

Spring 4-1-2016

Student Queue and Tracking System

Don R. Davis
Dakota State University

Follow this and additional works at: <https://scholar.dsu.edu/theses>

Recommended Citation

Davis, Don R., "Student Queue and Tracking System" (2016). *Masters Theses*. 221.
<https://scholar.dsu.edu/theses/221>

This Thesis is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.



STUDENT QUEUE AND TRACKING SYSTEM

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

April, 2016

By

Don R Davis

Project Committee:

Dr. Insu Park

Dr. Shuyuan Deng

Dr. Jun Liu




PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Don Davis

Master's Project Title: Student Queue and Tracking System

Signature Approval:

Faculty supervisor:  Date: 04/28/2016

Committee member:  Date: 4/8/06

Committee member:  Shuyuan Deng Date: 04/28/2016

ACKNOWLEDGMENT

I wish to acknowledge and thank the people that work in the Student Services department at Daytona State College for the assistance they were able to provide during my work on this project. They were extremely busy with our PeopleSoft implementation but still managed to devote a little time to help me.

I would also like to thank my wife Anna and my daughter Kristy for enduring the past few years as I worked fulltime and spent many hours each week in pursuit of my MSIS degree. Without their understanding and overwhelming support I could not have been successful in this quest.

Most of all I want to thank my Lord and Savior Jesus Christ for giving me the strength and ability to achieve this education.

ABSTRACT

No one wants to wait for anything, especially anxious college students trying to get assistance from college personnel. Waiting is a fact of life, though, on a college campus. The key is to orchestrate the service so customer waiting is minimum, and people are served methodically and quickly. The chosen solution was to incorporate a queuing system to track when students and non-students arrive for services and find out what services they need assistance with.

Since the college's Student Services department does not want to set appointments, it was decided that a web application which can be accessed anywhere, so the student can get in line without actually being on campus, was to be developed. Five offices within the Student Services department were consulted as to their desired functionality for such a system. The basic requirements of logging the name, the reason for the visit, and the time of login were outlined and agreed upon by each office. A couple offices had additional items they wanted to include.

As always, the application programming was a great deal of fun. This time especially since it required learning a new Integrated Development tool, Microsoft Visual Studio. The .Net environment is quite different than the Borland C++ Builder IDE, which the tool of choice was during the early years of the 21st century at Daytona State College. The development team has switched over in recent years to .Net for portal applications, hence the decision to use Visual Studio for this project.

Pinning the functional personnel down for testing proved more difficult than originally expected due to a new Enterprise Resource Planning system implementation. DSC's

PeopleSoft implementation activities required much more time and energy than anticipated and the functional areas simply could not spare the resources for testing the Student Queue and Tracking system. This slowed the project some but it did not derail it entirely.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

A handwritten signature in black ink, appearing to read "Don R Davis", written over a horizontal line.

Don R Davis

TABLE OF CONTENTS

STUDENT QUEUE AND TRACKING SYSTEM.....	I
PROJECT APPROVAL FORM.....	II
ACKNOWLEDGMENT	III
ABSTRACT	IV
DECLARATION.....	VI
TABLE OF CONTENTS.....	VII
LIST OF TABLES	X
LIST OF FIGURES	XI
CHAPTER 1	1
INTRODUCTION.....	1
BACKGROUND OF THE PROBLEM	1
STATEMENT OF THE PROBLEM	1
OBJECTIVES OF THE PROJECT	2
Proposed High-Level Scope of Work Statement.....	2
Project Justification.....	3
Assumptions & Constraints	3
Major Risks.....	3
Operational Considerations.....	4
Project Deliverables	4
Internal Priority.....	4
PROJECT NEED.....	5
What.....	5
How	5
Why	6
Success Matrix.....	6
CHAPTER 2	7
LITERATURE REVIEW.....	7

CHAPTER 3	9
SYSTEM DESIGN (RESEARCH METHODOLOGY).....	9
REQUIREMENT ANALYSIS.....	10
Requirements	10
SYSTEM DESIGN	12
IMPLEMENTATION (DEVELOPMENT)	17
CHAPTER 4	23
RESULTS AND DISCUSSION	23
CHAPTER 5	27
CONCLUSIONS	27
REFERENCES.....	29
APPENDICES	31
APPENDIX A: USERS' MANUAL.....	31
STUDENT QUEUE LOGIN	31
STAFF LOGIN	34
STAFF SESSION	34
DEPARTMENT REASONS	38
STAFF DEPARTMENTS.....	41
APPENDIX B: SYSTEM TECHNICAL DOCUMENTATION	45
DATABASE DDL.....	46
SOURCE CODE	50
QueueLogin.aspx	50
QueueLogin.aspx.cs.....	53
StaffLogin.aspx.....	63
StaffLogin.aspx.cs	64
StaffSession.aspx	67
StaffSession.aspx.cs.....	70
StaffDepartments.aspx	77
StaffDepartments.aspx.cs.....	80
DepartmentReasons.aspx	85
DepartmentReasons.aspx.cs.....	87
Web.config.....	92

LIST OF TABLES

Table 1. Project tables	13
Table 2. person table	14
Table 3. staff table.....	14
Table 4. department table.....	15
Table 5. dept_reason table.....	16
Table 6. staff_dept table.....	16
Table 7. session table	17
Table 8. Hours for requirements gathering, analysis, and initial project documents...	23
Table 9. Hours during development of software.....	24
Table 10. Additional requested functionality.....	25

LIST OF FIGURES

Figure 1. Waterfall Model of system design.....	9
Figure 2. Student Queue and Tracking data flow diagram	12
Figure 3. Queue Login page.....	18
Figure 4. Staff Login page	19
Figure 5. Staff Session page.....	20
Figure 6. Department Reasons page.....	21
Figure 7. Staff Departments page.....	22
Figure 8. Entity Relationship Diagram	45

CHAPTER 1

INTRODUCTION

Background of the Problem

Waiting one's turn in line for help is not a unique experience of Daytona State College students. It is, however, a problem that DSC desires to resolve, and is considering committing resources to do so.

The offices of the Student Services department at Daytona State College take care of several thousand students each major semester. Of those thousands, more than seventy-five percent are assisted in-person, while on campus. Historically, the department has utilized a sign-up sheet in the lobby of each office to track students and attempt to aid them in-turn.

While functional, this system has proven to not be the most efficient solution for students. When students sign-in, part of the process is to accurately note the time of their arrival in hopes of delivering services on a first-come, first-served basis. That is often not the case, however, as some students attempt to circumvent the process by noting a time earlier than their actual time. Frequent arguments ensue regarding who should be next, sometimes escalating to physical violence.

Statement of the problem

Daytona State College needs a better system for controlling the flow of students in the Student Services department offices. The system needs to track current students as well as non-students. The system should allow each student to indicate the reason for their visit. These

reasons should be associated with the particular office (i.e. Financial Aid, Student Accounts, Admissions, Records, etc.) where assistance is needed.

The system needs to be available through the web and it should allow students to sign in whether on campus or off campus. The system should automatically apply a time-stamp when a student signs in to prevent tampering.

There is also a need for college personnel to be able to document the session with students. Staff members need to be able to make notes of discussions and recommendations made during the sessions. These notes should be available to any and all office personnel for future perusal.

Objectives of the project

Proposed High-Level Scope of Work Statement

A software package used by students to sign in for services from several offices within Student Services for the college. The package would also allow college employees of each office to monitor the queue for the next student to be seen. The information collected from the student would include, but not necessarily be limited to, the following selections:

- Department from which services are required
- Designated staff member desired, or first available
- Topic of service needed (Reason)
- Date/Time of sign in (Automatic by program)

The information collected from the employee would include, but not necessarily be limited to, the following:

- Date/Time of session start/end

- Session Notes/Recommendations

Project Justification

Student Services is the fulcrum of a student's college experience. The objective of this department is to ensure students receive the education they pursue. When a student makes contact with any of the areas in the department, it is because of a specific need for which timely attention is required. The Student Services personnel must be able to move smoothly from one student to the next and they must be able to ascertain correctly which student should be assisted next.

Assumptions & Constraints

Assumptions:

- Existing hardware and DBMS will support the new software package
- Functional personnel will thoroughly test the software and provide feedback
- Change Management will be exercised to prevent scope creep

Constraints:

- Software must be web based
- Software must be ready to go live by Summer semester 2016

Major Risks

- Illness on the part of the student, project supervisor, or Student Services personnel
- Student Services personnel not being able to devote time for testing

Operational Considerations

It is not expected that testing and implementation activities will cause great disruption in normal business activities of each department. It is expected that the new software will greatly improve the customer service provided to the students as long as college personnel are committed to follow the first-come, first-served paradigm for which the software is designed to ensure.

Project Deliverables

The deliverables of the project include:

- Data Flow chart of program
- Database to store the data associated with each session
- Entity Relationship Diagram
- Source code documents

Internal Priority

The priority for this project is a Level 2. This software package will provide much needed structure to the waiting process within the Student Services areas. The ability for students to establish their place in the queue upon arrival to the specific area will help to ensure calm interactions and smoother processing.

The need to concentrate on providing students the best possible attention and service during their entire association with the college is paramount to the college's continued success. Current students as well as future students will share their experience with friends and family. Improved student experiences will bolster the college's stance against its competing schools.

Project Need

What

A software application whereby students requiring assistance from any area within the Student Services department can sign in and establish their place “in line” among all those seeking help. The application will also have an employee component whereby staff members can monitor the queue to ensure the students receive help in turn.

The student interface will include:

- The ability for students to sign in
- The ability for students to select a specific staff member, if desired (May alter the first-come, first-served restriction for some offices)
- The ability for students to select the reason for their visit (Options may vary by area)

The staff interface will include:

- The ability for staff to monitor the queue and select the next student in line
- The ability for staff to add notes regarding the session

How

The software package will be developed as a browser based solution. Authentication will be required for the staff interface, since student data will be involved. No authentication will be required for the student interface.

The development of this solution will entail the following:

- Business needs analysis and requirements gathering
- Quality assurance testing

- C# development expertise
- SQL development expertise

Why

Student success is the heart of the college's mission. It has become evident that the frustration experienced by students when waiting to receive services from areas such as Financial Aid, Records & Registration, Assessment, Advising, and Student Accounts, not only causes disruptions in the waiting areas but may also be driving students to look to other schools.

To help ensure student success and keep them as their students, the college must focus on improving the experience in these Student Services areas. The first step in the renewed focus is the development of the queuing solution.

Success Matrix

- Fully qualified and agreed upon software requirements document
- Full commitment from Student Services personnel to thoroughly test software
- On time delivery of software package
- Full documentation of software

CHAPTER 2

LITERATURE REVIEW

Waiting is a normal part of everyone's life. It is not, however, an enjoyable part for most people. People wait for their doctor appointments, wait in line at the supermarket, get stuck in heavy traffic, and even spend extra time mailing packages at the post office. Even though the waiting is expected, it can still be unpleasant. David Maister quoted a Federal Express advertisement from a July 28, 1980 issue of Fortune Magazine stating "Waiting is frustrating, demoralizing, agonizing, aggravating, annoying, time consuming and incredibly expensive" (Maister, n.d.).

Educational institutions have become very competitive in the last decade, especially at the community college and state college levels. Providing the best possible customer service, especially for those students that come on-site, can mean the difference between losing and gaining a loyal long-term customer. "A long and unpleasant wait can damage a customer's view of a brand, cause people to leave a line or not enter it in the first place, or discourage them from coming back to the store entirely" (Swanson, 2015). Improving the perceived experience for students that are waiting in-line is a great opportunity to win one over.

One possible way to improve the waiting experience is to shorten the amount of time that students must wait to be helped. This seems like a logical approach but how long of a wait is too long? How long are the students currently waiting in each office? The only way to know for certain is to design a tracking mechanism that automatically notes the time-stamp for each student at the moment they sign in. Reports can then be generated for evaluation to determine

the average wait times for each time slice of a business day. Then, and only then, can an informed decision be made on how to attack and shorten those wait times.

The other part of the problem facing the Student Services Department at DSC is to track the interaction between the student and the college staff. Many recommendations are made that will assist the student with getting and keeping financial aid, registering for the required classes needed for the degree being sought, etc. These notes can be valuable to any staff member that will have a session with this student in the future. By tracking the time that college personnel spend with the students, Daytona State College can help each office improve efficiency as well.

CHAPTER 3

SYSTEM DESIGN (RESEARCH METHODOLOGY)

It was determined that the basic Waterfall Model of system design was well suited for this project. Finishing one part of the model before moving on to the next worked well as anticipated.

Following is a diagrammatic representation of different phases of waterfall model.

