Zach Smith

CIS 410

Portfolio

4/9/2018

Table of Contents

1.1 Resume

My resume depicts the knowledge I have gained through my time spent at University of Louisville and my work experience. It gives a summary of what I have done up until this point and shows that I have acquired the skills necessary to be successful in the information technology field.

Zachary Smith

4435 Dyer Avenue, Louisville, KY 40213 | (502)-457-0442 | zrsmit04@louisville.edu

Education

BACHELOR OF SCIENCE IN BUSINESS ADMINISTRATION | MAY 2019 | UNIVERSITY OF LOUISVILLE

- Major: Computer Information Systems
- Concentration: Information Security
- Related coursework: Information Technology Ethics, Software Development, Database Management, Business Communication, Database Management, Systems Analysis and Design

Skills & Abilities

INFORMATION TECHNOLOGY

- Microsoft Office Suite
- Microsoft Project
- Visual Studio
- Visio
- SQL Server Management Studio
- Windows
- C#
- SQL
- HTML
- CSS

COMMUNICATION

- Excellent written and verbal communication skills.
- Strong background in customer service.

LEADERSHIP

- Proficient in leading small teams to achieve goals and meet deadlines.

Experience

ENTERPRISE HELPDESK TECHNICIAN | ADVANCED BUSINESS SOLUTIONS | 5/2017 – PRESENT

- Troubleshoot software and hardware both over the phone and in person.
- Manage small team projects.
- Create and distribute weekly phone reports for enterprise helpdesk.
- Provide excellent customer service both other the phone and in person.

DELIVERY DRIVER | PAPA JOHNS | 12/2016 – 4/2017

- Accurately take payments from customers and provide change.
- Provide excellent customer service both other the phone and in person.

PACKAGE HANDLER | UPS | 8/2014 – 2/2015

- Work as part of a team to meet deadlines.
- Load and unload packages from various aircraft.

2.1 System Request

The system request document is the first step in a system analysis and design project. It clearly states the project sponsor who will serve as the primary point of contact for the project. This document explains the business needs for a project and the benefits of completing the project. It provides the system requirements and acknowledges any special issues or constraints for the project.

System Request

The Project sponsor for Surgery on Sunday is Barbara Martin. She is the main point of contact for this project. The president and CEO of Surgery on Sunday is Erica Sutton. She can be contacted if needed, but because of her busy schedule, it is preferred that we contact Barbara.

The business need for Surgery on Sunday is to improve the current IT system and processes. This improvement is needed to increase productivity and prepare for business growth. The current IT structure is running through “G-Suite” using Excel workbooks to store donor and volunteer information. The business needs a database system to replace the current Excel workbook structure. The Surgery on Sunday website will also need to be adjusted to work with this new database structure.

These adjustments will provide Surgery on Sunday resources to run their business smoother. An improved database structure will allow the business to match volunteers with surgeries in a more efficient way. The database will be relational, allowing the business to search for volunteers that fit certain criteria for a surgery. This should cause a 10% increase in surgeries performed within the first year. The improvement of the website should also cause a 20% increase in donations, due to an easier donation process.

Problems we will have to overcome while implementing this system include a currently unknown budget, HIPPA compliance and a limited IT department. The team does not currently know the budget SOS has for this system. This will make it hard to gage how complex we can make this system. We will not be dealing directly with patient data, but we will still need to make sure all HIPPA rules are being followed when the system is being implemented. SOS currently has a one-person IT department. Implementing a brand-new system is going to be tough for one person.

2.2 Team Charter

A team charter details a project's scope, participants, and objectives. It explains the roles and responsibilities for the team's members as well as the goals of the project. It dictates when meetings will be held and how decisions about the project will be made. The team charter serves as a reference for team members responsibilities.

Solution Squad

Team Charter

Client: Surgery on Sunday

1) **Team Members**
   - **a)** Devan Henley
   - **b)** Tommy Tran
   - **c)** EJ Deguzman
   - **d)** Zach Smith
   - **e)** Kyle Jones
   - **f)** Brandon McWilliams

2) **Goals**
   - **a)** As the Solution Squad, our goal is to provide the client with a project team that works as a cohesive unit that ensures the delivery of fully functional and affordable system to support their business processes. We are very eager to begin our work with Surgery on Sunday and ready to meet the demands and surpass the expectations of our clients. Through the development of the system, the goal is to improve on some current business processes, as well as introduce new business processes that will allow for more fluency across the organization.
     - **i)** The current website lacks volunteer, donor and patient FAQ.
     - **ii)** A process that would allow volunteer information from the form to be mapped directly to the volunteer database.
     - **iii)** Due to a high volume of Spanish-speaking patients, it would be beneficial to implement a widget that would translate the current website to Spanish if prompted.
     - **iv)** Due to the growing change in our society it would be beneficial to accommodate all volunteers and patients by including a transgender option into the forms.

3) **Meetings**
   - **a)** Meetings for the Solution Squad will be scheduled in class every Tuesday and Thursday. Once we come to a consensus for when and where the meeting will take place, we will post it in the Solution Squad GroupMe. We will conduct meetings by first addressing issues at our current stage in the project, then collaborate on potential solutions and then bring forth any new business concerning the current or future stage of the project. Decisions during meetings will be documented in our Solution Squad google drive that

we created specifically for the project for Surgery on Sunday. The google drive will allow for us to have access to any information we may need at any time throughout the project.

