Journal of Information Systems Education, Vol. 18(1)

Teaching Case

A Term Project in Visual Basic: The Downhill Snowboard Shop

Mark G. Simkin
College of Business Administration
University of Nevada
Reno, Nevada 89557 USA
simkin@unr.edu

ABSTRACT

Most commercial programming applications are considerably more complex than the end-of-chapter exercises found in programming textbooks. This case addresses this problem by requiring the students in entry-level Visual Basic programming classes to create an application that helps users order ski equipment from a retailer. For convenience, the forms in this project are simplified versions of what users might actually see in a shopping cart application. Although integrative in nature, the case does not require advanced programming skills. However, it requires familiarity with data and control arrays, formatting, loop controls, and parsing, as well as considerable documentation. It is therefore most suitable as an integrative term project. The required tasks can be completed using either VB.6 or VB.Net.

Keywords: Visual Basic, VB.Net, Computer Programming, Systems Project Development, Data Validation, Multiple forms

1. INTRODUCTION

This case requires students to use Visual Basic programming tools within the context of a comprehensive project—a snowboard shopping cart case. The key interfaces required of the developer are a Main form that acts as a switchboard for the project, and purchasing forms that allow users to purchase snowboards, snow boots, bindings, and season lift passes. The project also requires an About window, a separate address screen for shipping information, a Customer Invoice screen, a Payments screen that allows users to pay for their purchases with their credit cards, and a special screen for Federal Express shipping if desired.

The skills required of this project require students to use global, form-level, and procedure-level variables to compute required purchasing values, IF tests, Select Case statements, data arrays, control arrays, and loops. It also requires students to validate selected input values and carefully document their work. An additional feature (that instructors can omit if desired) is the requirement that students create two enhancements. This project is more complex than the chapter assignments found in most programming textbooks, and instructors should encourage their students to plan on spending considerably more time on this case than on simple programming assignments. The author has used this case for five years in a Visual Basic programming course with great success and usually allows students to work in groups.

2. APPLICATION DESCRIPTION

The purpose of this assignment is to create a computer application that helps a snowboard shop prepare customer invoices. Although the final invoice screen is straightforward, there are several supporting screens, and the entire project takes an extensive amount of time to complete. Although a number of programming languages could be used for this task, the windows orientation of the user interfaces particularly lend themselves to the use of Visual Basic. Figures 1-13, which illustrate the user interfaces and system requirements of the project, may be found in the Appendix.

2.1 System Interfaces

This project requires its developers to create a number of user interfaces. This section describes each of them.

2.1.1 Main Screen. The startup interface is the Main screen shown in Figure 1. This screen displays the system date (using DateTime.Today) as the default date (which the user can change at run time) and a series of text boxes for the customer's name, address, etc., as well as an (optional) Email address. The Main screen also allows the user to access several other screens (described below). Regardless of which alternate screen the user might like to see, the system uses a custom subroutine to first check for complete customer information in the form. If any text box is blank, including the

date but excluding the optional Email address, the system displays a message box informing the user of the specific missing data item, sets the focus to the blank text box, and does not display an alternate form window. The system also uses global-level functions (that return Boolean values) to validate the phone number and zip code.

2.1.2 Shipping Information Screen. If the customer's shipping address differs from the address entered in the Main screen, the user can click on the "Shipping Address" button. This displays the Shipping Information screen shown in Figure 2, which allows the user to enter different address information. The system uses Text_Leave procedures to access the custom validation functions described above that check for a valid phone number and zip code. When the user clicks on the Main Screen button, the system closes the Shipping Information screen and returns the user to the Main screen.

2.1.3 Snowboards Screen. If the user clicks on the "Boards" button in the Main screen, the system displays the Snowboards screen shown in Figure 3. This screen allows the user to purchase Snowboards by entering the quantity desired in one or more of the text boxes. When the user clicks on the "Calculate Total" button, the system computes the total dollar value of snowboard purchases, as shown in the figure. If the user clicks on the Clear button, the system erases the contents of the text boxes and returns the focus to the first text box. If the user clicks on the "Main Screen" button, the system closes the Snowboards screen and displays the Main screen again. If the user clicks on the Exit button, the system ends execution.

2.1.4 Snowboard Bindings Screen. If the user clicks on the "Bindings" button in the Main screen, the system displays the Snowboard Bindings screen shown in Figure 4. The text boxes, labels, and buttons in this screen work similar to those in the Snowboards screen.

2.1.5 Snowboard Boots Screen. If the user clicks on the "Boots" button in the Main screen, the system displays the Snowboard Boots screen shown in Figure 5. Note that this screen differs slightly from the Snowboards or Snowboard Bindings screens in that the user must input a "size" as well as a "quantity" for each pair of boots desired. Note also the label at the bottom of the screen, which explains that all boots are available in sizes 6-12, except Viking (7-11), Moto (8-12), and Kids Moto (3-9). If the user accidentally enters a boot size that falls outside these ranges, the system displays an error message in a message box, sets the focus to the first error it encounters, and does not calculate a final total. The code in the click procedure of the "Calculate Totals" button should call a separate (custom) subroutine to perform this validation test.