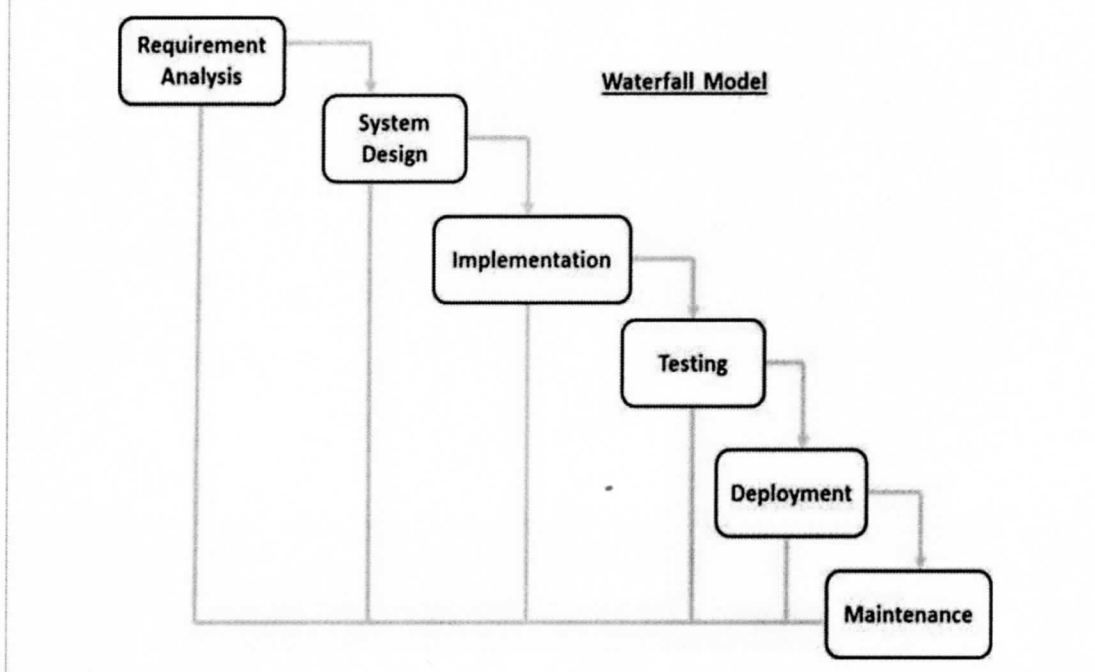


Figure 1. Waterfall Model of system design

("SDLC - Waterfall method," n.d.)

Requirement Analysis

The requirement analysis phase was carried out by interviewing the director of each of the offices within Student Services to discover what each one desired from a student queuing and tracking system. The requirements varied from simply providing a way for students to sign in, all the way to a sophisticated system for mobile phone viewing of the queue and notification of a student's spot in queue. The set of requirements agreed upon prior to the start of the project contain a fair compromise.

Requirements

The student sign-in queuing system should be developed based on the following criteria.

The system must...

- Allow students and non-students (customer) to sign-in and 'get in line' for services – Use College ID, phone #, or first & last name for lookup
- Allow customer to select a reason for the visit
- For some areas, allow customer to choose a specific staff member for the session
- Allow college staff to view the queue for their area and select the next customer in line, initiating the session
- Allow college staff to view all notes that exist for the customer regardless of which staff member entered them
- Authenticate staff upon login
- Notify area staff when new customer signs in if desired by area
- Capture the following data:
 - Customer ID if available

- Customer Name (look up from ID or phone number if available)
- Reason for visit
- Date of session
- Time of customer sign-in
- Time session initiated by staff
- Staff member for session
- Time session ended by staff
- Relevant session notes by staff

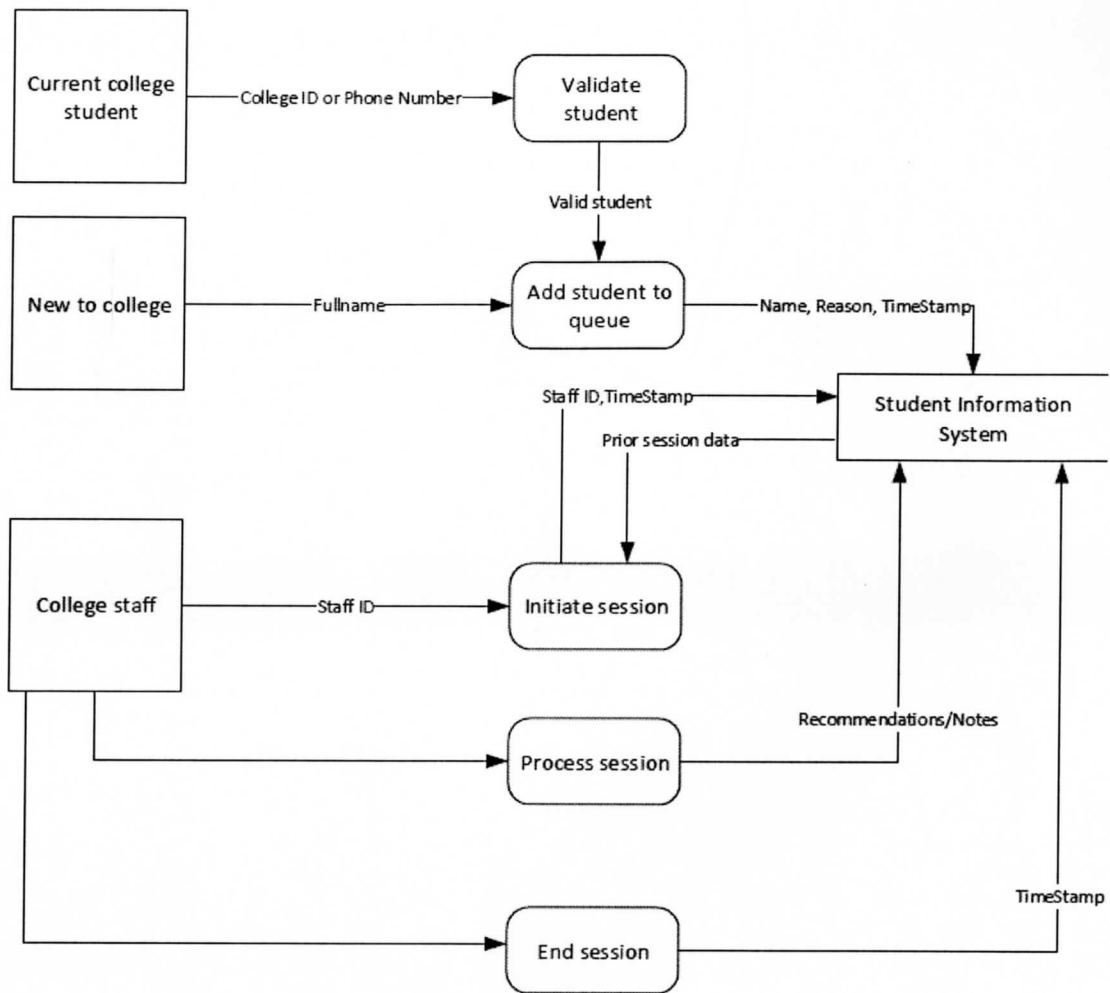


Figure 2. Student Queue and Tracking data flow diagram

System Design

The design phase of the project began after carefully scrutinizing the analysis of the requirements. Since the offices of the Student Services department desired a solution that was web-based, the first decision was to select a tool for the development. Daytona State College has used Microsoft's Visual Studio for its portal development for the last decade, and it made sense to continue using the .Net tool for this application. The problem with this decision, though, was

the fact that this MSIS student was not familiar with Visual Studio going into this project, so additional learning was required.

The next decision was which database engine to use for storing the data of the system. Daytona State College currently uses Informix for its Jenzabar ERP but is migrating to SQL Server for its PeopleSoft implementation. SQL Server would be a good choice but due to limited licensing for production instances and the fact that DSC has several other MySQL databases being used for third-party production systems, MySQL was chosen as the database to use, at least during development. After PeopleSoft is fully implemented the college may decide to migrate the Student Queue and Tracking system over to SQL Server, and perhaps even incorporate the functionality into its PeopleSoft instance.

Table 1. Project tables

Table name	Description
person	This table stores pertinent identifying attributes for persons associated with Daytona State College
staff	This table stores the id and password for the staff used in testing.
department	This table stores identifying codes for each office that will use the Student Queue and Tracking system.
dept_reason	This table stores the reason codes associated with each department.
staff_dept	This table stores the associations between staff and the department(s) they serve.
session	This table stores the data surrounding the session between the student and the college personnel.

Based on the requirements, it was determined that the system needed six tables. Table 1 lists the tables and gives a brief description of each one's purpose. The first table is for identification of each person that is associated with the college's ERP system. Since DSC is presently undergoing an ERP conversion from Jenzabar CX to PeopleSoft, a temporary person

record was created for use during development. This record contains only those attributes necessary for development and testing. The actual record in PeopleSoft will contain many more fields.

Table 2. person table

Column	Datatype	Length	Index/Key
id	INT	11	Primary Key
fleid	VARCHAR	20	Unique Index
first_name	VARCHAR	20	
last_name	VARCHAR	30	
phone	VARCHAR	12	

The staff table used here is also a temporary table being used for development and testing. This table simply provides the ability to test a staff member's college id and password for authentication to the application. When implemented, the software will utilize Daytona State College's production Microsoft Active Directory for its authentication and authorization mechanism.

Table 3. staff table

Column	Datatype	Length
staff_id	INT	11
pw	VARCHAR	10

The next table that needed to be created is the department table. This table is used to associate a code with each department that will use the Student Queue and Tracking system. The code has an effective date and an inactive date which will control when the departments can be used in the software. By controlling inactive codes with dates instead of deleting the records the

college can continue to run historical reports for all departments that have used the application and not just those that are active at the time of the reporting.

Two additional attributes have been included in the department table, one labeled `select_advisor` and the other is `dept_email`. The `select_advisor` is used to determine which departments wish for students to be allowed to choose a specific staff member when they login. The individual departments will make that determination and set the flag if they choose to allow it. The other column is used for storing a departmental email address if the department personnel are to be notified when a student has signed in for their services. If the field is left blank, no notification will take place.

Table 4. department table

Column	Datatype	Length	Index/Key
<code>dept_id</code>	VARCHAR	10	Primary Key / Unique Index
<code>dept_name</code>	VARCHAR	30	
<code>effective_date</code>	DATE		
<code>inactive_date</code>	DATE		
<code>select_advisor</code>	CHAR	1	
<code>dept_email</code>	VARCHAR	30	

All the Student Services departments desired to have the student select a reason for their visit to Daytona State College. Free-form text is not especially desirable for reporting purposes, therefore, it was decided that a table to control the reason codes for each office should be created. The `dept_reason` table was created for just that purpose. Once again, this table makes use of an effective date and an inactive date to control available options for selection. This table will be maintained by office personnel that have the security to do so.

Table 5. dept_reason table

Column	Datatype	Length	Index/Key
dept_id	VARCHAR	10	Unique Index
Reason	VARCHAR	20	Unique Index
effective_date	DATE		
inactive_date	DATE		

Most members of the Student Services staff at Daytona State College are associated with a single department (i.e. Financial Aid, Student Accounts, Advising, etc.), but those that serve on the regional campuses often assist students with multiple areas. The staff_dept table was created to maintain the associations between the staff members and the department(s) with which they serve.

Table 6. staff_dept table

Column	Datatype	Length	Index/Key
staff_id	INT	11	Primary Key
dept_id	VARCHAR	10	Primary Key
upd_this_tbl	CHAR	1	
upd_dept_reasons	CHAR	1	
effective_date	DATE		
inactive_date	DATE		

The final table created was the one for culminating the data of the actual session between the student and the staff member, the session table. The session is started when the student signs in. If the student is recognized in the system, the college id and name are transferred from the person table into the session table. Along with this data, the department the student needs assistance from and the reason code, if selected, for the visit. The time-stamp of the sign-in is

added as an indicator to the staff who is next in line as well as being used for reporting of wait times later.

When a student is selected from the queue by a college staff member, the selection is added to the session. Notes of the session can be added and updated by the staff member to be used in future sessions by any staff in the department. When the session has been completed, the staff member can end the session and the time-stamp is added to the record.


Table 7. session table

Column	Datatype	Length	Index/Key
sess_id	INT	11	Primary Key / Unique Index
stu_id	INT	11	Primary Key
stu_fname	VARCHAR	20	
stu_lname	VARCHAR	30	
dept_id	VARCHAR	10	
staff_id	INT	11	Primary Key
sess_date	DATE		
stu_sign_in_time	TIME		
start_time	TIME		
end_time	TIME		
reason	VARCHAR	20	
notes	VARCHAR	250	

Implementation (Development)

A total of five web pages were created for the system, Queue Login, Staff Login, Staff Session, Staff Departments, and Department Reasons. The Queue Login page (Figure 3) is used by students to sign in and enter the queue. Students must enter either their college id number, their phone number, or their first and last names to sign in. The department is also required, and

once it is selected, the appropriate reason codes for that department will populate the dropdown for selection.



**DAYTONA
STATE COLLEGE**

Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:

Phone #:

First Name:

Last Name:


Department:

Reason:

Select Advisor:

Figure 3. Queue Login page

The Staff Login page (Figure 4) is where the college personnel must enter their college id and password for authentication into the system. The system was developed using a simple table with two fields but it is anticipated that Active Directory will replace this mechanism upon the system's implementation into production.



**DAYTONA
STATE COLLEGE**

Staff Login:

Enter College ID and Password to Log in


ID Number:

Password:

Figure 4. Staff Login page

After successful authentication, the staff member is taken to the Staff Session page (Figure 5). This page will possibly display grids of records with which the employee can choose to interact. The top grid will display all those students waiting in the queue for any department the staff member is associated. The college personnel can elect any student in the list to start the session. Once selected, the student disappears from the grid.

The lower grid is for sessions with students that the staff member still has in an open status. Here, the notes from the session can be entered or updated and the session can be ended. Once ended, the session disappears from the list.



DAYTONA STATE COLLEGE

Edit Dept Reasons Edit Staff Depts

Staff Session:

Reload Queue

Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
49	STUACCTS	Martha	Washington	03-19-2016	14:08:36		Start Session
50	ADMISSIONS	Kristy	Davis	03-19-2016	18:09:24		Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Edit Notes	End Session

Figure 5. Staff Session page

At the very top of the page will appear two buttons if the staff member has the permissions to use them. One button is for updating the staff_dept table and the other is for updating the dept_reason table. These buttons open up separate web pages with separate security attached.