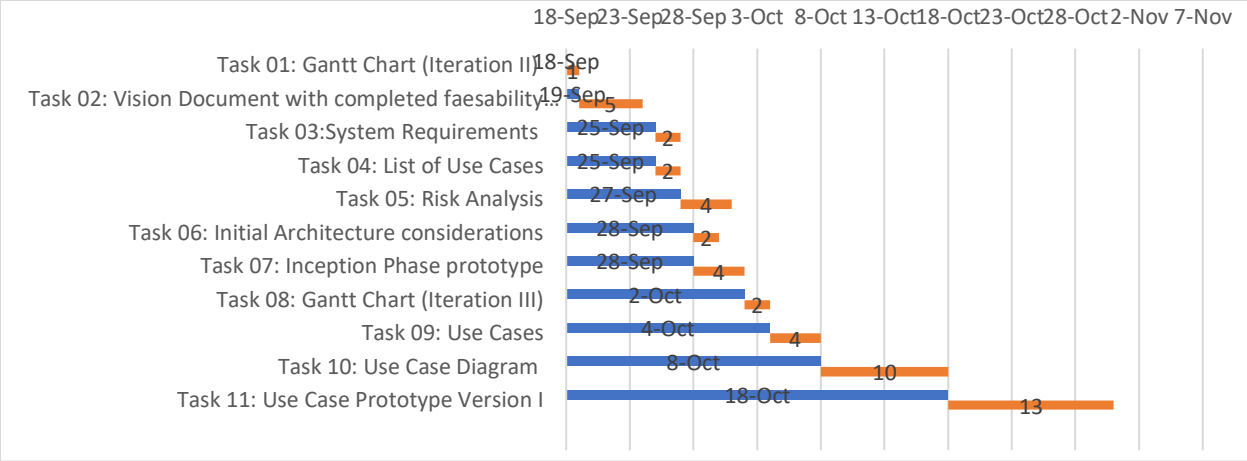
4) **Communications**
   - **a)** Member communication
     - **i)** See section 3 (Meetings)
   - **b)** Client communication
     - **i)** Our designated mediator between the project team and the client will be the main source of communication between the team and client. We will contact the client via email regarding ideas, technical materials, and decisions.
   - **c)** Instructor communication
     - **i)** Communication between the project team and instructor will be done mostly on our Tuesday/Thursday class meetings. However, under special circumstances we will reach out to Professor Barker via email regarding any new ideas, technical materials, or decisions.

5) **Decisions**
   - **a)** The Solution Squad will take every aspect of an issue into consideration when deciding when deciding on any idea that will affect the system. When we decide on issues for the client we will use a voting system. As a project team of 6, that would leave room for an issue to come to a 3-3 vote. In such a case, we would open the decision back to the floor, present the idea again and either settle for an alternative or reach out to the client to ensure satisfaction in our decision.
   - **b)** Any conflicts that arise within the project team will be handled similarly to the way we build a consensus and make decisions.

6) **Project Documentation**
   - **a)** The Solution Squad will maintain project documentation using a shared google drive. This drive is what will contain all documents from the team's deliverables for each iteration, as well as any information received from the client regarding questions we may have about the current and future stage of SOS's system.

2.3 Gantt Chart

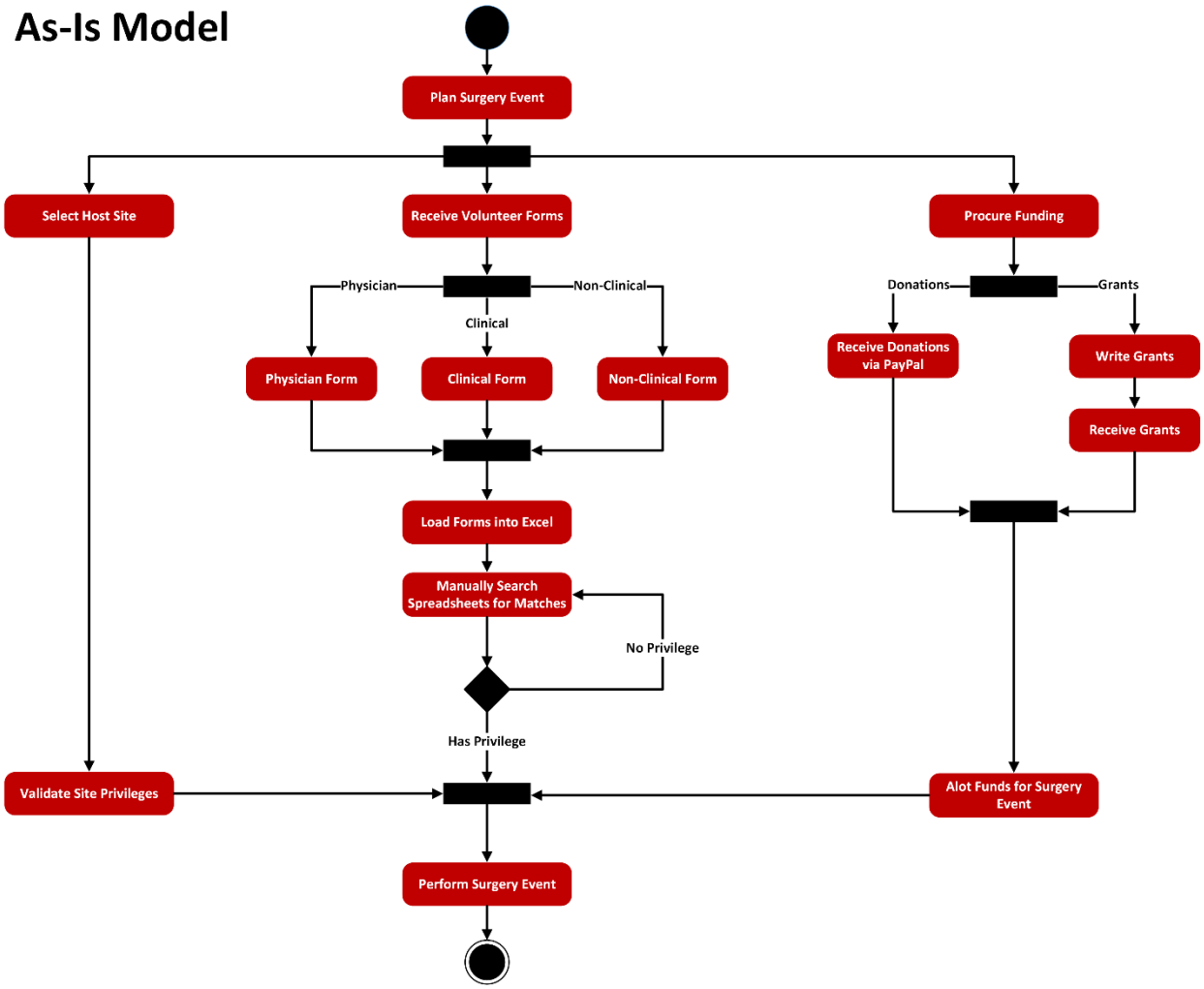
A Gantt chart marks the start and end dates for each milestone of the project. It allows a team to measure their progress towards each milestone and helps keep a project on track for timely delivery. It shows a clear order in which the project must be completed and which tasks can be worked on concurrently.


