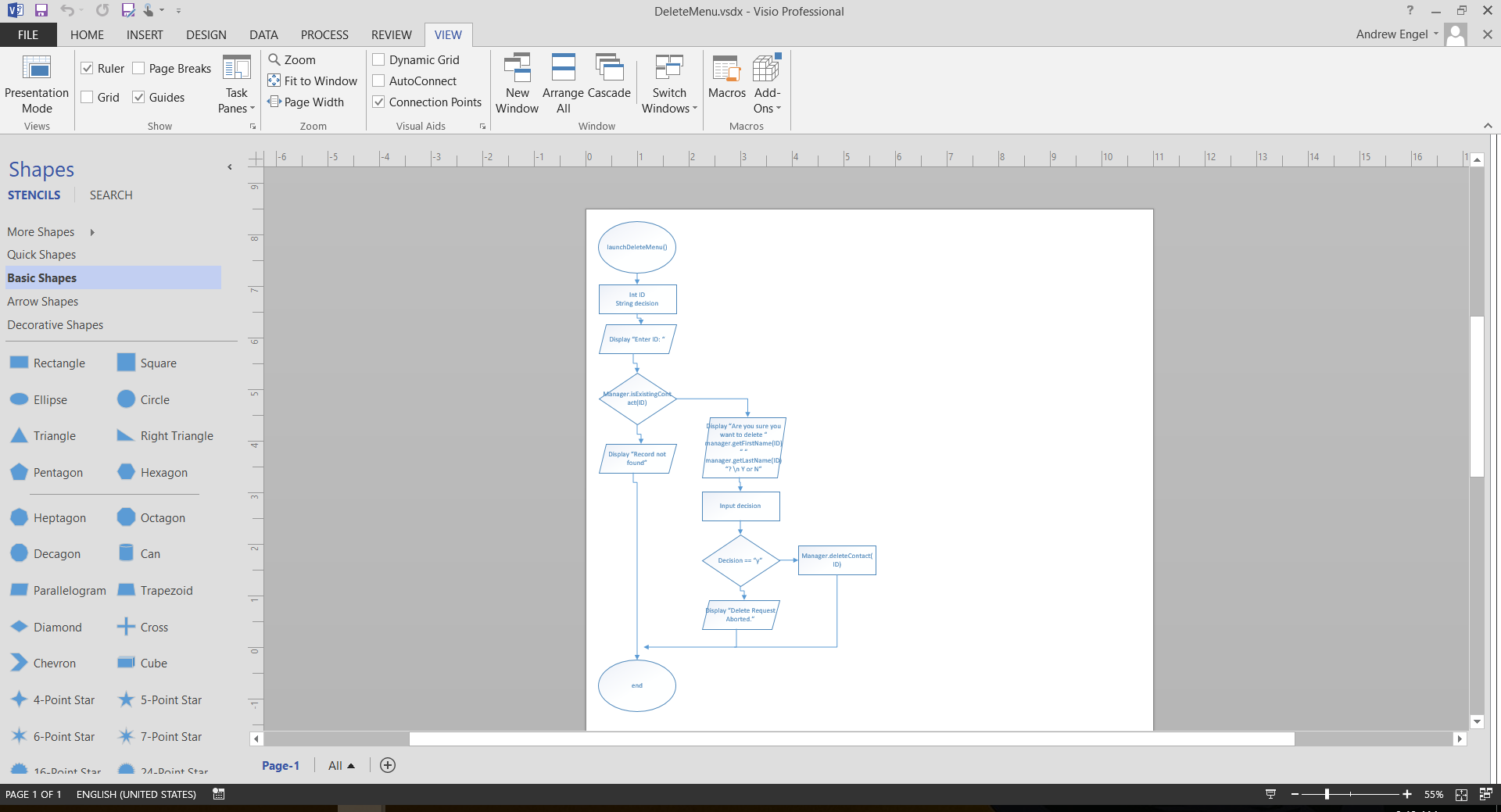
The updateMenu() method is comprised of two menus. The first begins by declaring an integer, and obtaining an employee ID from the user. That ID is validated by the contact manager’s isExistingContact(int ID) method. If it is not a valid ID, a message will display that the contact is not found, and the function will return back to the main menu. If it is a valid ID, a message will confirm that a record was found, and the submenu function launchUpdateSubMenu(int ID) is called.

This method is passed the validated ID, and begins by declaring a string to hold the user’s submenu selection. The submenu text is displayed asking which piece of contact data the user wishes to update. That value is passed to a switch statement that will call a set method, return to the main menu, or call the displayInvalidMessage() function.