The Department Reasons page (Figure 6) displays a grid containing all active reason codes that are associated with the departments for which the employee has permissions to alter. Here, the employee can inactivate one of the codes. This will prevent the code from being selected in the future but will not affect its use in reporting of past sessions. A new row can be added by selecting one of the departments available and then entering a reason code and effective date. The click on the 'Add a row' button will save it to the database.



DAYTONA STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	RESIDENCY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	OTHER	12-25-2015	Inactivate Dept. Reason
STUACCTS	OTHER	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	TEST1	03-03-2016	Inactivate Dept. Reason

Add a row:

Department:

Reason:

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

Figure 6. Department Reasons page

The Staff Departments page (Figure 7) is used to manage the employee / department associations. The grid will display active associations for the departments that the college staff member has permissions to alter. Any record in the grid can be inactivated by the employee. Adding an associated is performed by making selections from the dropdowns and then clicking on the 'Add a row' button.



DAYTONA STATE COLLEGE

Employee ID	Employee's First Name	Employee's Last Name	Department	Update This Table	Update Depart Reasons	Effective Date	Select to set this record inactive
1	Don	Davis	ADMISSIONS	Y	Y	01-01-2016	Inactivate This Record
1	Don	Davis	STUACCTS	N	N	01-01-2016	Inactivate This Record
2	Rusty	Davis	ADMISSIONS	N	N	01-01-2016	Inactivate This Record
5	Anna	Davis	STUACCTS	N	N	03-03-2016	Inactivate This Record

Add a row:

Choose Staff:

Department:

Update Staff Depts?

Update Dept Reasons?

Effective Date:

To add a row, choose Staff, Department, and Effective Date and then click 'Add a Row' button.

Figure 7. Staff Departments page

CHAPTER 4

RESULTS AND DISCUSSION

This project involved a great number of hours from start to finish. The requirements gathering and analysis for the software started in September 2015 and ran through to early November 2015. The weekly hour's breakdown during this phase is found in Table 8 below. Most of the effort in September was spent discussing requirements with functional personnel trying to get to the crux of what each area really needed to have to finalize the scope of the project.

The first two weeks of October mostly involved putting the pieces together and finalizing the project plan and scope documents. Data flow and database design went through a couple iterations during the latter half of the month before a final product was ready. Even after the ERD was created and the DDL document was initially established, additional tweaking of certain table structures became necessary to accommodate appropriate security functionality that had been overlooked during the design.

Table 8. Hours for requirements gathering, analysis, and initial project documents

Start Date	End Date	Hours	Details
September 5, 2015	September 11, 2015	6	Requirements
September 12, 2015	September 18, 2015	4.5	Requirements
September 19, 2015	September 25, 2015	5	Requirements
September 26, 2015	October 02, 2015	5.25	Requirements
October 3, 2015	October 09, 2015	4	Plan and Scope
October 10, 2015	October 16, 2015	3.5	Plan and Scope
October 17, 2015	October 23, 2015	2	Data flow and Database
October 24, 2015	October 30, 2015	2.5	Data flow and Database
October 31, 2015	November 06, 2015	3.5	ERD

November 7, 2015	November 13, 2015	4	ERD
------------------	-------------------	---	-----

Since my RAD programming experience was primarily with a C++ tool called Borland C++ Builder, the programming for the project's software entailed a great deal of time and effort simply to learn the Visual Studio environment. Much time was spent going through tutorials presented by software developers on YouTube. Most of the videos were of little use but there were two presenters, Marie Taylor and Dr. Ron Eaglin, which proved quite helpful.

Once the environment was somewhat understood, many .Net Googling sessions took place to discover how to accomplish the specific tasks the software needed to perform in order that the required data flow be accomplished. Table 9 details the hours spent during the actual development / testing phase.

Table 9. Hours during development of software

Start Date	End Date	Hours	Details
January 04, 2016	January 10, 2016	3.5	Visual Studio Environment
January 11, 2016	January 17, 2016	4.5	Visual Studio Environment
January 18, 2016	January 24, 2016	26	Development – Queue Login
January 25, 2016	January 31, 2016	28.25	Development – Staff Login / Staff Session
February 01, 2016	February 07, 2016	19.5	Development – Staff Login / Staff Session
February 08, 2016	February 14, 2016	17	Development – Department Reasons / Staff Departments
February 15, 2016	February 21, 2016	10	Testing / Bug fixes
February 22, 2016	February 28, 2016	2.5	Testing
February 29, 2016	March 06, 2016	9	Development – Final tweaks

The original database design was adjusted a couple times during the development and early testing of the application. The changes were primarily due to overlooked options, like table

maintenance and associated security for staff, during the initial discovery phase. Once detected, the additional functionality was quickly remedied and added to the system.

Several additional features have been mentioned for the next version of the software. Most of the additional requests come from the Admissions area, which seems a bit odd since that office is promoting the online functionality for tasks such as admission application, discovery and resolution of missing admission documents, and resolving account holds related to admission. Table 10 lists the current requested additional functionality for the system.

Table 10. Additional requested functionality

Office	Request
Admissions	When linking to PeopleSoft, add display of student photo for identity verification
Admissions	Add SMS as notification to requested advisor
Admissions	Add column for "Wait Time" that is calculated from student sign-in time and current time
Admissions	Add notification to supervisors when certain criteria has been reached (# of students in the queue, average wait time exceeds an established limit)
Admissions	Record the session as a contact in PeopleSoft
Admissions	Add a lobby presentation view of the Queue with ID or phone number only
Records	Add an association of reasons to staff and limit those staff to students with those reasons.
Records	Add ability to send SMS survey questions to students selected from Queue

The lessons learned from this project are varied. Perhaps the greatest lesson is that one cannot count on functional personnel for testing a new application and providing timely feedback on that testing if an ERP implementation is underway. The early commitment and exuberance of the Student Services department staff quickly faded as the PeopleSoft implementation gained momentum. This altered priority is understandable though not really anticipated.

Another lesson is more of a reinforcement of already gleaned wisdom. Project planning is necessary for a successful project but one has to be flexible when the actual work commences.

Regardless of how careful and thorough the gathering and analysis of system requirements is performed, rarely will those details be truly complete. One should expect to alter the requirements during the early stages of development and testing, and sometimes even through to implementation. The key is to be willing to negotiate with the users and other project stakeholders to determine which changes will be effected now and which can wait for a later time.

CHAPTER 5

CONCLUSIONS

This project was a definite success. The process undertaken reinforced the techniques learned during the MSIS program with the requirements gathering and analysis phase, carrying through the development and testing phase, and culminating with the deployment of the final product.

The testing by the individual offices of the Student Services department has been limited due to restricted resources. The migration, testing, implementation, and post go-live activities surrounding the PeopleSoft Campus Solutions implementation understandably have taken precedence over this application. It is fully expected, however, that greater attention will be focused on the Student Queue and Tracking system sometime in the fall of 2016 as DSC experiences its first major semester with PeopleSoft for students.

After the limited testing and, a demonstration and explanation of the final version of the software, it has been unanimously acknowledged by the leaders of each office that the system appears to have met all of the agreed upon requirements that were established at the onset of the project. The overwhelming expectation is that students will experience less anxiety with the system than is experienced currently, and the thinking is that with the additional ability afforded to staff for keeping notes in one system, all sessions for the student will be more beneficial.

Even though there was a great deal of additional learning involved to use Microsoft Visual Studio for development of the software, it was quite enjoyable. The new application can

serve the college well for some time to come but it is expected that a conversion to People Code, and placement within the new Daytona State College portal, will transpire in the next couple years. This conversion to PeopleSoft will require someone at DSC to become proficient with PeopleSoft's People Tools, a feat not yet pursued.

REFERENCES

Eaglin, R. (2013, April 12). Web programming - Visual studio 2012 getting started part 1.

Retrieved from <https://www.youtube.com/watch?v=reCJVsiN3Qw>

Eaglin, R. (2013, April 15). Web programming - Getting started with visual studio 2012 part

2. Retrieved from <https://www.youtube.com/watch?v=Aogm8TkxS2g>

Maister, D. H. (n.d.). *The psychology of waiting*. Retrieved from

http://www.columbia.edu/~ww2040/4615S13/Psychology_of_Waiting_Lines.pdf

MySQL :: MySQL 5.7 Reference Manual :: 11.1.2 Date and Time Type Overview. (n.d.).

Retrieved from <http://dev.mysql.com/doc/refman/5.7/en/date-and-time-type-overview.html>

MySQL :: MySQL 5.7 Reference Manual :: 13.1.8 ALTER TABLE Syntax. (n.d.). Retrieved

from <http://dev.mysql.com/doc/refman/5.7/en/alter-table.html>

SDLC – Waterfall model. (n.d.). Retrieved from

http://www.tutorialspoint.com/sdlc/pdf/sdlc_waterfall_model.pdf

Swanson, A. (2015, November 27). What really drives you crazy about waiting in line (it actually isn't the wait at all). Retrieved from

<https://www.washingtonpost.com/news/wonk/wp/2015/11/27/what-you-hate-about-waiting-in-line-isnt-the-wait-at-all/>

Taylor, M. (2014, July 15). Introduction to ASP.NET web forms: Basic server controls.

Retrieved from <https://www.youtube.com/watch?v=Tkm4mC4IYGQ>

Taylor, M. (2014, July 15). Introduction to ASP.NET web forms: Web server control events.

Retrieved from

<https://www.youtube.com/watch?v=3HQKq8eSymc&index=5&list=PLtMFeKNXw-GH0wC0tTM1Tb8wgOw7NV9EJ>

APPENDICES

APPENDIX A: USERS' MANUAL

Student Queue Login

1. Sign into the Queue and Tracking system by entering either your College ID, your Phone #, or your First and Last Name.



Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:	<input type="text"/>
Phone #:	<input type="text"/>
First Name:	<input type="text"/>
Last Name:	<input type="text"/>
Department:	<input type="text" value=" <Select Department>"/>
Reason:	<input type="text" value="▼"/>
Select Advisor:	<input type="text" value="▼"/>
	<input type="button" value="Log In"/> <input type="button" value="Clear Form Info"/>

2. Select a Department from the dropdown list.



Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:

Phone #:

First Name:

Last Name:

Department: Admissions ▼

Reason: <Select Reason> ▼

Select Advisor: <Select Advisor> ▼

3. Select a Reason from the dropdown list. Then if the select Department allows selection of a specific Advisor, select one from the list if you choose.



Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:

Phone #:

First Name:

Last Name:

Department: Admissions ▼

Reason: APPLY ▼

Select Advisor: <Select Advisor> ▼

4. Click the Log In button to join the queue.



Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:

Phone #:

First Name:

Last Name:

Department:

Reason:

Select Advisor:

5. If sign-in is successful, the message highlighted below will be seen.



Student Queue Login:

Please enter your college id, phone #, or first & last name:

College ID:

Phone #:

First Name:

Last Name:

Department:

Reason:

Select Advisor:

Entry into queue was successful. Please be patient while you wait your turn.

Staff Login

1. Enter your ID Number and Password and click the Log In button.



Staff Login:


Enter College ID and Password to Log in

ID Number:

Password:

Staff Session

1. Upon successful login, staff is presented with the session page. This page will display the Queued students in the top grid, and the current sessions for this staff member in the bottom grid. If the staff member has security to add/edit the Department Reasons and/or the Staff Departments, the buttons will display at the top of the page.



Staff Session:


Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	<input type="button" value="Start Session"/>

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	<input type="button" value="Edit Notes"/>	<input type="button" value="End Session"/>

2. To start a session with a student in the queue, simply click the Start Session button associated with the appropriate student in the top grid.



DAYTONA STATE COLLEGE

Edit Dept Reasons Edit Staff Depts

Staff Session:

Reload Queue


Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Edit Notes	End Session

3. To add/edit notes for a student's session, click the Edit Notes button.



DAYTONA STATE COLLEGE

Edit Dept Reasons Edit Staff Depts

Staff Session:

Reload Queue

Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Edit Notes	End Session

4. Add/edit notes as needed in the text box made available.



Edit Dept Reasons Edit Staff Depts

Staff Session:

Reload Queue

Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Update Notes Cancel	End Session

5. To save the changes, click the Update Notes button. To discard the changes, click on Cancel.



Edit Dept Reasons Edit Staff Depts

Staff Session:

Reload Queue

Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Update Notes Cancel	End Session

6. When the session is over, click the End Session button to end it.



DAYTONA
STATE COLLEGE

Edit Dept Reasons

Edit Staff Depts

Staff Session:

Reload Queue

Choose a student


Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	Start Session

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	Edit Notes	End Session

Department Reasons

1. To add/edit the department reasons, click on the Edit Dept Reasons button.



DAYTONA STATE COLLEGE

Staff Session:


Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	<input type="button" value="Start Session"/>

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	<input type="button" value="Edit Notes"/>	<input type="button" value="End Session"/>

2. To inactivate a department/reason association, click the Inactivate Dept. Reason button.



DAYTONA STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	<input type="button" value="Inactivate Dept. Reason"/>
ADMISSIONS	RESIDENCY	12-25-2015	<input type="button" value="Inactivate Dept. Reason"/>
ADMISSIONS	OTHER	12-25-2015	<input type="button" value="Inactivate Dept. Reason"/>
STUACCTS	OTHER	12-25-2015	<input type="button" value="Inactivate Dept. Reason"/>
ADMISSIONS	TEST1	03-03-2016	<input type="button" value="Inactivate Dept. Reason"/>
STUACCTS	DELACCOUNT	04-01-2016	<input type="button" value="Inactivate Dept. Reason"/>

Add a row:

Department:

Reason:

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

3. To add a new association, select a Department from the dropdown list.



DAYTONA
STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	RESIDENCY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	OTHER	12-25-2015	Inactivate Dept. Reason
STUACCTS	OTHER	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	TEST1	03-03-2016	Inactivate Dept. Reason
STUACCTS	DELACCOUNT	04-01-2016	Inactivate Dept. Reason

Add a row:

Department: <Select Department> ▼

Reason:

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

Add a Row

4. Enter the new Reason code.



DAYTONA
STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	RESIDENCY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	OTHER	12-25-2015	Inactivate Dept. Reason
STUACCTS	OTHER	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	TEST1	03-03-2016	Inactivate Dept. Reason
STUACCTS	DELACCOUNT	04-01-2016	Inactivate Dept. Reason

Add a row:

Department: Admissions ▼

Reason: ANY REASON

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

Add a Row

5. Select the date the new record is to be effective.



DAYTONA
STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	RESIDENCY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	OTHER	12-25-2015	Inactivate Dept. Reason
STUACCTS	OTHER	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	TEST1	03-03-2016	Inactivate Dept. Reason
STUACCTS	DELACCOUNT	04-01-2016	Inactivate Dept. Reason

Add a row:

Department:

Reason:

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

6. Click the Add a Row button to save the new record.



DAYTONA
STATE COLLEGE

Department	Reason	Effective Date	Select to set dept/reason inactive
ADMISSIONS	APPLY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	RESIDENCY	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	OTHER	12-25-2015	Inactivate Dept. Reason
STUACCTS	OTHER	12-25-2015	Inactivate Dept. Reason
ADMISSIONS	TEST1	03-03-2016	Inactivate Dept. Reason
STUACCTS	DELACCOUNT	04-01-2016	Inactivate Dept. Reason

Add a row:

Department:


Reason:

Effective Date:

To add a row, add Department, Reason, and Effective Date and then click 'Add a Row' button.