18-Sep
23-Sep
28-Sep
3-Oct
8-Oct
13-Oct
18-Oct
23-Oct
28-Oct
2-Nov
7-Nov
Task 01: Gantt Chart (Iteration II)
Task 02: Vision Document with completed faesability...
Task 03:System Requirements
Task 04: List of Use Cases
Task 05: Risk Analysis
Task 06: Initial Architecture considerations
Task 07: Inception Phase prototype
Task 08: Gantt Chart (Iteration III)
Task 09: Use Cases
Task 10: Use Case Diagram
Task 11: Use Case Prototype Version I
18-Sep
1
19-Sep
5
25-Sep
2
25-Sep
2
27-Sep
4
28-Sep
2
28-Sep
4
2-Oct
2
4-Oct
4
8-Oct
10
18-Oct
13

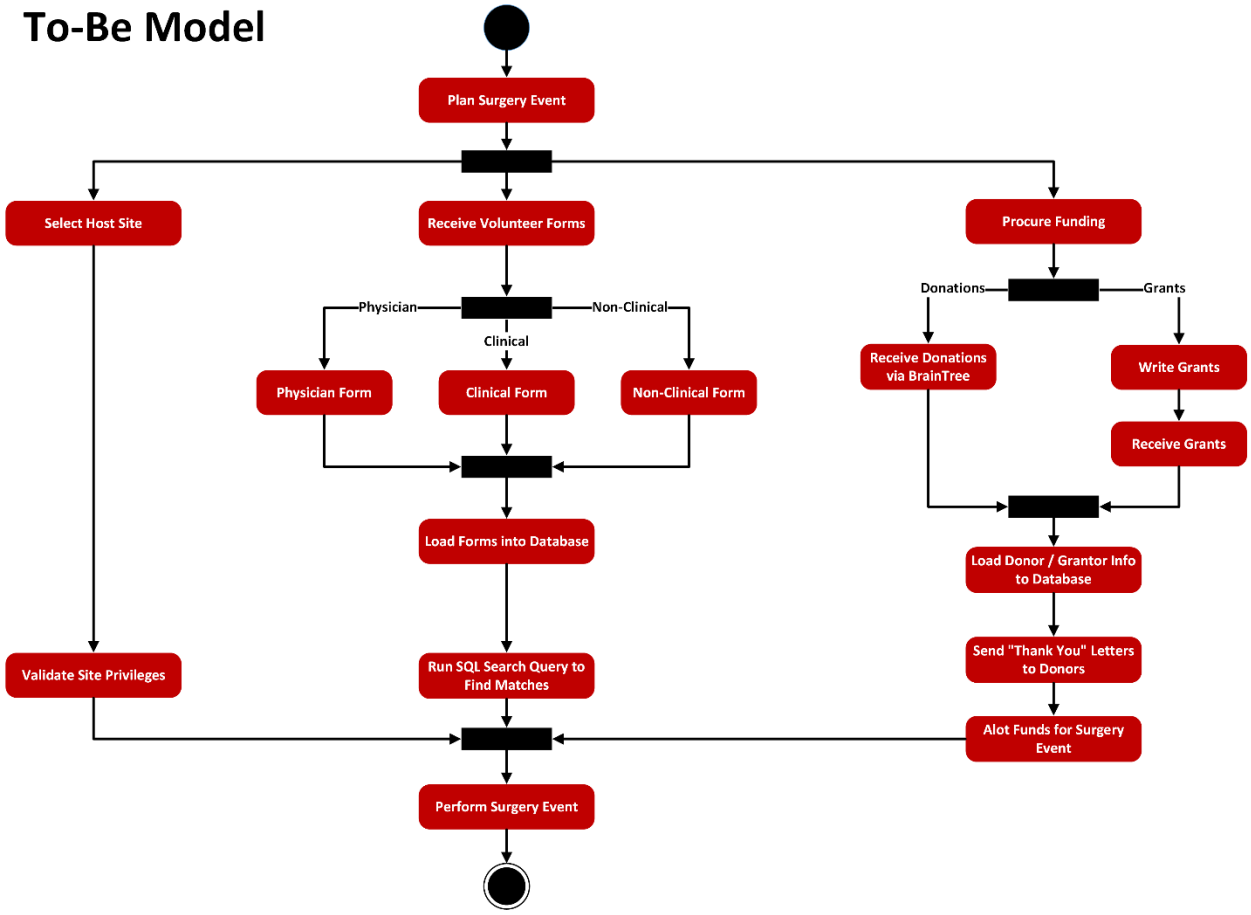
2.4 Process Models

Process models outline the process as it currently is and how it will be after the project is completed. They are a valuable reference that allow you to analyze the process and find areas for improvement. The process models provide a way to visualize the way the system will work and easily distinguish between the way the process works today and how it will be different at the completion of the project.

As-Is Process Models: Three main processes have been identified in the creation of a surgery event. Firstly, when an event is planned a location is selected and its site privileges are determined. Secondly, volunteers are added and sorted and selected. The volunteers are selected and sorted by hand through an excel spreadsheet, and if they do not have privilege to work at the host location, then another volunteer must be selected. Lastly, funds via donations or grants or procured to fund the event. When all three processes are completed, a surgery event is created. The current process is tedious, time-consuming, and has room for improvement in organization.



To-Be Process Models: The three primary processes in the creation of a surgery event will not be replaced, but rather refined for increased efficiency. A host location and its sites privileges will still be determined. The volunteer information (including location privileges) received and collected will be sorted and placed into a database. Volunteers will be selected via an SQL search query to quickly determine the participating and eligible volunteer for an event. The last process, the procurement of funds, will change in that all donor and grantor information will also be stored in the database to send out newsletters and thank-you letters to encourage additional donations. All three processes will join in the end for the creation of a surgery event. The new process will save a great deal of time by eliminating tedious processes, improve organization, and increase funds.



2.5 Use Case

A use case provides detail as to how a feature will operate. It lists the flow of events from start to finish and lists any special requirements. It clearly states the preconditions required for the use case to be initiated and the post conditions that will occur at the completion of the use case.

Use Case Specification: Modify Attorney

1. Use-Case Name: Modify Attorney

1.1 Brief Description

Attorney volunteer information is stored in the Surgery on Sunday Louisville database. The information will need to be edited and revised as needed by an SOSL admin. The use case will detail the process so that information on the database management system will reflect the changes made to the attorney volunteer.

