Fall 2015 CIS605 Semester-Long Project

Theme Park Management System

Overview

This is a semester-long project, with parts of the project due at various points during the term. The code and user-interface design will be looked at fairly closely in addition to checking for proper functioning of the running system.

The four deliverables, their due dates, and their brief coverage are summarized in the table below:

	Deliverable	Due	Points
1	User Interface	September 17	25
2	Class Design + Basic Code + Input Validation	October 6	50
3	Custom Events + hard coded test data	November 17	75
4	Full System with Arrays + File I/O	December 10	100
			250

For this project you will be developing a system in Visual Basic .Net, using Visual Studio 2012, 2013, or 2015, which implements some of the functionality that might exist with theme park management. Some functionality and data elements have been simplified in order to adequately scope the project for completion during the semester. As the semester progresses, you will be given more details on the project

The basic function of this system is to allow the purchase and use of theme park features. By the fourth project submission, you will have a completed working system that manages customers, features, and many logistics of theme park tracking.

The following use case diagram summarizes the various actors that may use the system. You do NOT need to create separate applications for each. In addition, you do NOT need to worry about user access, privileges, or passwords. You may assume that all actors will use the same program and that each actor will only use the parts of the UI that are applicable to them.

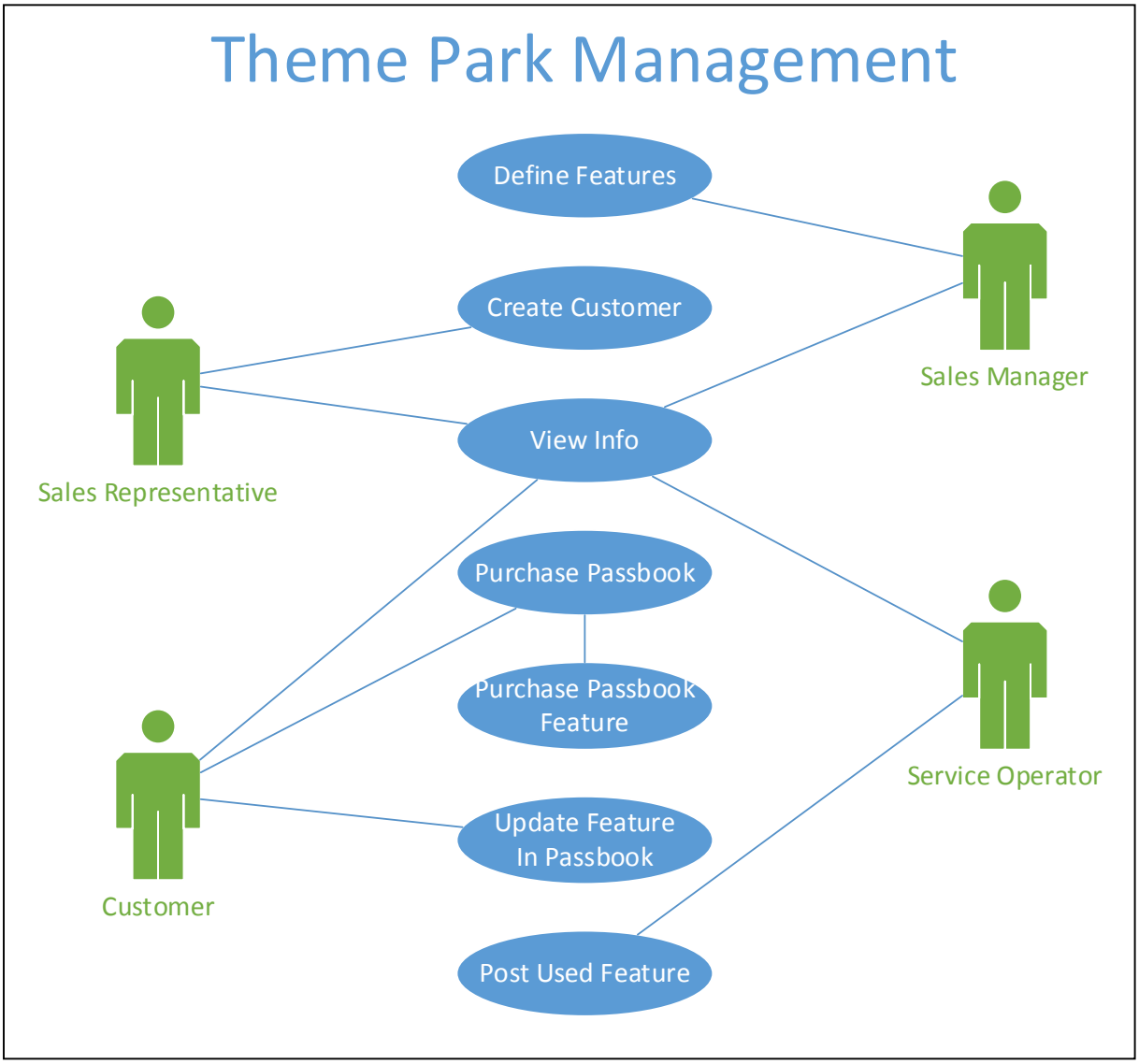

Figure 1: UML Use Case Diagram

The following high level use cases will help you to understand the user requirements better. This list represents functionality for the final project submission, and will be subject to change throughout the semester. Not all requirements are expected until the end of the semester; however, they are provided up front to keep you from going astray early. Carefully read each project assignment to understand the specific functionality required for that particular assignment. **NOTE: some modifications/refinements will likely be made throughout the semester as the project matures.**