Staff Departments

1. To add/edit staff/department associations, click on the Edit Staff Depts button.



DAYTONA STATE COLLEGE

Staff Session:


Choose a student

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Reason	Select to start session
52	ADMISSIONS	Don	Davis	04-09-2016	09:29:06	APPLY	<input type="button" value="Start Session"/>

Choose a student to end session

Session ID	Department	Student First Name	Student Last Name	Session Date	Sign In Time	Session Start Time	Reason	Notes	Edit	Select to end session
48	ADMISSIONS	Anna	Davis	03-06-2016	15:40:06	16:01:59	APPLY	Add some notes to display.	<input type="button" value="Edit Notes"/>	<input type="button" value="End Session"/>

2. To inactivate a staff/department association, click on the Inactivate This Record button for the appropriate record.



DAYTONA STATE COLLEGE

Employee ID	Employee's First Name	Employee's Last Name	Department	Update This Table	Update Depart Reasons	Effective Date	Select to set this record inactive
1	Don	Davis	ADMISSIONS	Y	Y	01-01-2016	<input type="button" value="Inactivate This Record"/>
1	Don	Davis	STUACCTS	N	N	01-01-2016	<input type="button" value="Inactivate This Record"/>
2	Rusty	Davis	ADMISSIONS	N	N	01-01-2016	<input type="button" value="Inactivate This Record"/>
5	Anna	Davis	STUACCTS	N	N	03-03-2016	<input type="button" value="Inactivate This Record"/>

Add a row:

Choose Staff:

Department:

Update Staff Depts?

Update Dept Reasons?

Effective Date:

To add a row, choose Staff, Department, and Effective Date and then click 'Add a Row' button.

5. If the staff member should be allowed to add/edit the Staff/Departments associations or add/edit the Department Reasons, update the appropriate dropdown to a Y.



Employee ID	Employee's First Name	Employee's Last Name	Department	Update This Table	Update Depart Reasons	Effective Date	Select to set this record inactive
1	Don	Davis	ADMISSIONS	Y	Y	01-01-2016	Inactivate This Record
1	Don	Davis	STUACCTS	N	N	01-01-2016	Inactivate This Record
2	Rusty	Davis	ADMISSIONS	N	N	01-01-2016	Inactivate This Record
5	Anna	Davis	STUACCTS	N	N	03-03-2016	Inactivate This Record

Add a row:

Choose Staff: <Select Staff> ▼

Department: <Select Department> ▼

Update Staff Depts? N ▼

Update Dept Reasons? N ▼

Effective Date:

To add a row, choose Staff, Department, and Effective Date and then click 'Add a Row' button.

Add a Row

6. Set the effective date for this association.



Employee ID	Employee's First Name	Employee's Last Name	Department	Update This Table	Update Depart Reasons	Effective Date	Select to set this record inactive
1	Don	Davis	ADMISSIONS	Y	Y	01-01-2016	Inactivate This Record
1	Don	Davis	STUACCTS	N	N	01-01-2016	Inactivate This Record
2	Rusty	Davis	ADMISSIONS	N	N	01-01-2016	Inactivate This Record
5	Anna	Davis	STUACCTS	N	N	03-03-2016	Inactivate This Record

Add a row:

Choose Staff: <Select Staff> ▼

Department: <Select Department> ▼

Update Staff Depts? N ▼

Update Dept Reasons? N ▼

Effective Date: 2016-04-01

To add a row, choose Staff, Department, and Effective Date and then click 'Add a Row' button.

Add a Row

7. Click the Add a Row button to save the new record.



DAYTONA
STATE COLLEGE

Employee ID	Employee's First Name	Employee's Last Name	Department	Update This Table	Update Dept Reasons	Effective Date	Select to set this record inactive
1	Don	Davis	ADMISSIONS	Y	Y	01-01-2016	Inactivate This Record
1	Don	Davis	STUACCTS	N	N	01-01-2016	Inactivate This Record
2	Rusty	Davis	ADMISSIONS	N	N	01-01-2016	Inactivate This Record
5	Anna	Davis	STUACCTS	N	N	03-03-2016	Inactivate This Record

Add a row:

Choose Staff: <Select Staff> ▼

Department: <Select Department> ▼

Update Staff Depts? N ▼

Update Dept Reasons? N ▼

Effective Date: 2016-04-01

To add a row, choose Staff, Department, and Effective Date and then click 'Add a Row' button.

Add a Row ←

APPENDIX B: SYSTEM TECHNICAL DOCUMENTATION

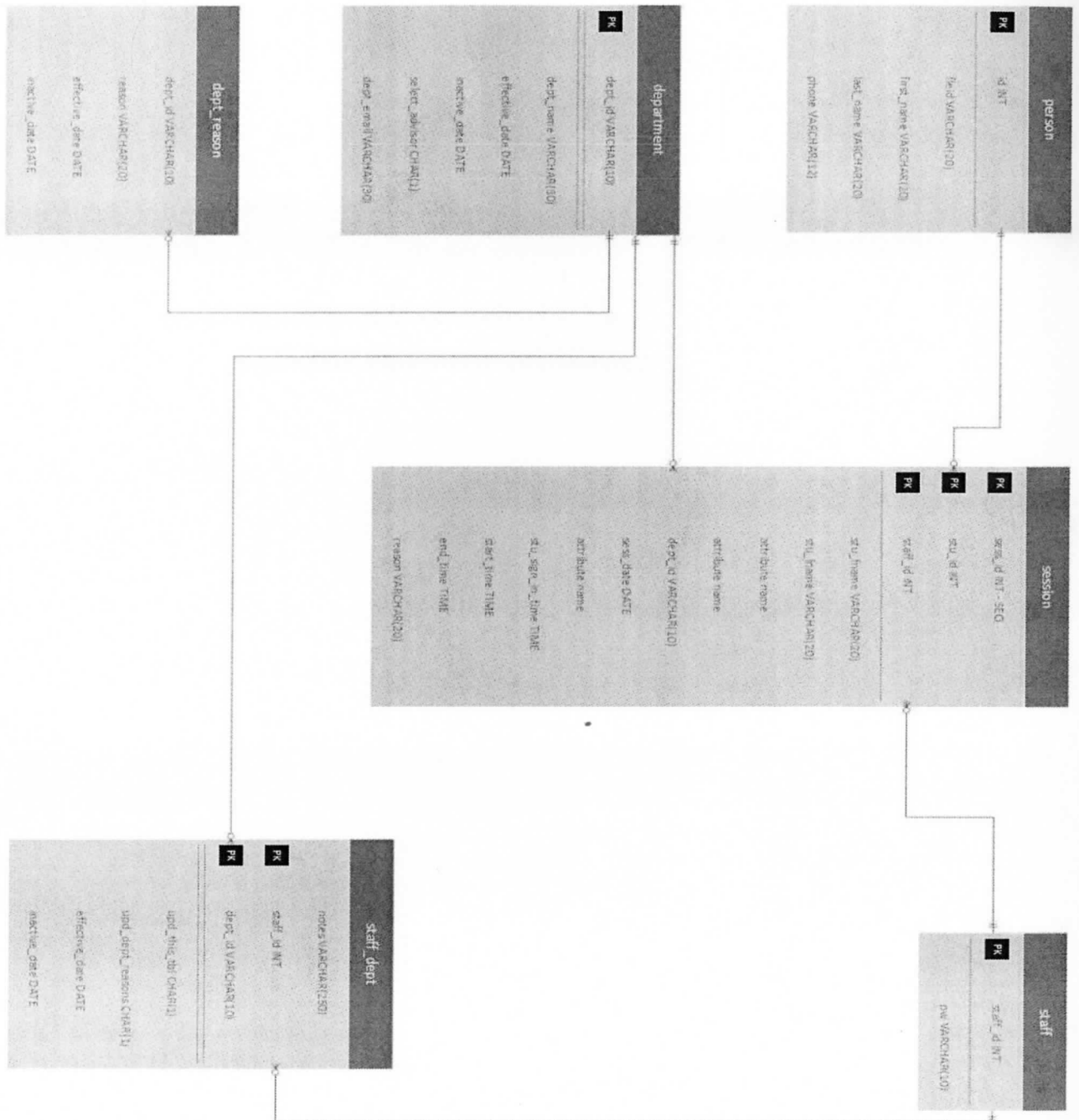


Figure 8. Entity Relationship Diagram

Database DDL

```

-- MySQL Script generated by MySQL Workbench
-- 01/12/16 20:22:34
-- Model: New Model   Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-----
-- Schema mydb
-----
-----
-- Schema drdavis
-----
-----
-- Schema drdavis
-----
CREATE SCHEMA IF NOT EXISTS `drdavis` DEFAULT CHARACTER SET utf8 ;
USE `drdavis` ;
-----
-- Table `drdavis`.`department`
-----
CREATE TABLE IF NOT EXISTS `drdavis`.`department` (
  `dept_id` VARCHAR(10) NOT NULL COMMENT 'Unique identifier for each
department.',
  `dept_name` VARCHAR(30) NOT NULL COMMENT 'Department name.',
  `effective_date` DATE NOT NULL COMMENT 'The date this department record
becomes valid.',

```

```
`inactive_date` DATE NULL DEFAULT NULL COMMENT 'The date this department
record is no longer valid.',
```

```
`select_advisor` CHAR(1) NULL DEFAULT NULL COMMENT 'Does this
department allow students to select a specific advisor?',
```

```
`dept_email` VARCHAR(30) NULL DEFAULT NULL COMMENT 'Email address is
necessary if the department wants its staff notified when a student has signed in for their
department',
```

```
PRIMARY KEY (`dept_id`),
```

```
UNIQUE INDEX `dept_id` (`dept_id` ASC)
```

```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;
```

```
-----
```

```
-- Table `drdavis`.`dept_reason`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `drdavis`.`dept_reason` (
```

```
`dept_id` VARCHAR(10) NOT NULL COMMENT 'The department id associated that
can use this code record.',
```

```
`reason` VARCHAR(20) NULL DEFAULT NULL,
```

```
`effective_date` DATE NOT NULL COMMENT 'The date this reason record becomes
valid.',
```

```
`inactive_date` DATE NULL DEFAULT NULL COMMENT 'The date this reason
record is no longer valid.',
```

```
UNIQUE INDEX `ucDeptReason` (`dept_id` -ASC, `reason` ASC)
```

```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;
```

```
-----
```

```
-- Table `drdavis`.`person`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `drdavis`.`person` (
```

```
`id` INT(11) NOT NULL DEFAULT '0' COMMENT 'Identification number for each
person. This is unique to each individual.',
```

```

`fleid` VARCHAR(20) NOT NULL COMMENT 'Florida Education ID. Unique to
each person that has ever been a student or teacher in any Florida school K-20.',
`first_name` VARCHAR(20) NULL DEFAULT NULL COMMENT 'First name of
person associated with this record.',
`last_name` VARCHAR(30) NULL DEFAULT NULL COMMENT 'Last name of
person associated with this record.',
`phone` VARCHAR(12) NULL DEFAULT NULL COMMENT 'The phone number is
one of the fields that can be used to identify that student as they sign in.',
PRIMARY KEY (`id`),
UNIQUE INDEX `fleid` (`fleid` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-----
-- Table `drdavis`.`session`
-----
CREATE TABLE IF NOT EXISTS `drdavis`.`session` (
`sess_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier
for each session record.',
`stu_id` INT(11) NOT NULL DEFAULT '0' COMMENT 'Identifies the student
associated with this session. Links to person.fleid.',
`stu_fname` VARCHAR(20) NULL DEFAULT NULL,
`stu_lname` VARCHAR(30) NULL DEFAULT NULL,
`dept_id` VARCHAR(10) NULL DEFAULT NULL,
`staff_id` INT(11) NOT NULL DEFAULT '0' COMMENT 'Identifies the staff member
associated with this session. Links to person.fleid.',
`sess_date` DATE NULL DEFAULT NULL COMMENT 'Date of this session.',
`stu_sign_in_time` TIME NULL DEFAULT NULL COMMENT 'Time the student
signed into the system.',
`start_time` TIME NULL DEFAULT NULL COMMENT 'Time the staff member
initiates the interaction with the student.',