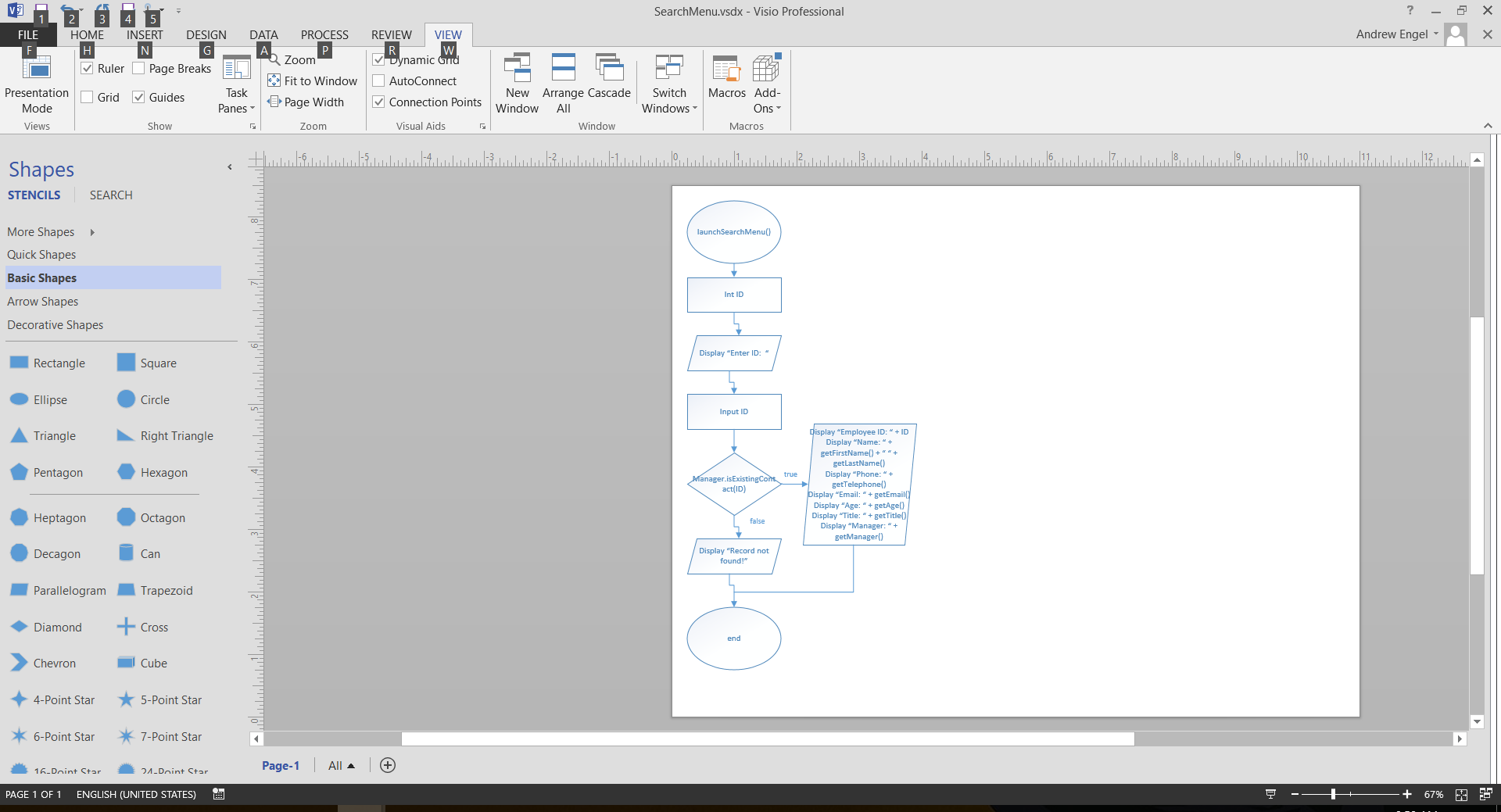
**Delete a Contact Method**

The launchDeleteMenu() method begins by declaring an integer and a string. The integer variable is assigned by asking the user for an employee ID. The ID is validated by the contact manager’s isExistingContact(int ID) method. If false, a message indicating the record is not found is displayed and the user is returned to the main menu. If true, a message displays the contact’s first and last name, and asks the user if they are sure they want to delete this contact. If the user types “Y”, the contact manager’s deleteContact(int ID) method is called, and a confirmation message is displayed to the user. If a string other that “Y” is entered, a message is displayed to indicate the delete request was aborted, and the program returns to the main menu.