<table>
<tr><td colspan="2">Use Case #1: Define a Feature</td></tr>
<tr><td>Description</td><td>A Sales Manager will periodically add features (services) that are sold to customers.</td></tr>
<tr><td>Main Success Scenario</td><td>Manager enters an alphanumeric identifier that is assigned externally and the feature name, a unit of measure (how the service is sold), and the price for Adults and Children separately. To simplify the project, you can assume that the price will never change and no coupons or discounts will ever be used. You can also assume that features are never removed. The data is confirmed and added to the system.</td></tr>
<tr><td>Other</td><td>• Identifier must be unique.
• Feature name and Unit of Measure must be a free form text entry.
• Prices must be decimals.
• Data Examples:
<table>
<tr><th>Feature Name</th><th>Unit of Measure</th><th>Price (Adult/Child)</th></tr>
<tr><td>Park Pass</td><td>Day</td><td>$100 / $80</td></tr>
<tr><td>Parking Lot Pass</td><td>Day</td><td>$15 / $15</td></tr>
<tr><td>Meal Plan</td><td>Meal</td><td>$30 / $20</td></tr>
<tr><td>Early Entry Pass</td><td>Day</td><td>$10 / $5</td></tr>
<tr><td>VIP Pass (cut in line at attraction)</td><td>Attraction</td><td>$...</td></tr>
<tr><td>Birthday Gift Package</td><td>Each</td><td>$...</td></tr>
<tr><td colspan="2">Etc… Free form text should allow any number of services and units.</td><td>$...</td></tr>
</table></td></tr>
</table>

<table>
<tr><td colspan="2">Use Case #2: Create Customer</td></tr>
<tr><td>Description</td><td>A Sales Rep will create a Customer when the Customer has their first interaction with the Theme Park. The Customer is typically the head of the household or the trip planner. Only an ID and the customer name is required. (Other typical fields such as address, email, etc… are not required for this project)</td></tr>
<tr><td>Main Success Scenario</td><td>Manager enters an alphanumeric utility identifier for the customer that is assigned externally and the name of the customer. The data is confirmed and added to the system.</td></tr>
<tr><td>Other</td><td>• Identifier must be unique
• Customer name must be free form text entry. (Given name and surname should be combined into just one entry for simplicity)</td></tr>
</table>