2.1.6 Season Pass Screen. The snowboard shop offers discount season passes to a local ski resort. If the user clicks on the "Season Pass" button on the Main screen, the system displays the Season Pass screen in Figure 6. This screen allows the user to order up to 4 tickets of each type shown. The system calculates prices by parsing the date in the Main screen and uses a Select Case statement to determine which column of values to use. If the user orders more than 4 tickets of any one type, the system displays an error message and returns the focus to the appropriate text box. The text boxes, labels, and buttons in this screen work similar to those in the other screens.

2.1.7 Order Summary Screen. When done ordering boards, boots, bindings, and season passes, the user can click on the "Check Out" button in the Main screen. The system then displays the Order Summary screen in Figure 7, which formats outputs to currency values. This screen uses labels to display a summary of the customer's order. The customer's first and last names that appear at the top of this form come from the Main screen, and the instructions in the Form_Load event should also compute the grand total as shown. If the user wishes to make changes to this order, he or she can click on the appropriate button, which displays the Snowboards, Snowboard Bindings, Snowboard Boots, or Season Pass screens described above.

2.1.8 Order Summary Screen. The Order Summary screen (Figure 7) summaries the user's various purchases. *This screen uses labels to display all information.* If the user approves of all purchases, he or she can click the "Check Out" button, which causes the system to display the Customer Invoice screen described below. Alternately, the user can click on any of the other buttons, which display the screens indicated and allow the user to change their orders.

2.1.9 Customer Invoice Screen. An example of the Customer Invoice screen is shown in Figure 8. The customer's address comes from the Shipping Screen (if separate shipping information has been provided) or the Main screen (if the Shipping Screen is not used). The total cost of the order comes from the Order Summary screen.

Assume that sales tax is 7.5% of the amount of the order. The cost for UPS shipping is $11.95 and for U.S. Mail shipping is $13.95. The cost for Federal Express Shipping is described below. The cost for insurance is $2 (Standard), $5 (Deluxe), or 3% (Full, computed as a percent of the amount of the order). Finally, the cost for Special Delivery is $14 additional and the cost for Saturday Delivery is $9 additional (both are possible). If the user clicks on *any* of the shipping objects (radio buttons or check boxes), the system changes the appropriate values as well as erases the "Total Charges" label. This forces the user to click on the "Total Charges" button to recompute the final charges.

When the user clicks on the "Total Charges" button, the system computes the total charges for the order. If the Shipping Charges or the Insurance Charges labels are blank, the system displays an error message in a Message box, exits the procedure, and does *not* compute the total charges. If the user clicks on the Quit button, execution ends. Finally, if the user clicks on the "Main Screen" button, the system closes the Customer Invoice screen and displays the Main form.

2.1.10 Federal Express Screen. If the user clicks on the Federal Express Radio button on the Customer Invoice screen, the system displays the Federal Express screen shown in Figure 9. This allows the customer to select a particular type of Federal Express service. Prices are as shown. When

the user selects a particular service and clicks on the "OK" button, the system displays the appropriate amount in the Shipping Charges label of the Invoice screen.

2.1.11 Payment Screen. When the user clicks on the "Total Charges" button in the Customer Invoice screen and *after* performing any other validations pertinent to that screen, the system displays the Payment Screen shown in Figure 10. The user must select one of four types of credit cards for payment as shown by the radio buttons in the form. If the user clicks on the Cancel button, the system displays the Customer Invoice Screen of Figure 8, but blanks out the label for "Total Charges," thus forcing the user to click on the Total Charges button again in the Customer Invoice Screen.

If the user clicks on the OK button, the system first validates the information provided as outlined in the system requirements table below. Of special interest is the way in which the system computes a "check sum" for the credit card that must match the expiration date's month code. To perform this test, the system uses a custom *function* that validates the card number length, numeric content, and the absence of decimal points. This function also repeatedly finds the sum of the digits of the credit card until it finds a single value (a "check digit"). It then compares that value to the month code of the card. For example, if a Visa card number were 1234567890123456, the sum of these digits would be "66," the sum of 6 + 6 would be "12," and the sum of these digits would be 1 + 2 = 3. The check digit is therefore "3."

A similar computation is performed for the month code. For example, if the expiration month of the credit card were "12," then the month code would be 1 + 2 = 3. If the month code matches the check digit for the credit card, the input data are assumed to be accurate. However, if the check digit and month code do not match, the system concludes that the clerk has made an input error. If the entered data fail any of the validation tests for this screen, the system should display a Message box similar to the ones in Figures 11 and 12 and exit the sub procedure for the OK button. If the data pass all the validation tests, the system should simply redisplay the Customer Invoice screen. The type of credit card (e.g., "Mastercard") appears in the banner of the screen in Figure 12.

2.2 System Requirements

This project includes several system requirements as follows:

1. All buttons have access keys.
2. The prices for snowboards, bindings, and boots are maintained by another system, which adjusts the prices shown in the labels of Figures 3, 4, and 5. Thus, the program must obtain these prices from the labels, not hard code them.
3. All dollar figures should be right-justified. Also, only the values in the "Total" line of the Snowboards, Snowboard Bindings, and Snowboard Boots screens and the dollar values in the Order Summary and Checkout screens should be formatted to currency values.
4. Your code should use control arrays to clear the text boxes and compute the totals in the various supporting screens.
5. Finally, your code should validate the items described in Figure 13 using one or more Functions or custom Subroutines.