2. Flow of Events

2.1 Basic Flow

- An SOSL admin will login to the DBMS.
- The DBMS will authenticate the admin.
- The SOSL admin will modify the attorney First Name.
- The SOSL admin will modify the attorney Last Name.
- The SOSL admin will modify the attorney Date of Birth.
- The SOSL admin will modify the attorney Address.
- The SOSL admin will modify the attorney State.
- The SOSL admin will modify the attorney City.
- The SOSL admin will modify the attorney Zip Code.
- The SOSL admin will modify the attorney Phone Number.
- The SOSL admin will modify the attorney Email.
- The SOSL admin will modify the attorney Driver's License Number.
- The SOSL admin will modify the attorney Immunization Records upload.
- The SOSL admin will modify the attorney TB Test Verification upload.
- The SOSL admin will modify the attorney Professional Title.
- The SOSL admin will modify the attorney Employer.
- The SOSL admin will modify the attorney Volunteer Type.
- The SOSL admin will modify the attorney Interest Areas.
- The SOSL admin will save the modifications of the attorney to the DBMS.
- The DBMS will validate the values of the modifications.

3. Special Requirements

3.1 Special Requirement One

- The information about the attorney needs to be present in the database.

4. Pre-conditions

4.1 Pre-condition One

- The SOSL admin needs access to login to the DBMS to administer changes.

4.2 Pre-condition Two

- The attorney needs to exist in the database.

5. Post-conditions

5.1 Post-condition One (Needed)

- The SOSL admin needs to verify that the modifications to the attorney information is correct.

5.2 Post-condition Two

- The use case is complete; attorney information is modified in the DBMS.

2.6 Agrico Case Study

The Agrico Case Study demonstrates my ability to think critically to analyze a business. It shows that I am able to communicate effectively and professionally. This case specifically touches on the issue of ethics in the information technology field. The Agrico Case Study allows me to show that I have a strong understanding of ethics in a business environment.

Zach Smith

Agrico

Case Study 5

CIS 410-02

Agrico, Inc. is a provider of farm and ranch management services. Founded in 1949 by two farmers in Des Moines, Iowa, Agrico provided management services for 691,000 acres of land across several midwestern states. They had three different arrangements for their properties, crop-share lease arrangements, cash-rent leases, and directly managed. The majority of Agrico's properties were either crop-share or cash-rent arrangements, only 2% of their properties were directly managed. By 1987 the market value of their portfolio had reached $500 million, making them one of the larger agricultural management firms in the nation.

During their 1985 planning process Agrico decided that their existing computer services were not sufficient for their present and future needs. It was decided that they would search for a new software package to support their business. The functional requirements for the system were very complex because one software package needed to work for all three property arrangements. Agrico insisted that the requirements be met by a single vendor offering an integrated package. A vendor called AMR with 12 clients already up and running was selected to provide the software. Some modifications to the software were needed, but AMR's software package would be able to meet all of Agrico's needs.

An agreement was reached with AMR to provide their software package to Agrico. AMR was very protective of the software's source code, worried that someone would steal a copy. The software purchase agreement required that AMR maintained the software in escrow with a third party to insure adequate backups. This portion of the agreement was ambiguous which caused problems between AMR and Agrico. As the project proceeded Agrico found that each instance of AMR's software was unique, making the source code for the software crucial. Agrico was not satisfied with AMR's escrow process, and found their system to be buggy, causing a strain on the relationship. Agrico had sunk a lot of money into the project, and with no other alternatives, they stuck with AMR.

The problem faced by Agrico is ethical in nature. An AMR software engineer working onsite for Agrico left the source code for the software package on an Agrico computer when she left for dinner. Should they copy the code and store it as a backup?

Agrico's mission statement is to provide agricultural management services for farmers and ranchers through cost leadership. Their organizational structure is functional. The generic strategy used by Agrico is cost leadership as they aim to obtain an extensive distribution through regional offices (Tanwar 12).

An analysis of Porter's five forces reveals the following:

Competitive Rivalry: High

There are other agricultural management providers, some even larger than Agrico. Agrico must compete on price in order to remain competitive. When there is little differentiation between your product and the product of your competitors, competitive rivalry is high (Team FME 15)

Threat of New Entrants: Low

Agrico is well established, and it would require a significant investment to enter the market. Agrico has large land holdings that they manage or lease. The industry requires a significant investment in both land and technology. The assets are also specific to the industry and have little use elsewhere leading to a low threat of new entrants (Team FME 19)

Threat of Substitutes: Low

There are few substitutes outside the industry that Agrico must worry about leading to low threat of substitutes (Team FME 20). It is possible however that software companies could market agricultural management software directly to farmers.

Bargaining Power of Suppliers: Low

Agrico provides management services and is not heavily reliant on raw materials from suppliers leading to a low bargaining power (Team FME 23). The main supplier they are reliant on is the supplier of their new system, AMR.

Bargaining Power of Customers: Low

Many of Agrico's customers lease the land they farm from Agrico. These agreements are not short term and moving to farm elsewhere would have significant costs associated with it. When switching costs are high customers have low bargaining power (Team FME 25).

Stakeholders

Agrico's Employees – All hourly and salaried employees of Agrico.

Agrico's Customers – The farmers and ranchers who use Agrico's management services.

Agrico's Shareholders – any person or organization who has invested in Agrico.

AMR's Employees – All hourly and salaried employees of AMR

AMR's Shareholders - any person or organization who has invested in AMR.

Alternatives

1. **Do Nothing.**

   By choosing not to copy the source code left on the computer Agrico could save themselves a lot of trouble down the road. While having the code would be good for Agrico, it would be a violation of their contract with AMR to copy it. They would have to continue to work with AMR to find an escrow solution that was satisfactory to both parties which could be extremely difficult. Conflict will always be present in organizations and Agrico must find a way to work with AMR (Morgan 163). Agrico does not have many options other than working with AMR so it would be in their best interest to avoid damaging the relationship. They are reliant on AMR to provide the software package in order to support their business functions today and into the future. The downside to this option is that Agrico cannot be sure that the code AMR places in escrow is the source code that generated the object code they were provided. If the source code for Agrico's software package was lost it would have a significant impact on Agrico's ability to make sure the software can grow with them.

2. **Copy the Source Code.**

   By copying the source code Agrico could completely alleviate one of their major concerns with AMR. They would have a copy of the source code stored safely to their standards available to them in case any modifications needed to be made in the future or if AMR was unable to support them for any reason. However, if they were caught copying the code it could lead to a legal battle with AMR. Not only would a legal battle with AMR be costly, it would likely bring

negative attention to Agrico which could potentially hurt business in the future. It is possible that Agrico may lose the ability to use the software package provided by AMR if they are found to be in breach of contract. Agrico had sunk a lot of money into testing the AMR software, and only one other vendor provided a similar software package, but it had never been put into production, and had only sold a few copies. Changing directions at this point would be extremely expensive for Agrico, and it is impossible to predict the cost of a legal battle coupled with negative publicity for the company.