Use Case #3: Purchase Passbook	
Description	Customers can create any number of passbooks which will hold all the features that they buy. One passbook is created per visitor related to the customer. For example, one Customer may be Joe with three of his passbooks belonging to visitors Joe, Jen, and Jack. The Passbook is given an ID, a reference to the owner (by choosing the owner in a dropdown list), the date that the passbook was purchased (always defaults to the system date), the name of the person belonging to the passport, that person's birthdate. The birthdate is then used to calculate an age based on the system date and a determination if the visitor is an adult or a child. A child is defined as anyone under the age of 13.
Main Success Scenario	An alphanumeric number that is assigned externally is entered. A dropdown list is provided of customers and one is chosen as the person purchasing the passbook. A textbox should be provided to enter the visitor's name (given and surname can be combined into one field). A control should also be used to enter the visitor's birthdate. The data is confirmed and the passbook is added to the system.
Other	• Identifier must be unique. • The customer must already exist in the system. • Date Purchased does not need to be entered – this can default to the current system date. • Birthdate must be entered. Age and IsChild must be calculated based on the birthdate entered and the current system date.

Use Case #4: Purchase Passbook Feature	
Description	The Customer must be able to buy Features and apply them to their passbooks. Customers can buy any quantity of any given Feature. Examples of Features are provided in Use Case #1. The amount should be calculated and stored in the object. The amount is the price of the feature (based on adult vs child prices of the passbook visitor's IsChild status) x quantity purchased
Main Success Scenario	An alphanumeric passbook Feature ID that is assigned externally is entered. The customer selects the Passbook and Feature based on drop down lists. When a Passbook is selected, the user should be able to see and verify the passbook visitor name, visitor age, IsChild status, and the customer name (which could be different than the visitor name). When a feature is selected, the user should be able to see and verify the feature name, unit of measure, and correct price of the feature based on the visitor's IsChild status. A quantity is then entered and the transaction is validated and added to the system.
Other	• Identifier must be unique. • The customer and the feature must already exist in the system. • Units must be numeric (decimal). • Quantity purchased should never be negative.

	• Price of the feature must match the visitor's IsChild status

Use Case #5: Update a Passbook Feature	
Description	The customer must be able to increase or decrease any given feature quantity. For example, a customer may have originally purchased 4 days of park entrance feature, but would like to add 2 more for a total of 6.
Main Success Scenario	The sales rep enters or selects the alphanumeric ID that is assigned to the passbook being updated. When the ID is selected, the user should see passbook and customer information, feature information, units remaining, expiration date, and a list of all used features (if any). The sales rep then enters the new amount of total features (not the delta). The update textbox should default to the original quantity purchased. The data is validated and changed in the system: amount and quantity are adjusted.
Other	• Identifier must exist. • Units must be numeric (decimal). • Quantity purchased should never be negative. • Price adjustments must consider the visitor's IsChild status

Use Case #6: Post a Used Feature	
Description	The Park Employees must be able to register the use of a feature. For example, when the visitor arrives at the park, the system is checked that the Park Ticket feature is in the Passbook and that sufficient quantity exist to allow entrance. The location where the feature was used is indicated in a free-form text field along with the quantity used. Quantities could be decimal. For example, late arriving guests may only be charged a half day of park ticket. Or, a lunch may only be a half quantity of a meal ticket whereas dinner may be a full quantity. This does not need to be kept in the system, you can assume the employee will know and enter the correct quantity used.
Main Success Scenario	The employee enters an externally assigned alphanumeric ID that identifies the usage of the entitlement. The employee selects the Passbook and Passbook Feature being used. After the employee selects this information, they should see visitor information, feature information, and the number of features remaining. The employee then enters the number used. Finally, the employee enters the location where the feature was used (e.g. "The 80's Diner" or "Parking Lot A" or "Super Rollercoaster"): this field is always free form text. The date used always is set to the current date (no opportunity to modify it). After the employee enters all the information, the system should check if enough quantity exist to allow usage of the feature and then post the transaction.
Other	• Identifier must exist. • Rejected requests (not enough features left) should display an error.

	• Approved requests should automatically decrement the number of features left remaining.