```

```
`end_time` TIME NULL DEFAULT NULL COMMENT 'Time the staff member ends
the session.',
```

```
`reason` VARCHAR(20) NULL DEFAULT NULL COMMENT 'The reason for the
visit. Links to dept_reason.reason.',
```

```
`notes` VARCHAR(250) NULL DEFAULT NULL COMMENT 'Session notes added
by staff member.',
```

```
PRIMARY KEY (`sess_id`, `stu_id`, `staff_id`),
```

```
UNIQUE INDEX `sess_id` (`sess_id` ASC)
```

```
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 50
```

```
DEFAULT CHARACTER SET = utf8;
```

```
-----
```

```
-- Table `drdavis`.`staff`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `drdavis`.`staff` (
```

```
  `staff_id` INT(11) NULL DEFAULT NULL,
```

```
  `pw` VARCHAR(10) NULL DEFAULT NULL)
```

```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;
```

```
-----
```

```
-- Table `drdavis`.`staff_dept`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `drdavis`.`staff_dept` (
```

```
  `staff_id` INT(11) NOT NULL DEFAULT '0' COMMENT 'Staff identifier associated
with this record. Links to person.id.',
```

```
  `dept_id` VARCHAR(10) NOT NULL COMMENT 'The department identifier
associated with this record. Links to department.dept_id.',
```

```
  `upd_this_tbl` CHAR(1) NULL DEFAULT 'N' COMMENT 'Can the staff member
update this table?',
```

```
  `upd_dept_reasons` CHAR(1) NULL DEFAULT 'N' COMMENT 'Can the staff
member update the dept_reason table?',
```

```
`effective_date` DATE NULL DEFAULT NULL COMMENT 'The date that this record
becomes valid.',
```

```
`inactive_date` DATE NULL DEFAULT NULL COMMENT 'The date that this record
is no longer valid.',
```

```
PRIMARY KEY (`staff_id`, `dept_id`),
```

```
INDEX `dept_id_rel_to_staff_dept_dept_id` (`dept_id` ASC),
```

```
CONSTRAINT `dept_id_rel_to_staff_dept_dept_id`
```

```
FOREIGN KEY (`dept_id`)
```

```
REFERENCES `drdavis`.`department` (`dept_id`),
```

```
CONSTRAINT `id_rel_to_staff_dept_staff_id`
```

```
FOREIGN KEY (`staff_id`)
```

```
REFERENCES `drdavis`.`person` (`id`))
```

```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
```

```
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
```

```
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Source Code

QueueLogin.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="QueueLogin.aspx.cs"
Inherits="DonDavisMSISFinalProject.QueueLogin" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<title>Student Queue</title>
```

```
<link rel="stylesheet" href="StudentQueueStyleSheet.css" />
```

```
<!-- ##### -->
```

```
<!-- Example jQuery Reference -->
```

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
```

```

<!-- Example jQuery Masking Script -->
<script src="http://digitalbush.com/wp-content/uploads/2013/01/jquery.maskedinput-
1.3.1.min.js"></script>
<meta charset=utf-8 />
<script type='text/javascript'>
$(function(){
  // Define your mask (using 9 to denote any digit)
  $('#txtPhone').mask('(999) 999-9999');
});
</script>
<!-- ##### -->
</head>
<body>
  <form id="frmQueueLogin" runat="server">
    <asp:Image ID="imgLogo" ImageUrl="~/logo.png" runat="server" />
    <h3>Student Queue Login: </h3>
    <br />
    <div>
      <asp:Label ID="lblInstruction" Text="Please enter your college id, phone #, or first & last
name:" runat="server" />
    </div>
    <br />
    <div>
      <asp:Label ID="lblID" Text="College ID:" runat="server" CssClass="labelClass"/>
      <asp:TextBox ID="txtID" runat="server" AutoComplete="off" />
    </div>
    <br />
    <div>
      <asp:Label ID="lblPhone" Text="Phone #:" runat="server" CssClass="labelClass" />
      <asp:TextBox ID="txtPhone" runat="server" AutoComplete="off" />
    </div>
    <br />
    <div>
      <asp:Label ID="lblFirstName" Text="First Name:" runat="server" CssClass="labelClass"/>
      <asp:TextBox ID="txtFirstName" runat="server" AutoComplete="off" />
    </div>
    <br />
    <div>
      <asp:Label ID="lblLastName" Text="Last Name:" runat="server" CssClass="labelClass"/>
      <asp:TextBox ID="txtLastName" runat="server" AutoComplete="off" />
    </div>
    <br />
    <div>

```

```

    <asp:Label ID="lblDepartment" Text="Department:" runat="server"
    CssClass="labelClass"/>
    <asp:DropDownList ID="ddlDepartment" runat="server" AutoPostBack="true"
    OnSelectedIndexChanged="ddlDepartment_SelectedIndexChanged"/>
  </div>
  <br />

  <div>
    <asp:Label ID="lblReason" Text="Reason:" runat="server" CssClass="labelClass"/>
    <asp:DropDownList ID="ddlReason" runat="server"
    OnSelectedIndexChanged="ddlReason_SelectedIndexChanged"/>
  </div>
  <br />
  <div>
    <asp:Label ID="lblAdvisor" Text="Select Advisor:" runat="server"
    CssClass="labelClass"/>
    <asp:DropDownList ID="ddlAdvisors" runat="server" />
  </div>
  <br />
  <div>
    <asp:Button ID="btnGetInfo" Text="Log In" runat="server" OnClick="getInfo" />
    <asp:Button ID="btnClearForm" Text="Clear Form Info" runat="server"
    OnClick="btnClearForm_Click" />
  </div>
  <br />
  <div>
    <asp:GridView ID="gvStudentLogin" AutoGenerateColumns="false" runat="server"
    OnRowCommand="gvStudentLogin_RowCommand">
      <Columns>
        <asp:BoundField DataField="id" HeaderText="ID" ReadOnly="true" />
        <asp:BoundField DataField="first_name" HeaderText="First Name" ReadOnly="true"
        />
        <asp:BoundField DataField="last_name" HeaderText="Last Name" ReadOnly="true"
        />
        <asp:BoundField DataField="phone" HeaderText="Phone" ReadOnly="true" />
        <asp:buttonfield buttontype="Button" commandname="Select" headertext="Select to
        Login to Queue" text="Login to Queue"/>
      </Columns>
    </asp:GridView>
  </div>
  <br />
  <div>
    <asp:Label ID="lblMessage" Text="" runat="server" />
  </div>
</form>

```

```
</body>  
</html>
```

QueueLogin.aspx.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.Net;  
using System.Net.Mail;  
using System.Data;  
  
namespace DonDavisMSISFinalProject  
{  
    public partial class QueueLogin : System.Web.UI.Page  
    {  
        MySql.Data.MySqlClient.MySqlConnection conn;  
        MySql.Data.MySqlClient.MySqlConnection connInsert;  
        MySql.Data.MySqlClient.MySqlCommand cmd;  
        MySql.Data.MySqlClient.MySqlCommand cmdInsert;  
        MySql.Data.MySqlClient.MySqlDataReader reader;  
  
        String qryString;  
        String id;  
        int defId = 0;  
  
        protected void Page_Load(object sender, EventArgs e)  
        {  
            if(!IsPostBack)  
            {  
                loadDepartments();  
            }  
        }  
  
        private void connectToDB()  
        {  
            try  
            {
```



```

        String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
        conn = new MySql.Data.MySqlClient.MySqlConnection(connString);
        conn.Open();
    }
    catch(Exception e)
    {
        lblMessage.Text = e.Message;
    }
}

private void loadDepartments()
{
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    connectToDB();
    qryString = "select dept_id, dept_name from department where effective_date <=?today
and ";
    qryString = qryString + " (inactive_date is null or inactive_date = '' or inactive_date >
?today)";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?today", nowDate);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch(Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
    }
    ListItem newItem = new ListItem();
    newItem.Text = "<Select Department>";
    newItem.Value = "0";
    ddlDepartment.Items.Add(newItem);
    while (reader.HasRows && reader.Read())
    {
        newItem = new ListItem();
        newItem.Text = reader["dept_name"].ToString();
        newItem.Value = reader["dept_id"].ToString();
        ddlDepartment.Items.Add(newItem);
    }
    reader.Close();
}

private void loadReasons()

```

```

{
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    connectToDB();
    qryString = "select dept_id, reason from dept_reason where dept_id=?dept_id and
effective_date <=?today ";
    qryString = qryString + " and (inactive_date is null or inactive_date = '' or inactive_date
> ?today)";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?dept_id", ddlDepartment.SelectedItem.Value);
    cmd.Parameters.AddWithValue("?today", nowDate);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
    }
    ddlReason.Items.Clear();
    ListItem newItem = new ListItem();
    newItem.Text = "<Select Reason>";
    newItem.Value = "0";
    ddlReason.Items.Add(newItem);
    int knt = 1;
    while (reader.HasRows && reader.Read())
    {
        newItem = new ListItem();
        newItem.Text = reader["reason"].ToString();
        newItem.Value = knt.ToString();
        ddlReason.Items.Add(newItem);
        knt++;
    }
    reader.Close();
    loadAdvisors();
}

```

```

private void loadAdvisors()
{
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    bool selectAdvisor = false;
    connectToDB();
    ddlAdvisors.Items.Clear();
    ddlAdvisors.Enabled = false;
    qryString = "select sd.staff_id, CONCAT(p.first_name, ' ', p.last_name) as name from
staff_dept sd, department d, person p ";

```

```

    qryString = qryString + "where d.dept_id =?dept_id and sd.dept_id = d.dept_id and p.id =
sd.staff_id and d.effective_date <= " + nowDate + " ";
    qryString = qryString + "and (d.inactive_date is null or d.inactive_date = " or
d.inactive_date > " + nowDate + ") and d.select_advisor = 'Y'";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?dept_id", ddlDepartment.SelectedItem.Value);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
    }
    ListItem newItem = new ListItem();
    newItem.Text = "<Select Advisor>";
    newItem.Value = "0";
    ddlAdvisors.Items.Add(newItem);

    while (reader.HasRows && reader.Read())
    {
        newItem = new ListItem();
        newItem.Text = reader["name"].ToString();
        newItem.Value = reader["staff_id"].ToString();
        ddlAdvisors.Items.Add(newItem);
        selectAdvisor = true;
    }
    ddlAdvisors.Enabled = selectAdvisor;
}

protected void getInfo(object sender, EventArgs e)
{
    int chk4data = checkForData();
    if (chk4data == -1)
    {
        lblMessage.Text = "Either College ID, Phone #, or both First and Last Name must be
entered.";
        return;
    }
    else if(chk4data == -2)
    {
        lblMessage.Text = "Department must be selected.";
        return;
    }
    bool addSess = true;
    id = txtID.Text;

```

```

String phone = txtPhone.Text;
connectToDB();
if (string.IsNullOrEmpty(txtID.Text))
{
    bool returnEarly = false;
    MySql.Data.MySqlClient.MySqlCommand cmdKnt;
    MySql.Data.MySqlClient.MySqlDataReader readerKnt;
    qryString = "select count(*) knt from person where ";
    if (!string.IsNullOrEmpty(txtPhone.Text))
    {
        qryString = qryString + "phone =?phone";
        cmdKnt = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
        cmdKnt.Parameters.AddWithValue("?phone", phone);
    }
    else
    {
        qryString = qryString + "first_name =?fname and last_name =?lname";
        cmdKnt = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
        cmdKnt.Parameters.AddWithValue("?fname", txtFirstName.Text);
        cmdKnt.Parameters.AddWithValue("?lname", txtLastName.Text);
    }
    readerKnt = cmdKnt.ExecuteReader();
    if (readerKnt.HasRows && readerKnt.Read())
    {
        int knt = Convert.ToInt32(readerKnt.GetString(readerKnt.GetOrdinal("knt")));
        if (knt > 1)
        {
            readerKnt.Close();
            lblMessage.Text = "More than one record returned for search criteria. Please
select appropriate record.";
            returnEarly = true;
            MySql.Data.MySqlClient.MySqlCommand cmdLookup;
            qryString = "select id, first_name, last_name, phone from person where ";
            if (!string.IsNullOrEmpty(txtPhone.Text))
            {
                qryString = qryString + "phone =?phone";
                cmdLookup = new MySql.Data.MySqlClient.MySqlCommand(qryString,
conn);
                cmdLookup.Parameters.AddWithValue("?phone", phone);
            }
            else
            {
                qryString = qryString + "first_name =?fname and last_name =?lname";
                cmdLookup = new MySql.Data.MySqlClient.MySqlCommand(qryString,
conn);
                cmdLookup.Parameters.AddWithValue("?fname", txtFirstName.Text);