Recommended Alternative

I would recommend that Agrico choose to do nothing and not copy the source code. While losing the source code for any reason would be costly for Agrico, it pales in comparison to the cost they could face in a legal battle with AMR if they lose. They may lose access to the software package provided by AMR and be forced to start back from square one. It was not advised that they design their own system and only one other vendor can provide a similar software package. Being forced to start the process of finding a new software package now would be extremely expensive and Agrico has already sunk money into testing the AMR system. If they are caught copying the code it would damage the company's reputation and it may make it harder to find a software provider willing to work with them. It may also affect customer's perception of Agrico and hurt business. Choosing not to copy the code is the ethical decision in this case.

Works Cited

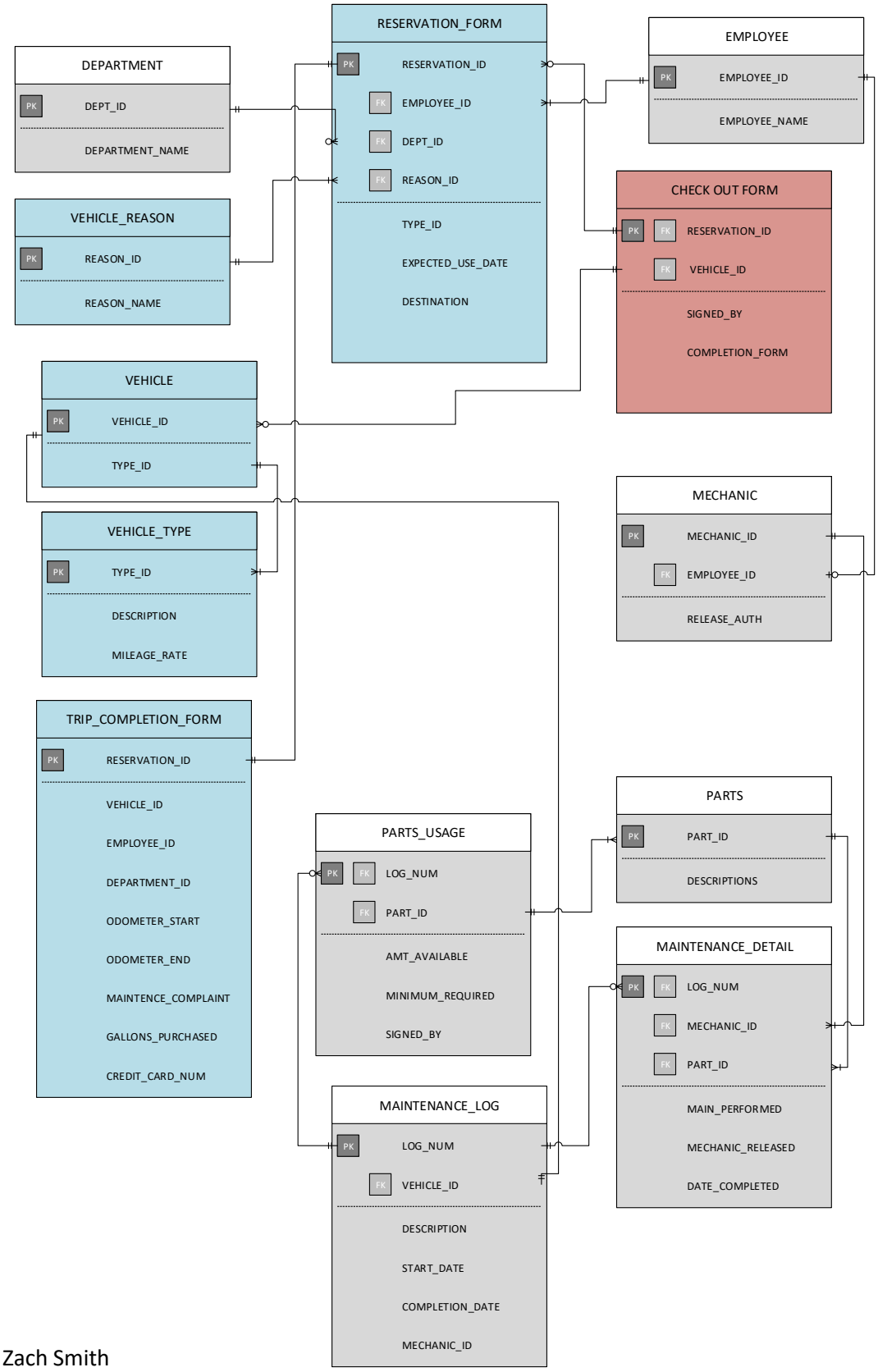
Tanwar, Rikita. “Porter’s Generic Competitive Strategies.” IOSR Journal of Business and Management Volume 15, Issue 1 (Nov. – Dec. 2013)

Team FME “Porters Five Forces: Strategy Skills” www.freemanagementebooks.com

Morgan, Gareth. Images of Organization. Updated Edition, Sage Publications, 2006.

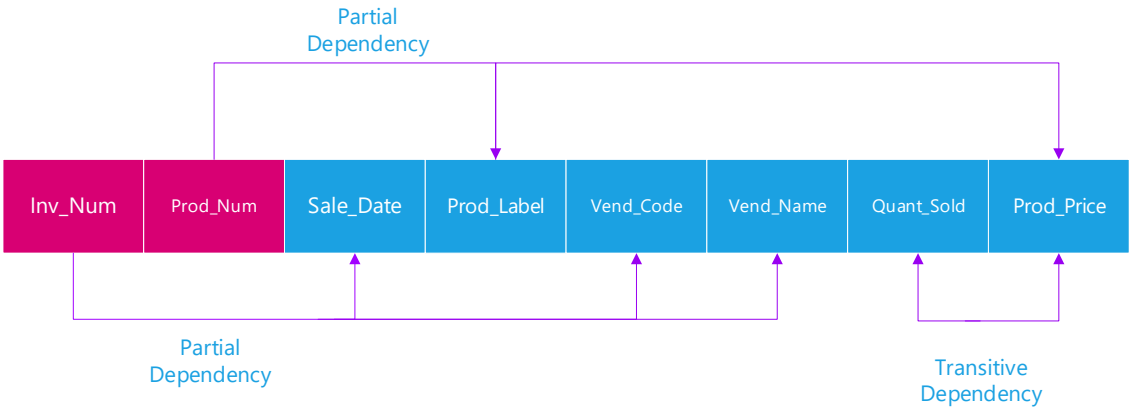
3.1 Entity Relationship Diagram

An entity relationship diagram (ERD) is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts, or events within that system. They provide a visual starting point for a database and can be used to help determine information system requirements. The ERD is a crucial part of building a relational database.