Use Case #7: View Info and KPI's (Key Performance Indicators)	
Description	All users need to see key indicators and transaction logs.
Main Success Scenario	The user views correct calculations for various types of information. These calculations will be provided as the project progresses. The user also needs to see scrollable lists of all data in the system: customers, services, and entitlements. Finally, the user needs to see a transaction log: a scrollable list of all transactions that have occurred in the system, as they occur.
Other	• Calculations should be accurate at all times. • Be careful to not divide by zero when calculating averages. • More details on this requirement will be provided later.

Use Case #8: Test Button; Read / Write Files	
Description	The system should have a Process Test Data button that runs some "hard-coded" transactions. The system must be able to import data files and export backups.
Main Success Scenario	Users should be able to load transactions from 1) UI per previous requirements, 2) hard coded data in a process test data button, 3) flat data files. Users should also be able to export all transactions in a flat data file.
Other	• Not required for Project 1 • Specific file formats will be provided later. • More details on this requirement will also be provided later.

The following UML Class Diagram summarizes this structural information and the interrelationships between Classes. All data will be stored in memory in the application (i.e. no database management systems will be used). More details will be provided on the business logic classes as the project progresses, and the UML Diagram will be refined over the duration of the course.

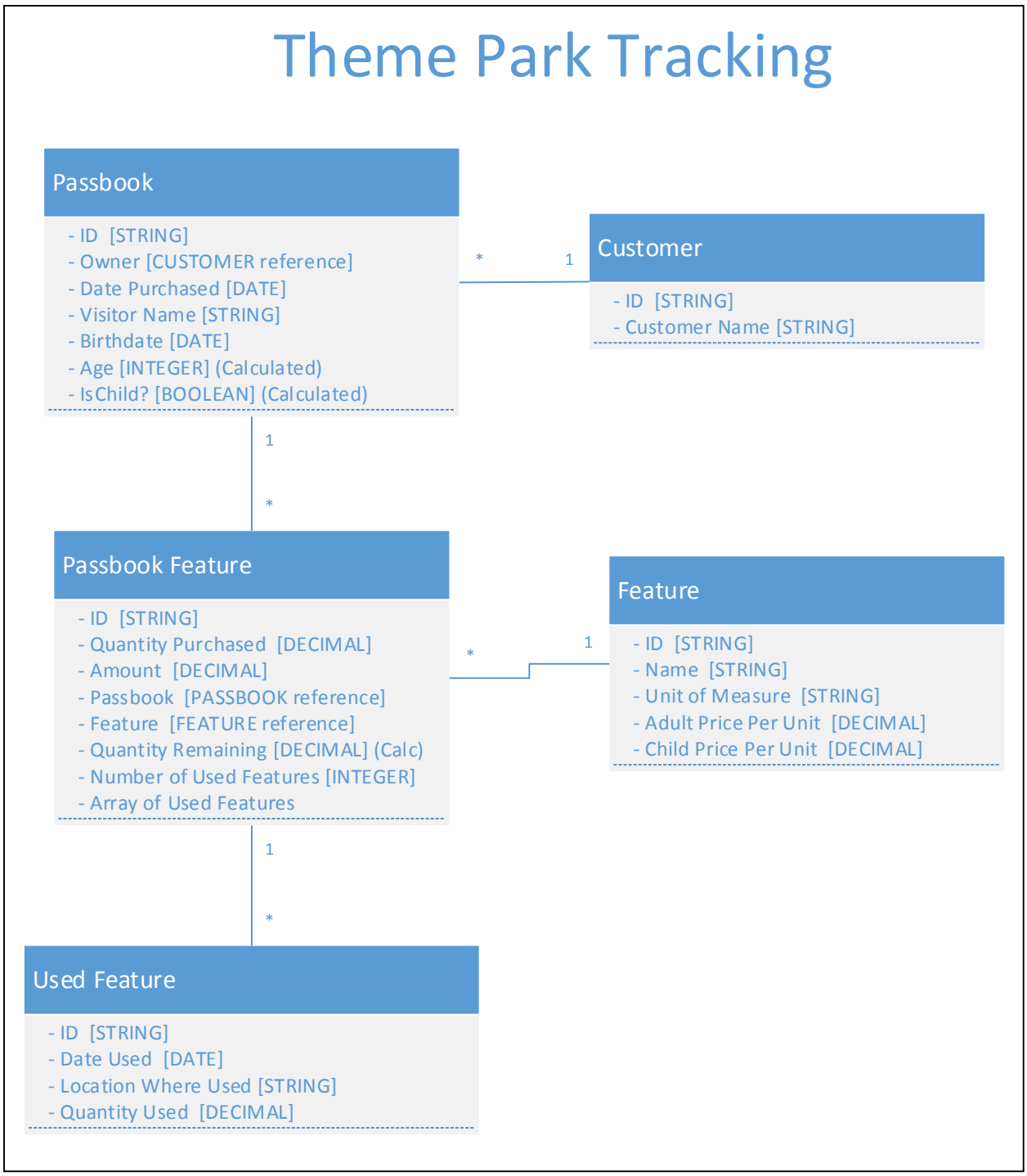

Figure: UML Class Diagram

Semester Project: Deliverable 1 – User Interface

Due: Thursday, September 17, 2015 at 11:59 PM Mountain Time

Points: 25

Solution Name: Proj01-LLLL-FFFF (LLLL = your last name, FFFF = your first name)

Project Name: ThemePark

For this increment of the project, you will be doing an initial User Interface design and coding. You will use many of the user-interface controls that we've been discussing in class: Windows Form, Tab Control, Tab Page, Label, TextBox, ListBox, ComboBox, Button, NumericUpDown, GroupBox, …

The UI will be the primary focus while grading. You should use professional creativity in the bounds of the project requirements. You must have user friendly captions, ToolTips, correct tab stops, and keyboard shortcuts.