```

```

        cmdLookup.Parameters.AddWithValue("?lname", txtLastName.Text);
    }
    MySql.Data.MySqlClient.MySqlDataAdapter sdaLookup = new
MySql.Data.MySqlClient.MySqlDataAdapter(cmdLookup);
    DataSet dsLookup = new DataSet();
    sdaLookup.Fill(dsLookup);
    if (dsLookup.Tables[0].Rows.Count >= 0)
    {
        gvStudentLogin.DataSource = dsLookup;
        gvStudentLogin.DataBind();
        gvStudentLogin.Visible = true;
    }
}

}
readerKnt.Close();
if(returnEarly)
{
    return;
}
}
qryString = "select * from person where ";

if (!string.IsNullOrEmpty(txtID.Text))
{
    qryString = qryString + "id=?id";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?id", id);
}
else if (!string.IsNullOrEmpty(txtPhone.Text))
{
    qryString = qryString + "phone=?phone";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?phone", phone);
}
else
{
    qryString = qryString + "first_name=?fname and last_name=?lname";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?fname", txtFirstName.Text);
    cmd.Parameters.AddWithValue("?lname", txtLastName.Text);
}
reader = cmd.ExecuteReader();

if (reader.HasRows && reader.Read())
{

```

```

txtID.Text = reader.GetString(reader.GetOrdinal("id"));
txtFirstName.Text = reader.GetString(reader.GetOrdinal("first_name"));
txtLastName.Text = reader.GetString(reader.GetOrdinal("last_name"));
id = txtID.Text;
}
else if (string.IsNullOrEmpty(id) && string.IsNullOrEmpty(phone))
{
    id = defId.ToString();
}
else
{
    addSess = false;
    txtID.Text = "";
    txtPhone.Text = "";
    lblMessage.Text = "Unable to find college id: " + id + " or phone #: " + phone + " .
Please enter first and last names to enter the queue.";
}
if (conn.State != System.Data.ConnectionState.Broken && conn.State !=
System.Data.ConnectionState.Closed)
{
    conn.Close();
}
if (addSess)
{
    String reason = " ";
    if (ddlReason.SelectedItem.Text != "<Select Reason>")
    {
        reason = ddlReason.SelectedItem.Text;
        lblMessage.Text = reason;
    }
    int advisor = 0;
    if (ddlAdvisors.SelectedItem.Text != "<Select Advisor>")
    {
        advisor = Convert.ToInt32(ddlAdvisors.SelectedItem.Value);
    }
    if (addSession(id, txtFirstName.Text, txtLastName.Text,
ddlDepartment.SelectedItem.Value, reason, advisor))
    {
        lblMessage.Text = "Entry into queue was successful. Please be patient while you
wait your turn.";
    }
    else
    {
        //lblMessage.Text = "Entry into queue was not successful. Please try again or
contact college personnel for help.";
    }
}

```

```

    }
    //clearForm(); where should I put this???
}

private int checkForData()
{
    int retValue = 0;
    if (string.IsNullOrEmpty(txtID.Text))
    {
        if (string.IsNullOrEmpty(txtPhone.Text))
        {
            if ((string.IsNullOrEmpty(txtFirstName.Text)) ||
(string.IsNullOrEmpty(txtLastName.Text)))
            {
                retValue = -1;
            }
        }
    }
    if(retValue == 0)
    {
        if(ddlDepartment.SelectedIndex == 0)
        {
            retValue = -2;
        }
    }
    return retValue;
}

private void clearForm()
{
    lblMessage.Text = "";
    txtFirstName.Text = "";
    txtLastName.Text = "";
    txtID.Text = "";
    ddlDepartment.ClearSelection();
    ddlDepartment.Items.FindByValue("0").Selected = true;
    //need to clear the list of reasons here
    ddlReason.Items.Clear();
    txtPhone.Text = "";
    ddlAdvisors.Items.Clear();
    ddlAdvisors.Enabled = false;
    gvStudentLogin.Visible = false;
}

private bool addSession(String id, String fname, String lname, String dept, String reason, int
advisor)

```

```

{
    bool retValue = true;
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    String nowTime = DateTime.Now.ToString("HH:mm:ss");
    qryString = "insert into session (sess_id, stu_id, stu_fname, stu_lname, dept_id, reason,
staff_id, sess_date, stu_sign_in_time) ";
    qryString = qryString + "values (0,'" + id + "','" + fname + "','" + lname + "','" + dept +
',' + reason + "','" + advisor + "','" + nowDate + "','" + nowTime + "')";
    String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
connInsert = new MySql.Data.MySqlClient.MySqlConnection(connString);
connInsert.Open();
cmdInsert = new MySql.Data.MySqlClient.MySqlCommand(qryString, connInsert);
try
{
    cmdInsert.ExecuteNonQuery();
}
catch(Exception e)
{
    lblMessage.Text = e.Message; //for debugging...
    retValue = false;
}
connInsert.Close();
sendEmail(dept);
clearForm();
return retValue;
}

private void sendEmail(String dept)
{
    bool notifyAdvisors = false;
    String dept_email = "";
    connectToDB();
    qryString = "select dept_email from department where dept_id =?dept_id and
(dept_email is not null and dept_email <> ") ";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?dept_id", dept);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
    }
}

```



```

while (reader.HasRows && reader.Read())
{
    dept_email = reader["dept_email"].ToString();
    notifyAdvisors = true;
}
if(notifyAdvisors)
{
    SmtplibClient smtp = new SmtplibClient("smtp.gmail.com");
    smtp.EnableSsl = true;
    smtp.Port = 587;
    smtp.Credentials = new
NetworkCredential(System.Configuration.ConfigurationManager.AppSettings["smtp"],
System.Configuration.ConfigurationManager.AppSettings["appPW"]);
    smtp.Send("drdavis2020@gmail.com", dept_email, "Student Queue", "A student has
signed in");
}
}

```

```

protected void btnClearForm_Click(object sender, EventArgs e)
{
    clearForm();
}

```

```

protected void ddlDepartment_SelectedIndexChanged(object sender, EventArgs e)
{
    loadReasons();
}

```

```

protected void gvStudentLogin_RowCommand(object sender,
GridViewCommandEventArgs e)
{
    int index = Convert.ToInt32(e.CommandArgument);
    GridViewRow selectedRow = gvStudentLogin.Rows[index];
    TableCell tcLookupId = selectedRow.Cells[0];
    TableCell tcLookupFirstName = selectedRow.Cells[1];
    TableCell tcLookupLastName = selectedRow.Cells[2];
    TableCell tcLookupPhone = selectedRow.Cells[3];
    txtID.Text = tcLookupId.Text;
    txtPhone.Text = tcLookupPhone.Text;
    txtFirstName.Text = tcLookupFirstName.Text;
    txtLastName.Text = tcLookupLastName.Text;
    String reason = " ";
    if (ddlReason.SelectedItem.Text != "<Select Reason>")
    {
        reason = ddlReason.SelectedItem.Text;
    }
}

```

```

        lblMessage.Text = reason;
    }
    int advisor = 0;
    if (ddlAdvisors.SelectedItem.Text != "<Select Advisor>")
    {
        advisor = Convert.ToInt32(ddlAdvisors.SelectedItem.Value);
    }
    if (addSession(txtID.Text, txtFirstName.Text, txtLastName.Text,
ddlDepartment.SelectedItem.Value, reason, advisor))
    {
        lblMessage.Text = "Entry into queue was successful. Please be patient while you wait
your turn.";
        gvStudentLogin.DataSource = null;
        gvStudentLogin.Visible = false;
    }
    else
    {
        //lblMessage.Text = "Entry into queue was not successful. Please try again or contact
college personnel for help.";
    }
}
}
}
}
}

```

StaffLogin.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="StaffLogin.aspx.cs"
Inherits="DonDavisMSISFinalProject.StaffLogin" %>

```

```

<!DOCTYPE html>

```

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Staff Login</title>
    <link rel="stylesheet" href="StudentQueueStyleSheet.css" />
</head>
<body>
    <form id="frmStaffLogin" runat="server">
        <asp:Image ID="imgLogo" ImageUrl="~/logo.png" runat="server" />
        <h3>Staff Login: </h3>
        <br />
        <div>
            <asp:Label ID="lblDirections" Text="Enter College ID and Password to Log in"
runat="server"/>

```

```

</div>
  <br />
<div>
  <asp:Label ID="lblLogin" Text="ID Number:" runat="server" class="labelClass"/>
  <asp:TextBox ID="txtLogin" runat="server" />
</div>
<br />
<div>
  <asp:Label ID="lblPW" Text="Password:" runat="server" class="labelClass"/>
  <asp:TextBox ID="txtPW" TextMode="Password" runat="server" />
</div>
<br />
<div>
  <asp:Button ID="btnLogin" Text="Log In" runat="server" OnClick="staffLogin" />
</div>
<br />
<div>
  <asp:label ID="lblMessage" Text="" runat="server" />
</div>
</form>
</body>
</html>

```

StaffLogin.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace DonDavisMSISFinalProject
{
  public partial class StaffLogin : System.Web.UI.Page
  {
    MySql.Data.MySqlClient.MySqlConnection conn;
    MySql.Data.MySqlClient.MySqlCommand cmd;
    MySql.Data.MySqlClient.MySqlDataReader reader;

    String qryString;
    String staffId;
    String pw;
    String depts;

```

```

protected void Page_Load(object sender, EventArgs e)
{
    txtLogin.Focus();
}

protected void staffLogin(object sender, EventArgs e)
{
    if (!checkForData())
    {
        lblMessage.Text = "College ID must be entered.";
        return;
    }
    if(userAuthentication())
    {
        Session["staffId"] = staffId;
        Response.Redirect("StaffSession.aspx",true);
    }
}

private bool userAuthentication()
{
    bool retValue = true;
    depts = "";
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    connectToDB();
    qryString = "select sd.dept_id, count(*) knt, s.* from staff s, staff_dept sd where
sd.staff_id = s.staff_id and s.staff_id=?id and s.pw =?pw";
    qryString = qryString + " and effective_date <=?today and (inactive_date >?today or
inactive_date is null or inactive_date =)";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?id", staffId);
    cmd.Parameters.AddWithValue("?pw", pw);
    cmd.Parameters.AddWithValue("?today", nowDate);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
        retValue = false;
    }
    int knt = 1;
    int recs = 0;
    if(reader.HasRows && reader.Read())

```

```

{
    recs = Int32.Parse(reader["knt"].ToString());
    if (recs < 1)
    {
        retValue = false;
        lblMessage.Text = "Invalid user credentials or user does not have an associated
department.";
        txtLogin.Text = "";
        txtPW.Text = "";
        txtLogin.Focus();
    }
}
reader.Close();
if (retValue)
{
    lblMessage.Text = "Woo Hoo.";
    qryString = "select sd.dept_id, s.* from staff s, staff_dept sd where sd.staff_id =
s.staff_id and s.staff_id=?id";
    qryString = qryString + " and effective_date <=?today and (inactive_date >?today or
inactive_date is null or inactive_date =)";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?id", staffId);
    cmd.Parameters.AddWithValue("?today", nowDate);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
        retValue = false;
    }
    if(retValue)
    {
        while (reader.HasRows && reader.Read())
        {
            depts = depts + "" + reader["dept_id"].ToString() + "";
            if (knt < recs)
            {
                depts = depts + ",";
                knt++;
            }
        }
    }
}
Session["depts"] = depts;

```



```

<title>Staff Session</title>
</head>
<body>
  <form id="frmStaffSession" runat="server">
    <asp:Image ID="imgLogo" ImageUrl="~/logo.png" runat="server" />
    <div>
      <asp:Button ID="btnEditDeptReasons" Text="Edit Dept Reasons"
OnClick="btnEditDeptReasons_Click" Enabled="true" runat="server" />
      <asp:Button ID="btnEditStaffDepts" Text="Edit Staff Depts"
OnClick="btnEditStaffDepts_Click" Enabled="true" runat="server" />
    </div>
    <br />
    <div>
      <h3>Staff Session: </h3>
    </div>
    <br />
    <div>
      <asp:Button ID="btnRequery" Text="Reload Queue" runat="server"
OnClick="btnRequery_Click" />
    </div>
    <br />
    <div>
      <asp:Label ID="lblInfo" Text="Choose a student" runat="server" />
    </div>
    <br />
    <div>
      <asp:GridView ID="gvStudents" AutoGenerateColumns="false" runat="server"
OnRowCommand="gvStudents_RowCommand">
        <AlternatingRowStyle BackColor="#99FFCC" />
        <Columns>
          <asp:BoundField DataField="sess_id" HeaderText="Session ID" ReadOnly="true" />
          <asp:BoundField DataField="dept_id" HeaderText="Department" ReadOnly="true" />
          <asp:BoundField DataField="stu_fname" HeaderText="Student First Name"
ReadOnly="true" />
          <asp:BoundField DataField="stu_lname" HeaderText="Student Last Name"
ReadOnly="true" />
          <asp:BoundField DataField="s_date" HeaderText="Session Date" ReadOnly="true" />
          <asp:BoundField DataField="stu_sign_in_time" HeaderText="Sign In Time"
ReadOnly="true" />
          <asp:BoundField DataField="reason" HeaderText="Reason" ReadOnly="true" />
          <asp:buttonfield buttontype="Button" commandname="Select" headertext="Select to
start session" text="Start Session"/>
        </Columns>
      </asp:GridView>
    </div>

```