RESERVATION_FORM
RESERVATION_ID
EMPLOYEE_ID
DEPT_ID
REASON_ID
TYPE_ID
EXPECTED_USE_DATE
DESTINATION
EMPLOYEE
EMPLOYEE_ID
EMPLOYEE_NAME
DEPARTMENT
DEPT_ID
DEPARTMENT_NAME
VEHICLE_REASON
REASON_ID
REASON_NAME
CHECK OUT FORM
RESERVATION_ID
VEHICLE_ID
SIGNED_BY
COMPLETION_FORM
VEHICLE
VEHICLE_ID
TYPE_ID
MECHANIC
MECHANIC_ID
EMPLOYEE_ID
RELEASE_AUTH
VEHICLE_TYPE
TYPE_ID
DESCRIPTION
MILEAGE_RATE
TRIP_COMPLETION_FORM
RESERVATION_ID
VEHICLE_ID
EMPLOYEE_ID
DEPARTMENT_ID
ODOMETER_START
ODOMETER_END
MAINTENCE_COMPLAINT
GALLONS_PURCHASED
CREDIT_CARD_NUM
PARTS
PART_ID
DESCRIPTIONS
PARTS_USAGE
LOG_NUM
PART_ID
AMT_AVAILABLE
MINIMUM_REQUIRED
SIGNED_BY
MAINTENANCE_DETAIL
LOG_NUM
MECHANIC_ID
PART_ID
MAIN_PERFORMED
MECHANIC_RELEASED
DATE_COMPLETED
MAINTENANCE_LOG
LOG_NUM
VEHICLE_ID
DESCRIPTION
START_DATE
COMPLETION_DATE
MECHANIC_ID

3.2 Normalization

Normalization is the restructuring of a relational database through a series of normal forms in order to reduce data redundancy and improve data integrity. Normalization organizes a database to ensure that dependencies are properly enforced by database integrity constraints. The process of normalization helps prevent data anomalies in the database.



1NF: (INV_NUM, PROD_NUM, SALE_DATE, PROD_LABEL, VEND_CODE, VEND_NAME, QUANT_SOLD, PROD_PRICE)

Transitive Dependency:
QUANT_SOLD → PROD_PRICE

Partial Dependencies:
PROD_NUM → PROD_LABEL, PROD_PRICE
INV_NUM → SALE_DATE, VEND_CODE, VEND_NAME

2NF:

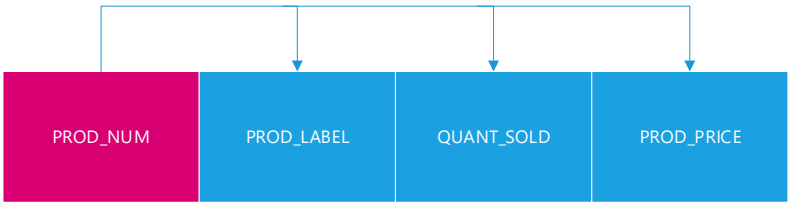


Table: PRODUCTS

PRODUCTS (PROD_NUM, PROD_LABEL, QUANT_SOLD, PROD_PRICE)

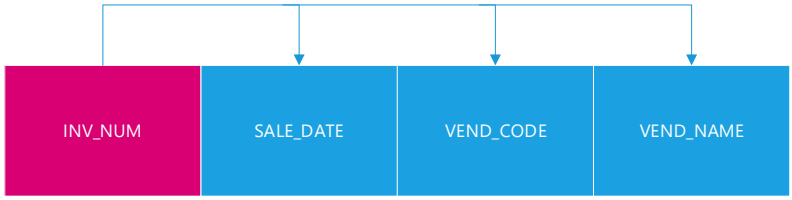


Table: INVOICES

INVOICES (INV_NUM, SALE_DATE, VEND_CODE, VEND_NAME)

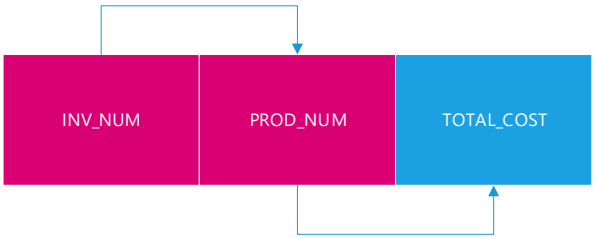


Table: SALES

SALES (INV_NUM, PROD_NUM, TOTAL_COST)

3NF:

Table: PRODUCTS

PRODUCTS (PROD_NUM, PROD_LABEL, QUANT_SOLD)

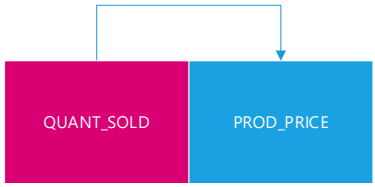


Table: QUANTITY

QUANTITY (QUANT_SOLD, PROD_PRICE)

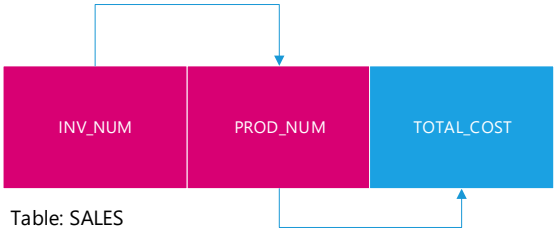


Table: SALES

SALES (INV_NUM, PROD_NUM, TOTAL_COST)

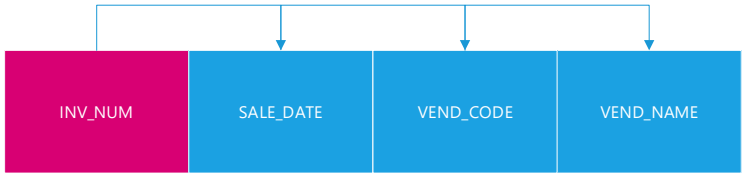


Table: INVOICES

INVOICES (INV_NUM, SALE_DATE, VEND_CODE, VEND_NAME)