3. PROJECT DELIVERABLES

A typical computer application contains a number of project deliverables. Sometimes, these items are delivered entirely at the completion of the project. More commonly, they are delivered in phases, enabling the project leader to assess progress during the development process. For this assignment, however, you will hand in everything at once.

3.1 Test Data.

Most programming professionals check their programs with test data. Accordingly, you should test your program with the data provided by your instructor. For each set of data, you should print a complete set of screen captures that show the various test inputs and resultant invoice screen. Collect these materials together, organized by test case, and label this set "Test Data."

3.2 Technical Documentation.

Documentation is critical. Create both internal and external technical documentation that describe your program in sufficient detail to allow someone else to understand it and maintain your code. To do this, print exemplary copies of all form images (at run time) as well as the code for all forms in this assignment, including an About form (not shown). You may print more than one form image on a page, but all code should be clearly identified so that the reader knows what code corresponds to what form. Please omit any system-generated code. Collect this set of pages together and label it "Technical Documentation."

3.3 User Manual

The programmers who create a project are rarely the end users. Thus, you must also create a manual that helps non-technical individuals use your application. This (small) manual should include a general introduction and a description of what each and every button in each form does (see pages 1-3 above). Collect this external documentation for your project together and label it "User Manual."

3.4 Enhancements

This is your chance to be creative. Develop two enhancements for this project, each of which uses an additional form and *document each one completely* (include a screen capture of your form *at run time* and a description of what your enhancement does). Collect these pages together and label them "Enhancements."

3.5 Publication Requirements

Combine the complete set of items described above in a 3-ring binder, *complete with tab dividers for each section*. The title page should contain the title of this project and the name(s) of the developers. Each subsequent page should be numbered, and the first page following your title page should be a table of contents that indicates the starting page of each major section.

4. CONCLUSION

This case requires students to create an integrated Visual Basic application that replicates simplified versions of typical purchasing screens that might be found in an actual application. The skills required of the developer(s) are those typically learned in a first course in Visual Basic, including the use of variables, data arrays, control arrays, loops, selection constructs, and parsing techniques, thereby permitting instructors to assign the case as an integrative term project. Because of the amount of work required by this project, the author recommends that instructors require students to work in teams.

In the author's classes, most student experiences with this project are positive. The individuals who finish it uniformly agree that it requires a lot of work, but also admit that it integrates a number of different, and often isolated, programming skills in a comprehensive application. They also comment that it reinforces the importance of teamwork in completing the various interface and system requirements and helps them appreciate the importance of computer documentation.

AUTHOR'S BIOGRAPHY

Mark G. Simkin is a Professor of Information Systems at the University of Nevada. He earned his BA degree in mathematics from Brandeis University and his MBA and Ph.D. degrees from the University of California, Berkeley. He is the author or coauthor of 15 textbooks, including 3 on Visual Basic and 9 in Accounting Information Systems. He is also the author or coauthor of over 100 research articles, some of which have been published in *Decision Sciences, JASA, the Communica-tions of the ACM, the International Journal of Information Management, the Journal of Accountancy, the Journal of Computer Information Systems, Interfaces, and the Journal of Systems Management.*