```

<br />

<div>
  <asp:Label ID="lblNoDataStudents" Text="" runat="server" />
</div>
<br />
<div>
  <asp:Label ID="lblCurrentSession" Text="Choose a student to end session" runat="server"
/>
</div>
<br />
<div>
  <asp:GridView ID="gvCurrentSession" AutoGenerateColumns="false" runat="server"
OnRowCommand="gvCurrentSession_RowCommand"
onrowediting="gvCurrentSession_RowEditing"
  onrowupdating="gvCurrentSession_RowUpdating"
onrowcancelingedit="gvCurrentSession_RowCancelingEdit">
  <AlternatingRowStyle BackColor="#99FFCC" />
  <Columns>
    <asp:BoundField DataField="sess_id" HeaderText="Session ID" ReadOnly="true" />
    <asp:BoundField DataField="dept_id" HeaderText="Department" ReadOnly="true" />
    <asp:BoundField DataField="stu_fname" HeaderText="Student First Name"
ReadOnly="true" />
    <asp:BoundField DataField="stu_lname" HeaderText="Student Last Name"
ReadOnly="true" />
    <asp:BoundField DataField="s_date" HeaderText="Session Date" ReadOnly="true" />
    <asp:BoundField DataField="stu_sign_in_time" HeaderText="Sign In Time"
ReadOnly="true" />
    <asp:BoundField DataField="start_time" HeaderText="Session Start Time"
ReadOnly="true" />
    <asp:BoundField DataField="reason" HeaderText="Reason" ReadOnly="true" />

    <asp:TemplateField HeaderText="Notes" ItemStyle-Width="600px" ItemStyle-
Wrap="true">
      <ItemTemplate>
        <asp:Label ID="lblNotes" runat="server" Text='<%# Bind("notes")
%>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="txtNotes" runat="server" Width="100%"
TextMode="MultiLine" Wrap="True" Text='<%# Bind("notes") %>'></asp:TextBox>
      </EditItemTemplate>

    </asp:TemplateField>
    <asp:TemplateField HeaderText="Edit" ItemStyle-Width="190px">
      <ItemTemplate>

```



```

        <asp:Button ID="btnEdit" runat="server" CausesValidation="False"
            CommandName="Edit" Text="Edit Notes"></asp:Button>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:Button ID="btnUpdate" runat="server" CausesValidation="True"
            CommandName="Update" Text="Update Notes"></asp:Button>
        &nbsp;<asp:Button ID="btnCancel" runat="server" CausesValidation="False"
            CommandName="Cancel" Text="Cancel"></asp:Button>
    </EditItemTemplate>
    <FooterTemplate>
        <asp:Button ID="btnadd" runat="server" Text="Add" CommandName="Add" />
    </FooterTemplate>
</asp:TemplateField>

<asp:buttonfield buttontype="Button" CommandName="EndSession"
headertext="Select to end session" text="End Session"/>

    </Columns>
</asp:GridView>
</div>
<br />
<div>
    <asp:Label ID="lblNoDataCurrentSession" Text="" runat="server" />
</div>
<br />
<div>
    <br />
    <asp:Label ID="lblMessage" Text="" runat="server" />
</div>
</form>
</body>
</html>

```

StaffSession.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace DonDavisMSISFinalProject
{

```

```

public partial class StaffSession : System.Web.UI.Page
{
    MySql.Data.MySqlClient.MySqlConnection conn;
    MySql.Data.MySqlClient.MySqlConnection connUpdate;
    MySql.Data.MySqlClient.MySqlConnection connEnd;
    MySql.Data.MySqlClient.MySqlCommand cmd;
    MySql.Data.MySqlClient.MySqlCommand cmd2;
    MySql.Data.MySqlClient.MySqlCommand cmdUpdate;
    MySql.Data.MySqlClient.MySqlCommand cmdEnd;
    MySql.Data.MySqlClient.MySqlDataAdapter sdaSession;
    MySql.Data.MySqlClient.MySqlDataAdapter sdaSession2;

    String qryString;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            getData();
            loadButtons();
        }
    }

    private void loadButtons()
    {
        MySql.Data.MySqlClient.MySqlCommand cmdKnt;
        MySql.Data.MySqlClient.MySqlDataReader readerKnt;
        qryString = "select count(*) knt from staff_dept where staff_id=?staffId and
upd_dept_reasons = 'Y'";
        cmdKnt = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
        cmdKnt.Parameters.AddWithValue("?staffId", Session["staffId"]);
        readerKnt = cmdKnt.ExecuteReader();
        if (readerKnt.HasRows && readerKnt.Read())
        {
            int knt = Convert.ToInt32(readerKnt.GetString(readerKnt.GetOrdinal("knt")));
            if (knt < 1)
            {
                btnEditDeptReasons.Enabled = false;
                btnEditDeptReasons.Visible = false;
            }
        }
        readerKnt.Close();

        qryString = "select count(*) knt from staff_dept where staff_id=?staffId and upd_this_tbl
= 'Y'";
        cmdKnt = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    }
}

```

```

cmdKnt.Parameters.AddWithValue("?staffId", Session["staffId"]);
readerKnt = cmdKnt.ExecuteReader();
if (readerKnt.HasRows && readerKnt.Read())
{
    int knt = Convert.ToInt32(readerKnt.GetString(readerKnt.GetOrdinal("knt")));
    if (knt < 1)
    {
        btnEditStaffDepts.Enabled = false;
        btnEditStaffDepts.Visible = false;
    }
}
readerKnt.Close();
}

private void connectToDB()
{
    try
    {
        String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
        conn = new MySql.Data.MySqlClient.MySqlConnection(connString);
        conn.Open();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message;
    }
}

private void getData()
{
    connectToDB();
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    String depts = Session["depts"].ToString();
    qryString = "select sess_id, dept_id, stu_fname, stu_lname,
DATE_FORMAT(sess_date, '%m-%d-%Y') as s_date, stu_sign_in_time, reason from session ";
    qryString = qryString + "where sess_date =?today and start_time is null and end_time is
null and dept_id in (" + depts + ")";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?today", nowDate);
    sdaSession = new MySql.Data.MySqlClient.MySqlDataAdapter(cmd);
    DataSet dsSession = new DataSet();
    sdaSession.Fill(dsSession);
    if(dsSession.Tables[0].Rows.Count > 0)
    {
        gvStudents.DataSource = dsSession;
    }
}

```

```

        gvStudents.DataBind();
        lblInfo.Text = "Choose a student";
        lblNoDataStudents.Text = "";
    }
    else
    {
        gvStudents.DataSource = null;
        gvStudents.DataBind();
        lblInfo.Text = "";
        lblNoDataStudents.Text = "No students in Queue.";
    }
    String staffId = Session["staffId"].ToString();
    qryString = "select sess_id, dept_id, stu_fname, stu_lname,
DATE_FORMAT(sess_date,'%m-%d-%Y') as s_date, stu_sign_in_time, start_time, reason,
notes from session ";
    qryString = qryString + "where staff_id=?staff_id and start_time is not null and end_time
is null";
    cmd2 = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd2.Parameters.AddWithValue("?staff_id", staffId);
    sdaSession2 = new MySql.Data.MySqlClient.MySqlDataAdapter(cmd2);
    DataSet dsSession2 = new DataSet();
    sdaSession2.Fill(dsSession2);
    if (dsSession2.Tables[0].Rows.Count > 0)
    {
        gvCurrentSession.DataSource = dsSession2;
        gvCurrentSession.DataBind();
        lblCurrentSession.Text = "Choose a student to end session";
        lblNoDataCurrentSession.Text = "";
    }
    else
    {
        gvCurrentSession.DataSource = null;
        gvCurrentSession.DataBind();
        lblCurrentSession.Text = "";
        lblNoDataCurrentSession.Text = "No current sessions to view.";
    }
}

private bool updateSession(String SessionID)
{
    bool retValue = true;
    String staffId = Session["staffId"].ToString();
    if(String.IsNullOrEmpty(staffId))
    {
        lblMessage.Text = "Staff ID not carried forward from Login.";
        return false;
    }
}

```

```

    }
    String nowTime = DateTime.Now.ToString("HH:mm:ss");
    qryString = "update session set start_time =?time, staff_id =?staffId where sess_id =
?sess_id ";
    String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
    connUpdate = new MySql.Data.MySqlClient.MySqlConnection(connString);
    connUpdate.Open();
    cmdUpdate = new MySql.Data.MySqlClient.MySqlCommand(qryString, connUpdate);
    cmdUpdate.Parameters.AddWithValue("?sess_id", SessionID);
    cmdUpdate.Parameters.AddWithValue("?time", nowTime);
    cmdUpdate.Parameters.AddWithValue("?staffId", staffId);

    try
    {
        cmdUpdate.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message; //for debugging...
        retVal = false;
    }
    connUpdate.Close();
    return retVal;
}

private bool endSession(String SessionID)
{
    bool retVal = true;
    String staffId = Session["staffId"].ToString();
    if (String.IsNullOrEmpty(staffId))
    {
        lblMessage.Text = "Staff ID not carried forward from Login.";
        return false;
    }
    String nowTime = DateTime.Now.ToString("HH:mm:ss");
    qryString = "update session set end_time =?time where sess_id = ?sess_id ";
    String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
    connEnd = new MySql.Data.MySqlClient.MySqlConnection(connString);
    connEnd.Open();
    cmdEnd = new MySql.Data.MySqlClient.MySqlCommand(qryString, connEnd);
    cmdEnd.Parameters.AddWithValue("?sess_id", SessionID);
    cmdEnd.Parameters.AddWithValue("?time", nowTime);

```

```

    try
    {
        cmdEnd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message; //for debugging...
        retValue = false;
    }
    connEnd.Close();
    return retValue;
}
private bool updateNotes(String notes, String sessId)
{
    bool retValue = true;
    qryString = "update session set notes =?notes where sess_id = ?sess_id ";
    String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
    connUpdate = new MySql.Data.MySqlClient.MySqlConnection(connString);
    connUpdate.Open();
    cmdUpdate = new MySql.Data.MySqlClient.MySqlCommand(qryString, connUpdate);
    cmdUpdate.Parameters.AddWithValue("?sess_id", sessId);
    cmdUpdate.Parameters.AddWithValue("?notes", notes);

    try
    {
        cmdUpdate.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " Error in updateNotes"; //for debugging...
        retValue = false;
    }
    connUpdate.Close();
    return retValue;
}

protected void gvStudents_RowCommand(object sender, GridViewCommandEventArgs e)
{
    int index = Convert.ToInt32(e.CommandArgument);
    GridViewRow selectedRow = gvStudents.Rows[index];
    TableCell tcSessionId = selectedRow.Cells[0];
    string sessionId = tcSessionId.Text;
    lblMessage.Text = "You selected " + sessionId + ".";
}

```

```

    if(updateSession(sessionId))
    {
        getData();
    }
}

```

```

protected void gvCurrentSession_RowCommand(object sender,
GridViewCommandEventArgs e)
{
    if (e.CommandName == "EndSession")
    {
        int index = Convert.ToInt32(e.CommandArgument);
        GridViewRow selectedRow = gvCurrentSession.Rows[index];
        TableCell tcSessionId = selectedRow.Cells[0];
        string sessionId = tcSessionId.Text;
        lblMessage.Text = "You selected " + sessionId + ".";
        if (endSession(sessionId))
        {
            getData();
        }
    }
}

```

```

protected void gvCurrentSession_RowEditing(object sender, GridViewEditEventArgs e)
{
    lblMessage.Text = "Row Editing";
    gvCurrentSession.EditIndex = e.NewEditIndex;
    getData();
    gvCurrentSession.Columns[8].ItemStyle.Width = 600;
    gvCurrentSession.Columns[8].ItemStyle.Height = 60;
}

```

```

protected void gvCurrentSession_RowUpdating(object sender, GridViewUpdateEventArgs
e)
{
    try
    {
        GridViewRow row = (GridViewRow)gvCurrentSession.Rows[e.RowIndex];
        TableCell tcSessionId = row.Cells[0];
        string sessionId = tcSessionId.Text;
        TextBox txtNotes =
        (TextBox)gvCurrentSession.Rows[e.RowIndex].FindControl("txtNotes");
        string sessNotes = txtNotes.Text;

        if (updateNotes(sessNotes, sessionId))
        {

```

```

        lblMessage.Text = "Record is updated successfully.";
    }
    gvCurrentSession.EditIndex = -1;
    getData();
}
catch(Exception a)
{
    lblMessage.Text = a.Message + " Error in RowUpdating"; //for debugging...
}
}

protected void gvCurrentSession_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
{
    gvCurrentSession.EditIndex = -1;
    getData();
}

protected void btnEditDeptReasons_Click(object sender, EventArgs e)
{
    Response.Redirect("DepartmentReasons.aspx", true);
}

protected void btnEditStaffDepts_Click(object sender, EventArgs e)
{
    Response.Redirect("StaffDepartments.aspx", true);
}

protected void btnRequery_Click(object sender, EventArgs e)
{
    getData();
}
}
}

```

StaffDepartments.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="StaffDepartments.aspx.cs"
Inherits="DonDavisMSISFinalProject.StaffDepartments" %>

```

```

<!DOCTYPE html>

```

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Staff Departments</title>

```



```
<asp:Label ID="lblMessage" Text="" runat="server" />
</div>
</form>
</body>
</html>
```

StaffDepartments.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace DonDavisMSISFinalProject
{
    public partial class StaffDepartments : System.Web.UI.Page
    {
        MySql.Data.MySqlClient.MySqlConnection conn;
        MySql.Data.MySqlClient.MySqlConnection connUpdate;
        MySql.Data.MySqlClient.MySqlConnection connInsert;
        MySql.Data.MySqlClient.MySqlCommand cmd;
        MySql.Data.MySqlClient.MySqlCommand cmdUpdate;
        MySql.Data.MySqlClient.MySqlCommand cmdInsert;
        MySql.Data.MySqlClient.MySqlDataReader reader;
        MySql.Data.MySqlClient.MySqlDataAdapter sdaStaffDept;