3.3 Trigger

A trigger is a special type of stored procedure that executes automatically when a certain event occurs in a database server. The trigger I have provided as an example executes when the database for a movie rental business is updated. When the due or return date of a movie is updated, the trigger is executed and calculates the amount of money owed by a customer.

```
Create TRIGGER A8

ON DETAILRENTAL

AFTER UPDATE

AS

BEGIN


DECLARE @RENT_NUM CHAR

DECLARE @PRIOR_DAYS_LATE INT

DECLARE @DAYS_LATE INT

DECLARE @DAILY_LATE_FEE DECIMAL (5,2)

DECLARE @PRIOR_UPDATE DECIMAL (5,2)

DECLARE @AFTER_UPDATE DECIMAL (5,2)

DECLARE @UPDATED_LATE_FEE DECIMAL (5,2)


IF(EXISTS (SELECT * FROM DELETED) AND EXISTS (SELECT * FROM INSERTED)) BEGIN


DECLARE UPDATE_CURSOR CURSOR

FOR SELECT I.RENT_NUM, DATEDIFF(DAY, D.DETAIL_DUEDATE, D.DETAIL_RETURNDATE) AS
PRIOR_DAYS_LATE,

DATEDIFF(DAY, I.DETAIL_DUEDATE, I.DETAIL_RETURNDATE) AS DAYS_LATE,
I.DETAIL_DAILYLATEFEE

FROM INSERTED I INNER JOIN DELETED D ON I.RENT_NUM = D.RENT_NUM AND I.VID_NUM =
D.VID_NUM


OPEN UPDATE_CURSOR

FETCH NEXT FROM UPDATE_CURSOR
```

```
INTO @RENT_NUM, @PRIOR_DAYS_LATE, @DAYS_LATE, @DAILY_LATE_FEE
WHILE (@@FETCH_STATUS = 0)

IF @DAYS_LATE IS NULL
BEGIN
SET @AFTER_UPDATE = 0
END
ELSE
BEGIN
SET @AFTER_UPDATE = (@DAYS_LATE * @DAILY_LATE_FEE)
END

 IF  ((@AFTER_UPDATE - @PRIOR_UPDATE) <> 0)
 BEGIN
 SET @UPDATED_LATE_FEE = (@AFTER_UPDATE - @PRIOR_UPDATE)
 END
 ELSE
 BEGIN
 SET @UPDATED_LATE_FEE = 0
 END

 BEGIN
 UPDATE MEMBERSHIP
 SET  MEM_BALANCE = MEM_BALANCE + @UPDATED_LATE_FEE
 WHERE MEM_NUM = (SELECT MEM_NUM FROM RENTAL WHERE RENT_NUM = @RENT_NUM)
 END
```

```
 FETCH NEXT
 FROM UPDATE_CURSOR
 INTO @RENT_NUM, @DAYS_LATE, @DAILY_LATE_FEE
 END
 CLOSE UPDATE_CURSOR
 DEALLOCATE UPDATE_CURSOR

 END



--TEST CODE

--TRANSACTIONS FOR TESTING
SELECT *
FROM DETAILRENTAL
WHERE RENT_NUM = '1006' OR RENT_NUM = '1007'

--SETTING RETURN DATE
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE = '3/9/2013'
WHERE RENT_NUM = '1006' AND VID_NUM = '61367'

UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE = '3/9/2013'
WHERE RENT_NUM = '1007'
```

```
--CHECK TO SEE THAT BALANCE CHANGED ON UPDATE
SELECT RENT_NUM, MEM_BALANCE
FROM MEMBERSHIP M JOIN RENTAL R ON M.MEM_NUM = R.MEM_NUM
WHERE RENT_NUM = '1006' OR RENT_NUM = '1007'

--UPDATE RETURN DATE BACK TO ORIGINAL VALUE OF NULL
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE = NULL
WHERE RENT_NUM = '1007'

UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE = NULL
WHERE RENT_NUM = '1006' AND VID_NUM = '61367'
```

4.1.1 Console Application

A console application runs directly in a text-based interface and does not require the use of a mouse, unlike a graphical user interface application. This application demonstrates the use of nested loops to produce 4 sets of patterns made up of asterisks which are displayed in the console.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab6
{
    class Program
    {
        static void Main(string[] args)
        {
            const int MAX_ROWS = 10;

            Console.Write("Pattern A");
            Console.WriteLine();
            Console.WriteLine();

            for (int row = 1; row <= MAX_ROWS; row++)
            {
                for (int star = 1; star <= row; star++)
                    Console.Write("*");
                Console.WriteLine();
            }

            Console.WriteLine();
            Console.Write("Pattern B");
            Console.WriteLine();
            Console.WriteLine();

            for (int row = 10; row <= MAX_ROWS && row > 0; row--)
            {
                for (int star = 1; star <= row; star++)
                    Console.Write("*");
                Console.WriteLine();
            }

            Console.WriteLine();
            Console.Write("Pattern C");
            Console.WriteLine();
            Console.WriteLine();

            for (int row = 10; row <= MAX_ROWS && row > 0; row--)
            {
                for (int space = 0; space <= MAX_ROWS - row; space++)
                    Console.Write(" ");
                for (int star = 1; star <= row; star++)
                    Console.Write("*");
                Console.WriteLine();
            }

            Console.WriteLine();
            Console.Write("Pattern D");
            Console.WriteLine();
            Console.WriteLine();
```