APPENDIX: FIGURES

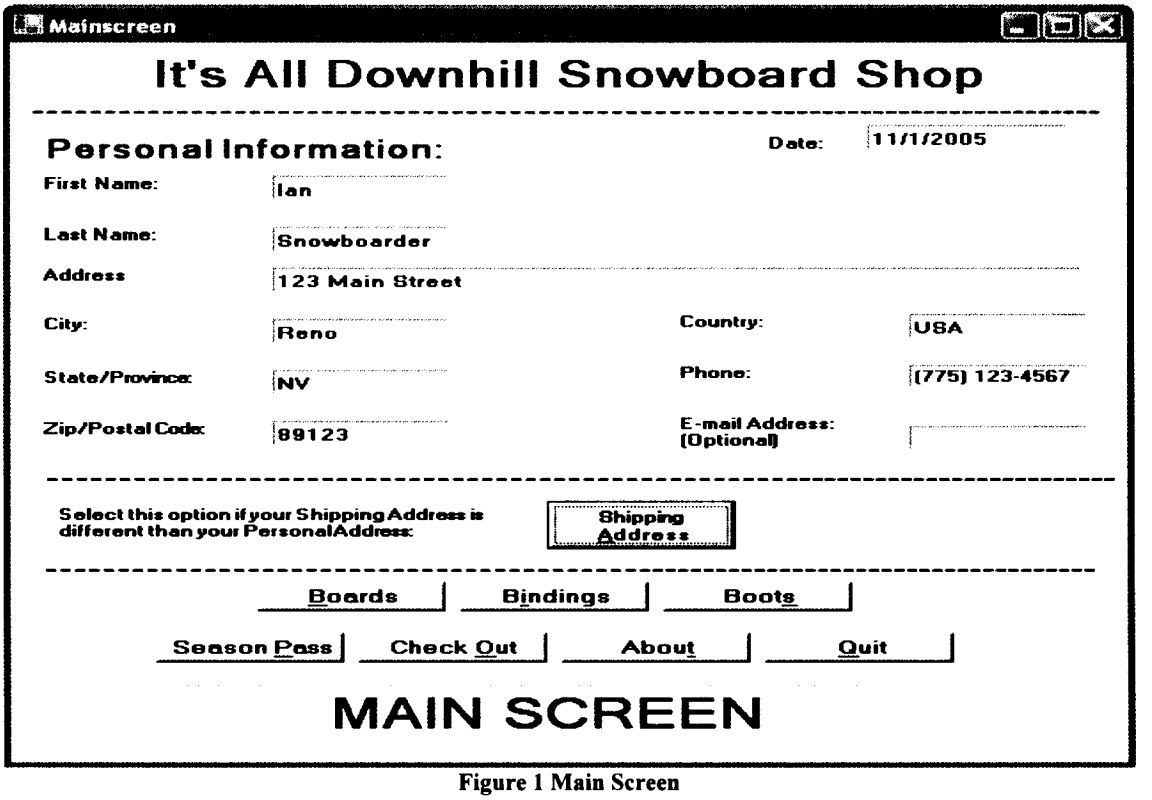

Figure 1 Main Screen

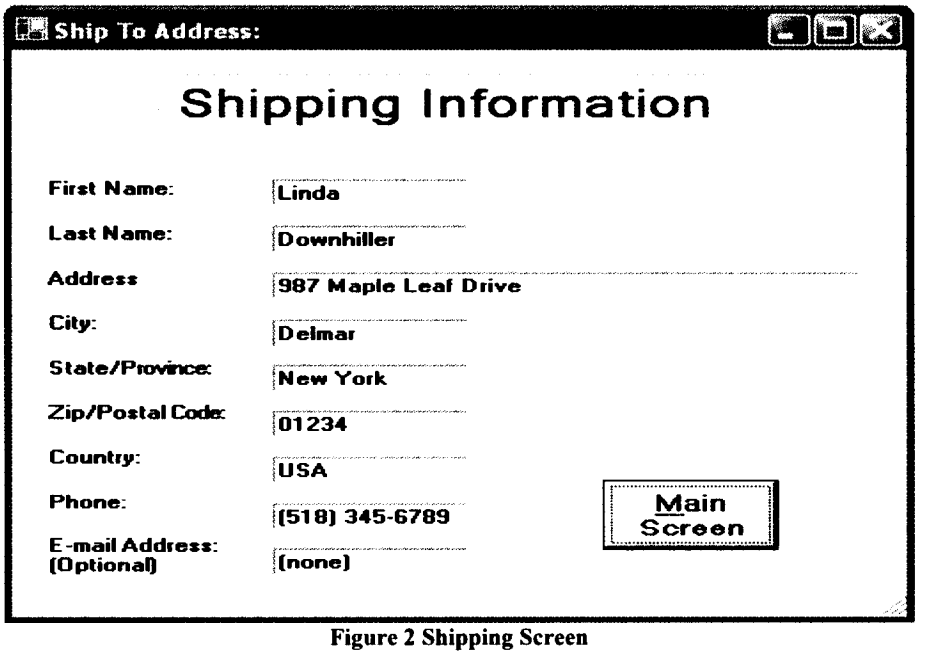

Figure 2 Shipping Screen

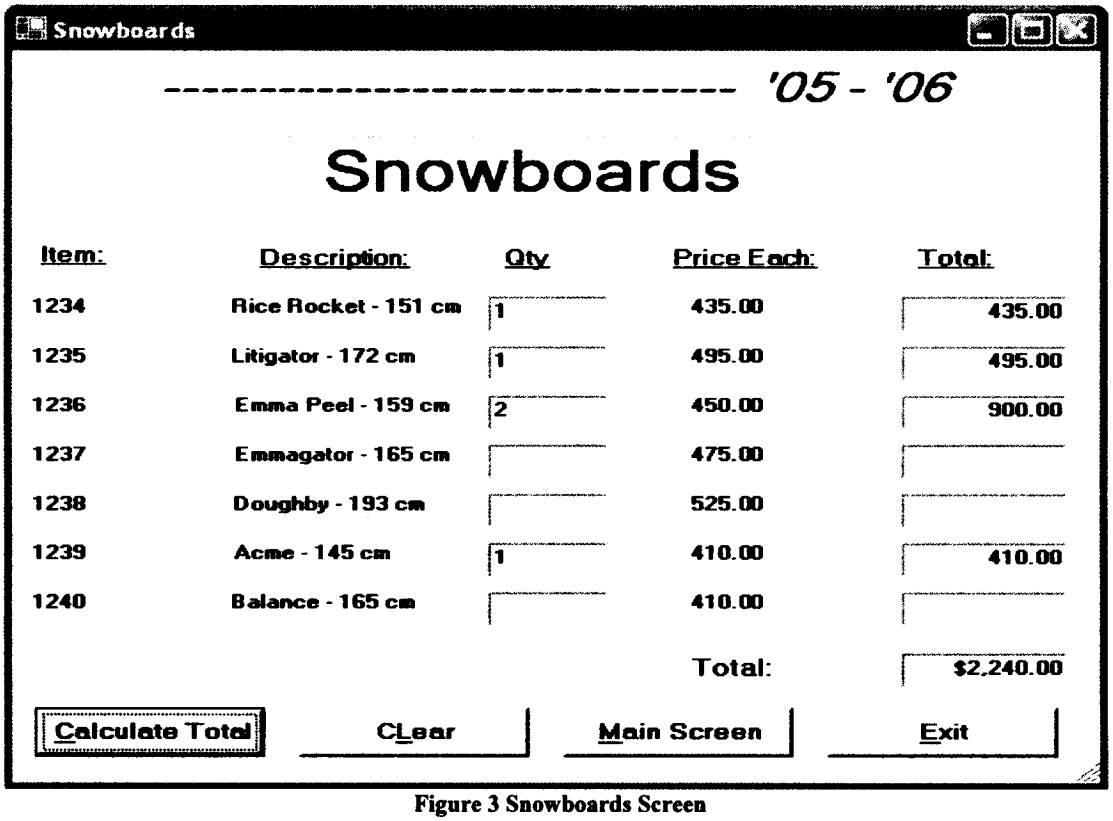

Figure 3 Snowboards Screen

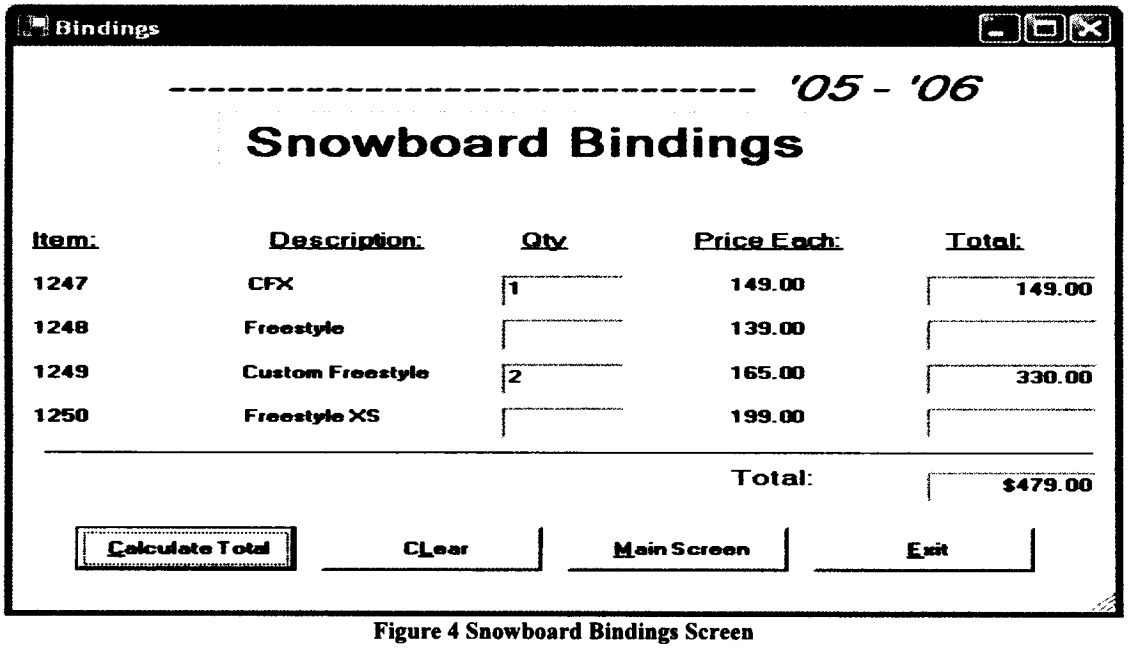

Figure 4 Snowboard Bindings Screen

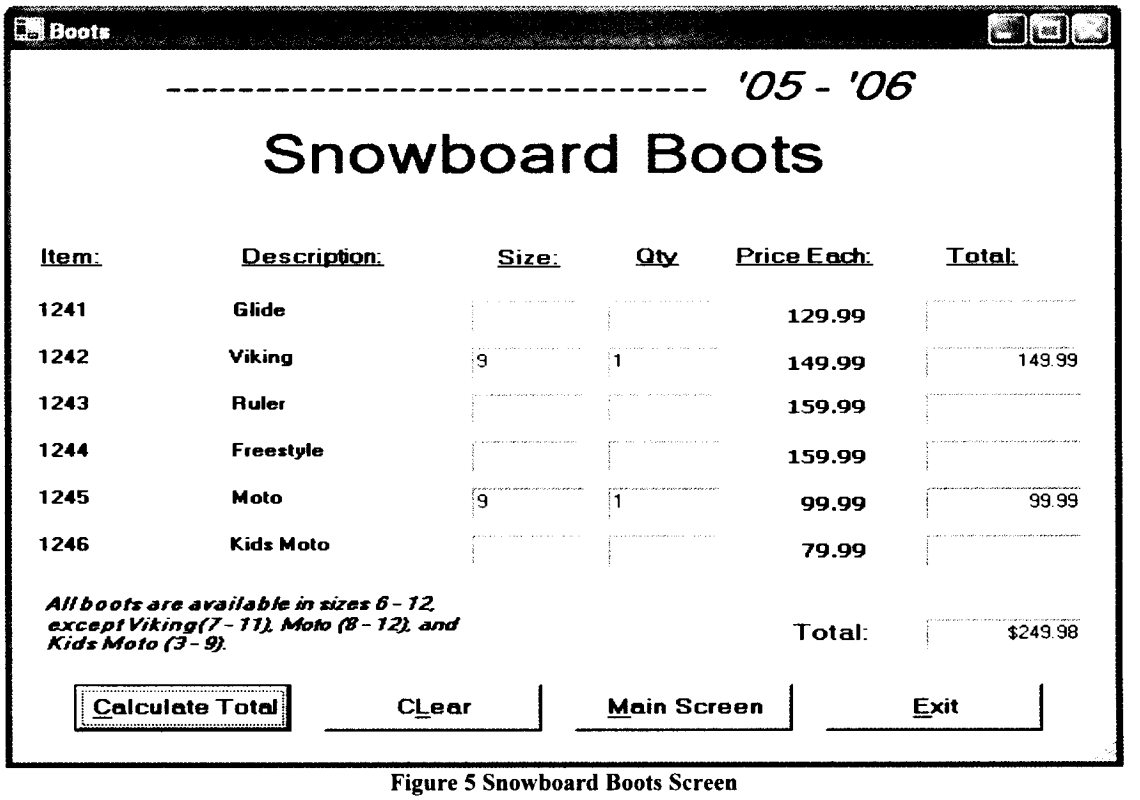

Figure 5 Snowboard Boots Screen

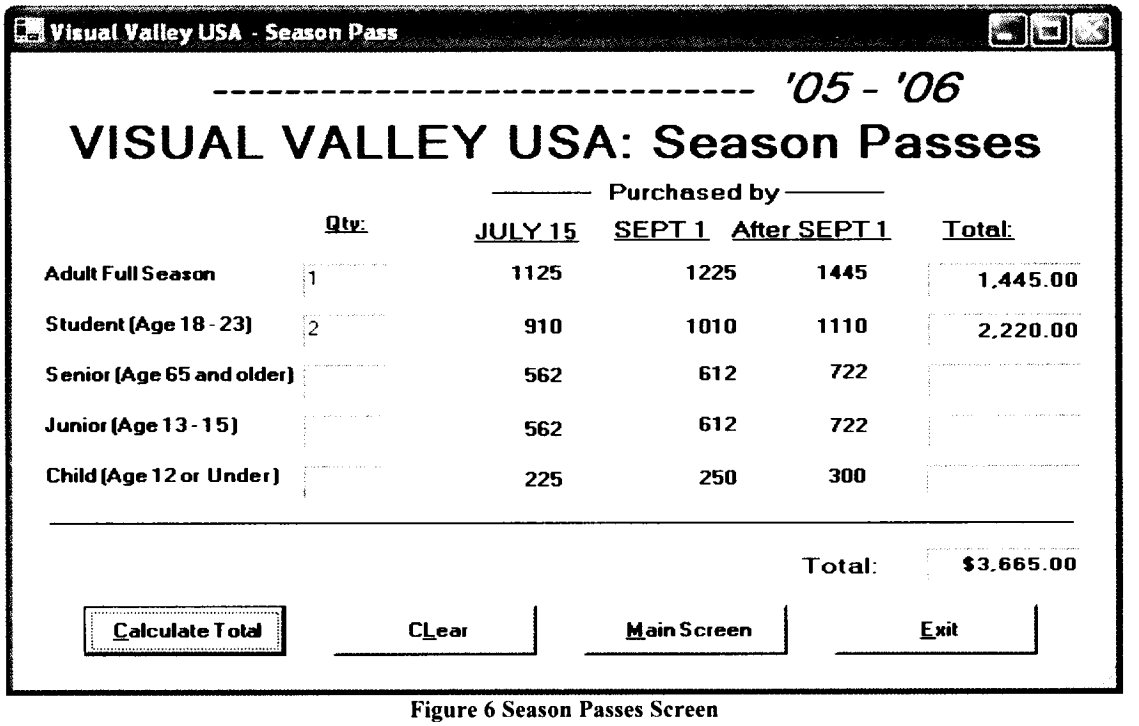

Figure 6 Season Passes Screen

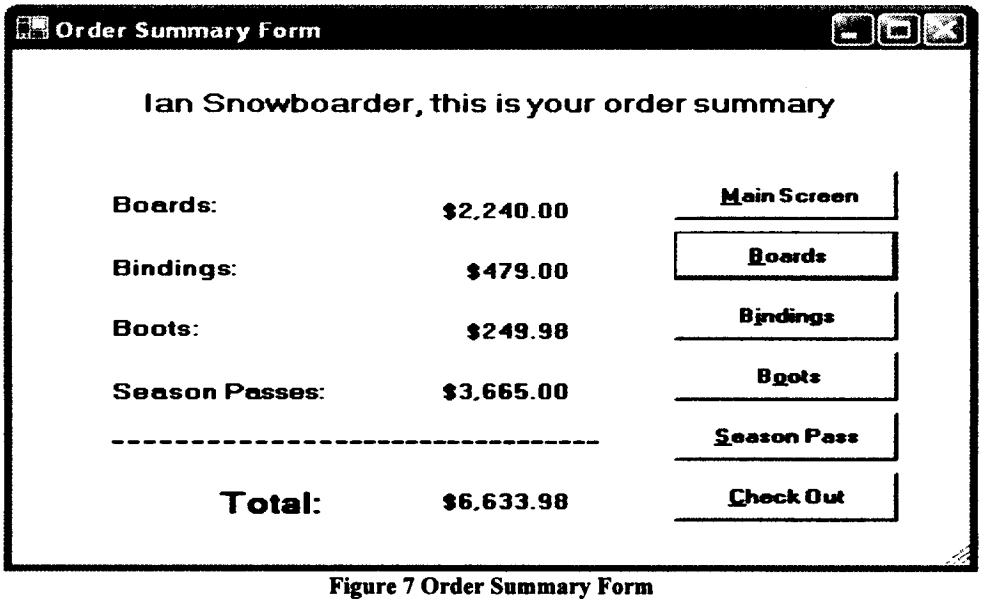

Figure 7 Order Summary Form