        String qryString;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                getData();
                loadStaff();
                loadDepartments();
            }
        }

        private void connectToDB()
        {
            try
```

```

    {
        String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
        conn = new MySql.Data.MySqlClient.MySqlConnection(connString);
        conn.Open();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message;
    }
}

private void loadStaff()
{
    connectToDB();
    qryString = "select s.staff_id, CONCAT(p.first_name, ' ', p.last_name) as name from staff
s, person p where p.id = s.staff_id";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    try
    {
        reader = cmd.ExecuteReader();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " | " + qryString;
    }
    ListItem newItem = new ListItem();
    newItem.Text = "<Select Staff>";
    newItem.Value = "0";
    ddlStaff.Items.Add(newItem);
    while (reader.HasRows && reader.Read())
    {
        newItem = new ListItem();
        newItem.Text = reader["name"].ToString();
        newItem.Value = reader["staff_id"].ToString();
        ddlStaff.Items.Add(newItem);
    }
    reader.Close();
}

private void loadDepartments()
{
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    connectToDB();

```

```

String depts = Session["depts"].ToString();
qryString = "select dept_id, dept_name from department where dept_id in (" + depts + ")
and effective_date <= " + nowDate + " and (inactive_date is null or inactive_date = ' ' or
inactive_date > " + nowDate + ")";
cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
try
{
    reader = cmd.ExecuteReader();
}
catch (Exception e)
{
    lblMessage.Text = e.Message + " | " + qryString;
}
ListItem newItem = new ListItem();
newItem.Text = "<Select Department>";
newItem.Value = "0";
ddlDepartment.Items.Add(newItem);
while (reader.HasRows && reader.Read())
{
    newItem = new ListItem();
    newItem.Text = reader["dept_name"].ToString();
    newItem.Value = reader["dept_id"].ToString();
    ddlDepartment.Items.Add(newItem);
}
reader.Close();
}

private void getData()
{
    connectToDB();
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    String depts = Session["depts"].ToString(); // "ADMISSIONS','STUACCTS";
    qryString = "select a.first_name, a.last_name, b.staff_id, b.dept_id, b.upd_this_tbl,
b.upd_dept_reasons, DATE_FORMAT(b.effective_date, '%m-%d-%Y') as effective_date ";
    qryString = qryString + "from person a, staff_dept b where a.id = b.staff_id and b.dept_id
in (" + depts + ") and effective_date <=?today and ";
    qryString = qryString + "(inactive_date >?today or inactive_date is null or inactive_date
= ")";
    cmd = new MySql.Data.MySqlClient.MySqlCommand(qryString, conn);
    cmd.Parameters.AddWithValue("?today", nowDate);
    sdaStaffDept = new MySql.Data.MySqlClient.MySqlDataAdapter(cmd);
    DataSet dsStaffDept = new DataSet();
    sdaStaffDept.Fill(dsStaffDept);
    if (dsStaffDept.Tables[0].Rows.Count > 0)
    {

```

```

protected void gvStaffDepts_RowCommand(object sender, GridViewCommandEventArgs
e)
{
    //lblMessage.Text = e.CommandName;
    if (e.CommandName == "InactivateRecord")
    {
        lblMessage.Text = "Inside Row Command";
        int index = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = gvStaffDepts.Rows[index];
        TableCell tcDeptId = row.Cells[3];
        string deptId = tcDeptId.Text;
        TableCell tcStaffId = row.Cells[0];
        string staff_id = tcStaffId.Text;
        String inactiveDate = DateTime.Now.ToString("yyyy-MM-dd");

        if (updateStaffDept(inactiveDate, deptId, staff_id))
        {
            lblMessage.Text = "Record is updated successfully.";
            getData();
        }
    }
    else if (e.CommandName == "Add")
    {
        lblMessage.Text = "Inside Row Command for Add";
    }
}

private bool updateStaffDept(String inactiveDate, String deptId, String staff_id)
{
    bool retValue = true;
    String parms = inactiveDate + " " + deptId + " " + staff_id;
    lblMessage.Text = parms;
    qryString = "update staff_dept set inactive_date =?inactiveDate where dept_id = ?dept_id
and staff_id =?staff_id";
    String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
    connUpdate = new MySql.Data.MySqlClient.MySqlConnection(connString);
    connUpdate.Open();
    cmdUpdate = new MySql.Data.MySqlClient.MySqlCommand(qryString, connUpdate);
    cmdUpdate.Parameters.AddWithValue("?dept_id", deptId);
    cmdUpdate.Parameters.AddWithValue("?staff_id", staff_id);
    cmdUpdate.Parameters.AddWithValue("?inactiveDate", inactiveDate);

    try

```

```

    {
        cmdUpdate.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message + " Error in updateStaffDept"; //for debugging...
        retVal = false;
    }
    connUpdate.Close();
    return retVal;
}
}
}

```

DepartmentReasons.aspx

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="DepartmentReasons.aspx.cs"
Inherits="DonDavisMSISFinalProject.DepartmentReasons" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Department Reasons</title>
    <link rel="stylesheet" href="StudentQueueStyleSheet.css" />
    <script type="text/javascript">
        function ToUpper(ctrl)
        {
            var t = ctrl.value;
            ctrl.value = t.toUpperCase();
        }
    </script>
</head>
<body>
    <form id="frmDepartmentReasons" runat="server">
        <asp:Image ID="imgLogo" ImageUrl="~/logo.png" runat="server" />
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            </asp:ScriptManager>
        <asp:GridView ID="gvDeptReasons" AutoGenerateColumns="false" runat="server"
OnRowCommand="gvDeptReasons_RowCommand"
OnRowEditing="gvDeptReasons_RowEditing"
    OnRowUpdating="gvDeptReasons_RowUpdating"
OnRowCancelingEdit="gvDeptReasons_RowCancelingEdit"

```

```

OnRowCreated="gvDeptReasons_RowCreated"
OnRowDataBound="gvDeptReasons_RowDataBound">
  <AlternatingRowStyle BackColor="#99FFCC" />
  <Columns>
    <asp:BoundField DataField="dept_id" HeaderText="Department" ReadOnly="true" />
    <asp:BoundField DataField="reason" HeaderText="Reason" ReadOnly="true" />
    <asp:BoundField DataField="effective_date" HeaderText="Effective Date"
ReadOnly="true" />
    <asp:buttonfield buttontype="Button" CommandName="InactivateReason"
headertext="Select to set dept/reason inactive" text="Inactivate Dept. Reason"/>

  </Columns>
</asp:GridView>
<asp:ValidationSummary runat="server" ShowMessageBox="true" ShowSummary="false"
/>
<br />
<br />
<h3>Add a row:</h3>
<div>
  <asp:Label ID="lblDepartment" Text="Department:" runat="server"
CssClass="labelClass"/>
  <asp:DropDownList ID="ddlDepartment" runat="server" AutoPostBack="true"
OnSelectedIndexChanged="ddlDepartment_SelectedIndexChanged"/>
</div>
<br />

<div>
  <asp:Label ID="lblAddReason" Text="Reason:" runat="server" CssClass="labelClass"/>
  <asp:TextBox ID="txtAddReason" runat="server" AutoComplete="off" CssClass="upper-
case" CausesValidation="true" onKeyUp="ToUpper(this)" MaxLength="10"/>
  <asp:RequiredFieldValidator ID="RequiredFieldValidator1 "
    runat="server" ControlToValidate="txtAddReason"
    Display="Dynamic" ForeColor="red"
    ErrorMessage="You must enter a reason for the new record.">
  * </asp:RequiredFieldValidator>
</div>
<br />
<div>
  <asp:Label ID="lblNewEffDate" Text="Effective Date:" runat="server"
CssClass="labelClass" />
  <asp:TextBox ID="NewEffDate" runat="server"></asp:TextBox>

  <ajaxtoolkit:CalendarExtender ID="CalendarExtender1" runat="server"
    TargetControlID="NewEffDate" PopupButtonID="NewEffDate"
Format="yyyy-MM-dd">

```

```

</ajaxtoolkit:CalendarExtender>
<asp:RequiredFieldValidator ID="RequiredFieldValidator4"
    runat="server" ControlToValidate="NewEffDate"
    Display="Dynamic" ForeColor="red"
    ErrorMessage="You must enter an effective date for the new record.">
* </asp:RequiredFieldValidator>
</div>
<br />
<div>
    <asp:Label ID="lblAddIns" Text="To add a row, add Department, Reason, and Effective
Date and then click 'Add a Row' button." runat="server" />
    <br /><br />
    <asp:Button ID="btnAddRow" runat="server" Text="Add a Row"
CommandName="AddRow" OnClick="btnAddRow_Click" />
</div>
<br />
<div>
    <br />
    <asp:Label ID="lblMessage" Text="" runat="server" />
</div>
</form>
</body>
</html>

```

DepartmentReasons.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace DonDavisMSISFinalProject
{
    public partial class DepartmentReasons : System.Web.UI.Page
    {
        MySql.Data.MySqlClient.MySqlConnection conn;
        MySql.Data.MySqlClient.MySqlConnection connUpdate;
        MySql.Data.MySqlClient.MySqlConnection connInsert;
        MySql.Data.MySqlClient.MySqlCommand cmd;
        MySql.Data.MySqlClient.MySqlCommand cmdUpdate;
        MySql.Data.MySqlClient.MySqlCommand cmdInsert;
        MySql.Data.MySqlClient.MySqlDataReader reader;
    }
}

```

```
MySql.Data.MySqlClient.MySqlDataAdapter sdaStaffDept;
```

```
String qryString;
```

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    if (!Page.IsPostBack)
    {
        getData();
        loadDepartments();
    }
}
```

```
protected void gvDeptReasons_RowCommand(object sender,
GridViewCommandEventArgs e)
```

```
{
    if(e.CommandName=="InactivateReason")
    {
        lblMessage.Text = "Inside Row Command";
        int index = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = gvDeptReasons.Rows[index];
        TableCell tcDeptId = row.Cells[0];
        string deptId = tcDeptId.Text;
        TableCell tcReason = row.Cells[1];
        string reason = tcReason.Text;
        String inactiveDate = DateTime.Now.ToString("yyyy-MM-dd");
        if (updateDeptReason(inactiveDate, deptId, reason))
        {
            lblMessage.Text = "Record is updated successfully.";
            getData();
        }
    }
    else if(e.CommandName=="Add")
    {
        lblMessage.Text = "Inside Row Command for Add";
    }
}
```

```
private bool updateDeptReason(String inactiveDate, String deptId, String reason)
```

```
{
    bool retValue = true;
    String parms = inactiveDate + " " + deptId + " " + reason;
    lblMessage.Text = parms;
}
```



```

        qryString = "update dept_reason set inactive_date =?inactiveDate where dept_id =
?dept_id and reason =?reason";
        String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
        connUpdate = new MySql.Data.MySqlClient.MySqlConnection(connString);
        connUpdate.Open();
        cmdUpdate = new MySql.Data.MySqlClient.MySqlCommand(qryString, connUpdate);
        cmdUpdate.Parameters.AddWithValue("?dept_id", deptId);
        cmdUpdate.Parameters.AddWithValue("?reason", reason);
        cmdUpdate.Parameters.AddWithValue("?inactiveDate", inactiveDate);

        try
        {
            cmdUpdate.ExecuteNonQuery();
        }
        catch (Exception e)
        {
            lblMessage.Text = e.Message + " Error in updateDeptReason"; //for debugging...
            retVal = false;
        }
        connUpdate.Close();
        return retVal;
    }

    private void connectToDB()
    {
        try
        {
            String connString =
System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
        }
        catch (Exception e)
        {
            lblMessage.Text = e.Message;
        }
    }

    private void getData()
    {
        connectToDB();
        String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
        String depts = Session["depts"].ToString(); //"ADMISSIONS','STUACCTS";
    }

```

```

        newItem = new ListItem();
        newItem.Text = reader["dept_name"].ToString();
        newItem.Value = reader["dept_id"].ToString();
        ddlDepartment.Items.Add(newItem);
    }
    reader.Close();
}

private bool addDeptReason(String deptId, String reason, String effectiveDate)
{
    bool retValue = true;

    qryString = "insert into dept_reason (dept_id, reason, effective_date) values (" + deptId +
    ", " + reason + ", " + effectiveDate + ")";
    String connString =
    System.Configuration.ConfigurationManager.ConnectionStrings["WebAppConnString"].ToString();
    connInsert = new MySql.Data.MySqlClient.MySqlConnection(connString);
    connInsert.Open();
    cmdInsert = new MySql.Data.MySqlClient.MySqlCommand(qryString, connInsert);
    try
    {
        cmdInsert.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        lblMessage.Text = e.Message; //for debugging...
        retValue = false;
    }
    connInsert.Close();
    return retValue;
}

protected void btnAddRow_Click(object sender, EventArgs e)
{
    //Add a row
    String nowDate = DateTime.Now.ToString("yyyy-MM-dd");
    String nowTime = DateTime.Now.ToString("HH:mm:ss");
    addDeptReason(ddlDepartment.Text, txtAddReason.Text, NewEffDate.Text);
    getData();
}
}
}

```

Web.config

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5.1" />
    <httpRuntime targetFramework="4.5.1" />
    <pages>
      <controls>
        <add tagPrefix="ajaxToolkit" assembly="AjaxControlToolkit"
namespace="AjaxControlToolkit"></add>
      </controls>
    </pages>
  </system.web>

  <connectionStrings>
    <add name="WebAppConnString"
      connectionString="server=127.0.0.1;User
      ID=signin_app;Password=signin;Database=drdavis;"
      providerName="MySQL.Data.MySqlClient" />
  </connectionStrings>

  <appSettings>
    <add key="smtp" value="drdavis2020@gmail.com"/>
    <add key="appPW" value="fvngarzfvguptjua"/>
    <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
  </appSettings>
</configuration>
```

11-08-20