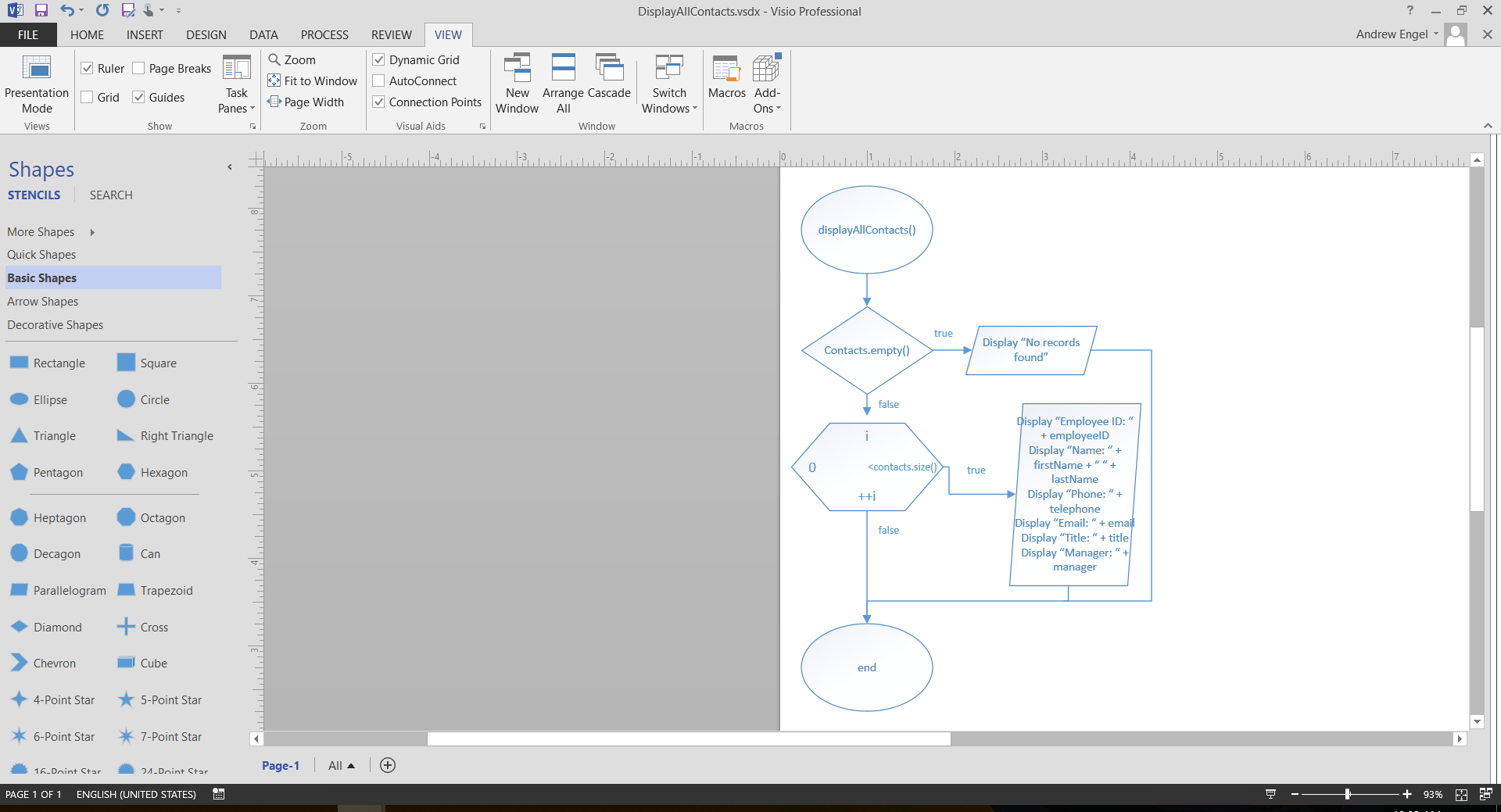
**Search for a Contact Method**

The launchSearchMenu() method begins by declaring an integer. A message is displayed to gather the user ID. After a value is entered, it is passed to the contact manager’s isExistingContact(int ID) method. If it returns false, a message states the contact is not found and returns to the main menu. If it returns true, a message is displayed returning all of that records data.



**Display All Contact Method**

The displayAllContacts() method begins by checking if the vector of contacts is empty. If true, the message “No records found” is displayed. If false, a for loop begins iterating through the vector and displaying each records personal and employer information.



# SYSTEM INTEGRITY CONTROLS

This application is meant to be used as a reference tool to all employees, so the need for security is minimal. All users have access to use the software to its fullest extent, and no user ID/passwords are required. The main safety features are built into the program as validation methods, and asking the user if they are sure they want to delete a record.