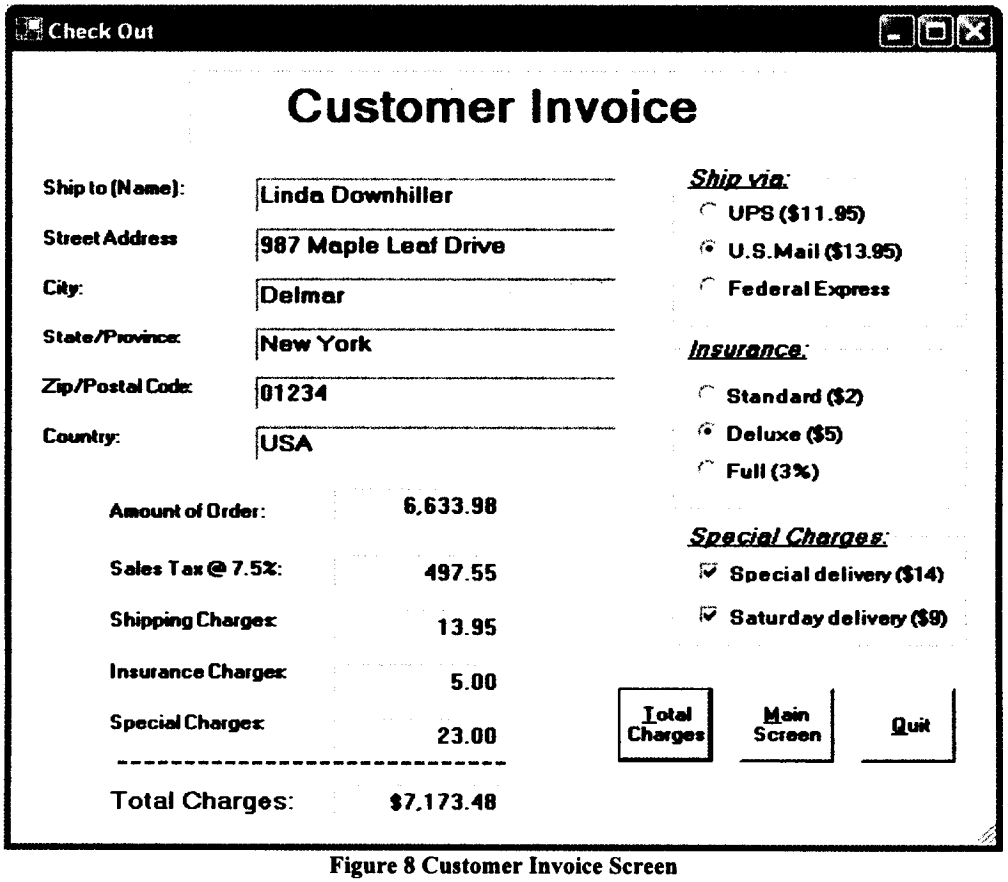

Figure 8 Customer Invoice Screen

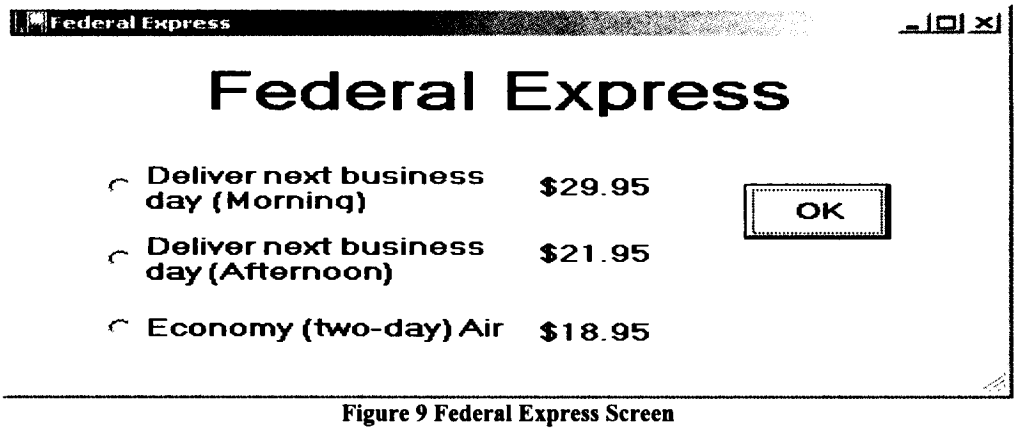

Figure 9 Federal Express Screen

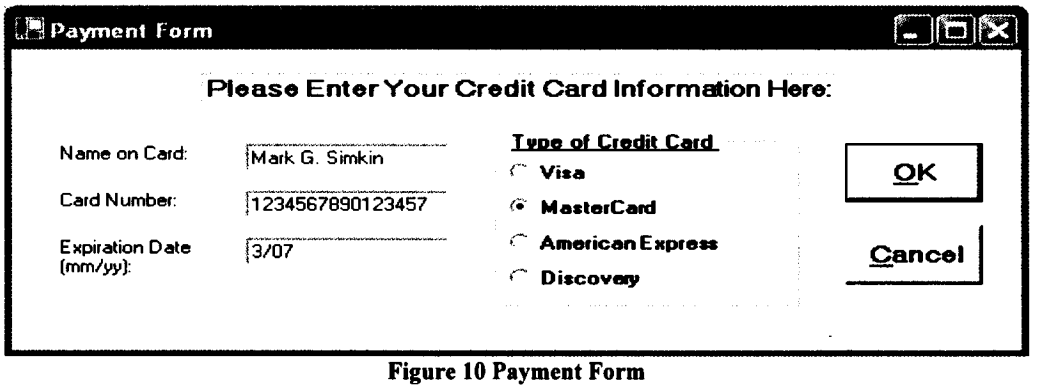

Figure 10 Payment Form

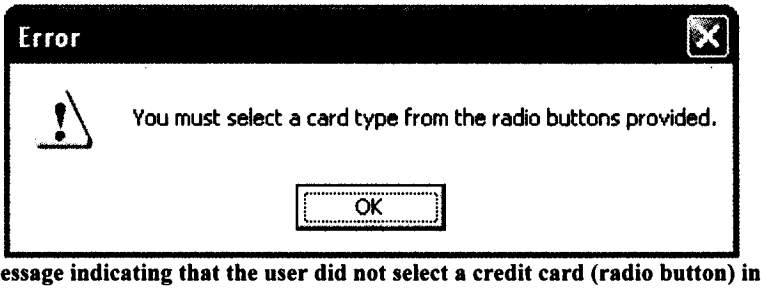

Figure 11 An error message indicating that the user did not select a credit card (radio button) in the Payment Form.

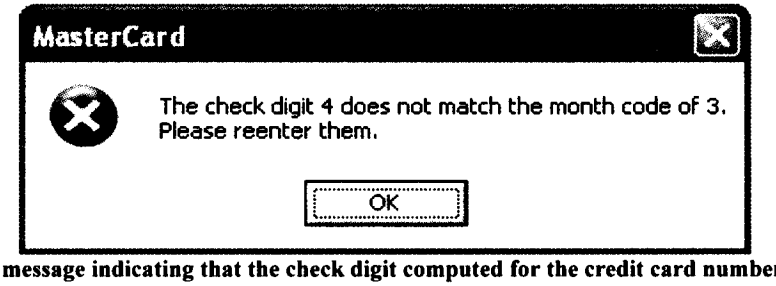

Figure 12 An error message indicating that the check digit computed for the credit card number did not match the month code.