```
            for (int row = 1; row <= MAX_ROWS; row++)
            {
                for (int space = 1; space <= (MAX_ROWS - row); space++)
                    Console.Write(" ");
                for (int star = 1; star <= row; star++)
                    Console.Write("*");
                Console.WriteLine();
            }
        }
    }
}
```

4.1.2 Graphical User Interface Application

A graphical user interface application allows a user to interact with the application through icons and visual indicators, often using a mouse in addition to a keyboard. This application allows you to input the square footage of the area to be painted, the number of coats you would like to apply, and the price per gallon of the paint to be used. It will use your inputs to calculate the total square feet to be painted, gallons of paint required, hours of labor required, paint cost, labor cost, and total cost and display the results.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Program1
{
    public partial class Form1 : Form
    {
        //Declare constants
        private const float SQ_FT_PER_GALLON = 330; //total square feet that can be
painted with one gallon of paint
        private const int LABOR_PER_GALLON = 6; //the number of hours of labor required
to paint one gallon of paint
        private const float LABOR_COST_PER_HOUR = 10.5F; //the cost of labor per hour

        public Form1()
        {
            InitializeComponent();
        }

        //calculates total square feet to be painted, gallons required, hours required,
paint and labor cost, and total cost
        //displays totals in the appropriate output label
        private void calculateButton_Click(object sender, EventArgs e)
        {
            float sqFtPainted; //holds square feet to be painted
            int coats; //holds number of coats to be painted
            double price; //holds the paint price per gallon
            float totalSqFt; //holds the total square feet to be painted
            double gallonsRequired; //holds the number of gallons required
            float hours; //holds the number of hours of labor required
            double paintCost; //holds the cost of paint
            float laborCost; //holds the cost of labor
            double totalCost; //holds the total cost

            sqFtPainted = float.Parse(sqFtPaintedTextBox.Text); //gets number of square
feet to be painted from text box

            coats = int.Parse(coatsTextBox.Text); //gets number of coats from text box

            price = float.Parse(priceTextBox.Text); //gets price of paint per gallon from
text box

            totalSqFt = coats * sqFtPainted; //calculates total square feet to be painted

            totalSqFtOutputLabel.Text = totalSqFt.ToString("F1"); //displays the the
number of total square feet to be painted in the output label rounded to one decimal
```

```
            gallonsRequired = Math.Ceiling(totalSqFt / SQ_FT_PER_GALLON); //calculates
gallons required to the closest whole gallon

            gallonsOutputLabel.Text = gallonsRequired.ToString("N0"); //displays the
number of gallons required in the output text box

            hours = (totalSqFt / SQ_FT_PER_GALLON) * LABOR_PER_GALLON; //calculates the
number hour labor hours required

            hoursOutputLabel.Text = hours.ToString("N1"); //displays the hours required
in the output text box

            paintCost = gallonsRequired * price; //calculates the cost of paint

            paintCostOutputLabel.Text = paintCost.ToString("C"); //displays the cost of
paint in the output text box in currency format

            laborCost = hours * LABOR_COST_PER_HOUR; //calculates labor cost

            laborCostOutputLabel.Text = laborCost.ToString("C"); //displays labor cost in
the output box in currency format

            totalCost = laborCost + paintCost; //calculates total cost

            totalOutputLabel.Text = totalCost.ToString("C");

        }

        //clears all input text boxes and output labels
        private void clearButton_Click(object sender, EventArgs e)
        {
            sqFtPaintedTextBox.Text = "";

            coatsTextBox.Text = "";

            priceTextBox.Text = "";

            totalSqFtOutputLabel.Text = "";

            gallonsOutputLabel.Text = "";

            hoursOutputLabel.Text = "";

            paintCostOutputLabel.Text = "";

            laborCostOutputLabel.Text = "";

            totalOutputLabel.Text = "";

        }
    }
}
```

3.2 CSHTML

The following example code for a form I wrote using CSHTML. This is a general volunteer form written for the website of a non-profit organization. It allows a volunteer to enter various pieces of information that will ultimately be entered into the relational database of the non-profit organization.

```
@{
    ViewBag.Title = "GeneralForm";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<br />
<h2 style="color:crimson; padding-left:5vw; padding-right:5vw">Non-Clinical General
Volunteer Form</h2>
<div style="padding-left:5vw">
<span class='right-header-link'>
    <li><a href="/user/sign_in_or_sign_up">Are you a Donor? Sign In</a></li>
    <li><a href="/user/sign_in_or_sign_up">Create An Account</a></li>
</span>
    </div>
<br />

<div link rel="stylesheet" type="text/css" href="~/Custom_CSS_Styles/inputboxes.css"
style="padding-left:5vw; padding-right:5vw">
    <label>First Name</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <div id="right" style="float:right; width:50%; position:fixed; top:10vw; right:5vw;
box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0,
0.19);padding:1vw">
        This volunteer form is for all individuals who wish to volunteer with SOSL in a
non-clinical fashion, or for undergraduate and medical students who wish to shadow our
clinical operations.
    </div>

    <label>Last Name</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
    </div>


    <label>Address</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <label>City</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <label>State</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
```

```
    </div>

    <label>Postal Code</label>
    <div>
        <div><input required maxlength="12" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <label>Mobile Phone</label>
    <div>
        <div><input required maxlength="15" aria-required="true" size="1000" type="tel"
/></div>
    </div>

    <label>Email</label>
    <div>
        <div><input required maxlength="100" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <label>Birthday</label>
    <div>
        <div><input required maxlength="50" aria-required="true" size="1000" type="text"
/></div>
    </div>

    <label>Preferred Name</label>
    <div>
        <div><input required maxlength="50" aria-required="false" size="1000" type="text"
/></div>
    </div>

    <label>Employer</label>
    <div>
        <div><input required maxlength="50" aria-required="false" size="1000" type="text"
/></div>
    </div>

    <label>Which Best Describes You?</label>
    <div>
        <select>
            <option>Undergraduate Student</option>
            <option>Medical Student (1st-3rd Year)</option>
            <option>Medical Student (4th Year)</option>
            <option>Other</option>
        </select>
    </div>

    <label>Which areas are you interested in volunteering in?</label>
    <div style="height:5vw;width:10vw;border:1px solid #ccc;overflow:auto;">
        <div class="checkbox"><label><input name="interestArea[]" type="checkbox"
value="Registration">Registration</label></div>
        <div class="checkbox"><label><input name="interestArea[]" type="checkbox"
value="Cleaning">Cleaning</label></div>
```

```
        <div class="checkbox"><label><input name="interestArea[]" type="checkbox"
value="Clinical Shadowing">Clinical Shadowing</label></div>
        <div class="checkbox"><label><input name="interestArea[]" type="checkbox"
value="Fundraising">Fundraising</label></div>
        </div>

    <label>Briefly Explain to us why you are interested in serving with SOSL.</label>
    <div>
        <textarea style="resize:none" rows="4" cols="50"></textarea>
    </div>

    <label>What is your T-Shirt Size?</label>
    <div>
        <select>
            <option>XS</option>
            <option>S</option>
            <option>M</option>
            <option>L</option>
            <option>XL</option>
            <option>2XL</option>
        </select>
    </div>
    <br>
    </div>

    <div style="padding-left:5vw; padding-right:5vw"><input type="submit" name="commit"
value="Submit" class="btn btn btn-dark btn-lg" data-disable-with="Saving..." /></div>
```