A good idea is to organize common things on different Tabs: Summary Info, Customer, Features, Passports, Passport Features, and Transactions would be some good Tab Pages with which to start your thinking. Then put controls on to allow the user to enter data. Use the requirements and class diagram in the section above to decide your layout.

You will have opportunities later to refine the UI as you learn more programming techniques. As the requirements for the summary info/KPI screen are not complete yet, you will not need to put any controls on that tab. All other tabs should have appropriate controls.

Your program can assume that all users will use the same application. Logins and passwords are not required. You will only need to create a single VB.Net program, with a single window, and use multiple Tabs to separate functionality.

A sample rubric is listed below.

CIS605 Project Deliverable 1: User Interface

Topic	Expectations	Points Possible	Points Received	Specific Feedback
General Functionality				
Zip File, Solution, Projects	Program should be zipped properly and include all the necessary files required by Visual Studio	Mandatory to receive a grade		
Projects compile & bug-free	The project should compile cleanly (no compile errors, but compile warnings are sometimes okay)	-20% off of final point total		
	Program should not crash while running			
General Aesthetics				
Look and Feel	UI elements align properly on form	5		
	UI elements appropriately sized/spaced			
	Good creativity while maintaining professionalism			
	User friendly captions and messages with proper spelling/grammar			
	Understandable ToolTips			
	Justification as appropriate (e.g. numbers are right justified)			
Usability	All tab stops correct	3		
	Keyboard shortcuts as appropriate			
General Supportability				
Template & Header	Official class template used	2		
	Header completed (LNs 3-15)			
Naming convention	FrmMain and all other code files are named properly	5		
	UI elements are prefixed correctly (per Appendix C) and named in clear business terms (exception: static labels)			
Required End User Functionality (Graded as the project is running)				
General	Understanding of requirements (Code is NOT required, but UI controls must be present):	10		
	o Multiple tabs that organize similar functions together			
	o Summary/Dashboard			
	o Customer			
	o Feature			
	o Passport			
	o Passport Feature			
	o Transaction Log			
	o Command button controls to perform actions (Create, etc…)			
	o Other controls (labels, text boxes, date/time pickers, etc…) necessary to meet requirements			