Form	Item	Requirement
Main	Date	If the date is changed, it should be in the form mm/dd/yyyy, where mm is an integer between 1 and 12, dd is an integer between 1 and 31, and yy is 2005 exactly.
	Address values	Last Name, Street Address, City, State, Zip Code, Country, Phone Number are all required fields.
	Zip code	Should be numeric value of exactly five digits.
	Phone	Should be in the form (ddd) ddd-dddd, where the d's are numeric values. There is one space embedded in the phone number, and validation should also test for the parentheses and the dash.
Shipping	Zip code	Same as zip code above
	Phone	Same as zip code above
	All Text boxes	Values should contain information or be blank.
Snowboards	Quantity fields	Must be blank or positive numbers
Snowboard Bindings	Quantity fields	Must be blank or positive numbers
Snowboard Boots	Size Fields	Limited to sizes described in the case.
	Quantity fields	Must be blank or positive numbers.
Season Pass	Quantity fields	Maximum of four per type of ticket.
	Total fields	Should uses prices based on the Date in the Main Screen
Payment Form	Text Box fields	Must not be blank
	Card Number field	Must be exactly *numeric* digits, contain no decimal point, and its check digit must match the month code for the expiration date of the card, as explained in the text
	Expiration date field	Must be a valid month and year (>= year in system date)
	Radio buttons	At least one radio button (credit card) must be selected

Figure 13 Application data validation requirements

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096