Orthopaedic Exercise Feedback System
Final Report
Michigan State University
ECE 480

Joe Fabbo
Michael Juricny
Rafee Mahmud
Erica Ramsey
Glen Simon

December 7, 2016

Executive Summary

During shoulder rehabilitation, doctors and therapists cannot guarantee that patients are completing the proper care while on their own. While at home or unattended in a therapeutic establishment, some patients have a tendency to skip routines and cheat the system, which can make therapy all for naught. Also, if the therapist is not assisting during every exercise, the patient can easily be using improper form. By doing so, incorrect muscles and joints are being worked, which can have a negative impact on the result of therapy. In order to solve this issue, Design Team One of MSU's ECE 480 class has developed a control system to force correct progress and deliver feedback to the doctors and therapists. By integrating multiple hardware components, the new machine has the capability to halt operation if improper form is sensed. Through software applications, the doctors and therapists will be able to assign customized exercises based on each patient's needs and the patient will have an interactive screen at the machine which shows these assigned custom exercises. After a therapy routine has been completed, a progress report will be sent automatically to the user who assigned the exercises. Thereby, guaranteeing that the routines have been done correctly and completely.

Acknowledgment

We would like to express our great appreciation to our generous sponsors, Panther Global Technologies and Great Lakes Controls & Engineering, for the monetary gift which allowed us to perform the designated project. We are thankful for the guidance received from them, and the diligent work that they performed throughout the semester to integrate our pieces together. Also deserving a sincere recognition is that of our guidance staff, Dr.'s McGough, Albrecht, and Udpa, as well as the ECE Department, all of whom provided us the resources to work on our project.

Contents

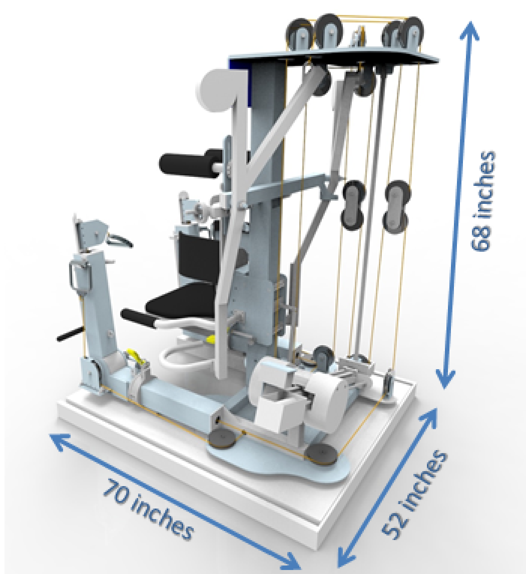
Chapter 1

1.1 Introduction and Background

ECE 480's Design Team One has created a feedback system for a shoulder exercise machine. Great Lakes Controls and Engineering, and Panther Global Technologies have posed the issue of therapists lacking the ability to ensure that patients are correctly performing routines, and also to monitor detailed progress. While the Design Team created the feedback system, the sponsors also worked to create the custom-built mechanical portion to the machine. They have requested an online application, available to therapists, to prescribe designated workout plans and to view completed reports. Also requested was a touchscreen device in which the patients will operate to fetch the individual's prescribed routine, linked with the website's information via a secure database. In order to monitor the progress from the machine and ensure correct form, several hardware components are incorporated and integrated into the overall design.

In today's therapeutic practices, it is uncommon to have shoulder exercises closely monitored in real-time without the need of a therapist standing by, constantly observing. Instead, therapists must rely on faulty methods, such as assigning take-home exercises in which they are forced to assume that it is being completed. With this method, there is no guarantee that the patient is improving his or her shoulder. The current practice severely limits positive results, as there is incredible room for error. The Design Team's approach of creating a controlled feedback system will dissipate the worry of unknown results. Panther Global Technologies had developed a first generation prototype, which allowed for four rotator cuff exercises from a seated position. The team's new system has been added onto a second generation prototype, designed to allow for an extra five shoulder-strengthening exercises, sitting or standing. By incorporating a feedback system for four rotator cuff and five shoulder-strengthening exercises, the marketability of the exercise machine can be greatly increased. Assuming the project is successful in its application, it can significantly improve the statistics of therapy outcome.

Figure 1.1: Shoulder Therapy Machine



(a) References [16]

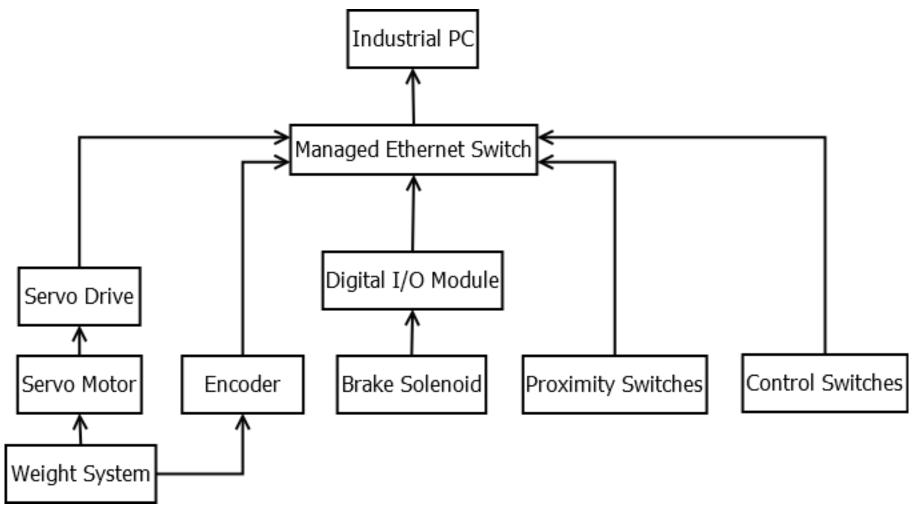
Chapter 2

2.1 Solution Space

The team decomposed the problem into two major subsystems to integrate together near the end of the design stages: hardware components to attach onto the machine, and software applications to loop together the monitoring capability. As this feedback system required a wide scope, it was particularly important to work on both subsystems concurrently. Refer to Appendix 7.2 for the FAST diagram.

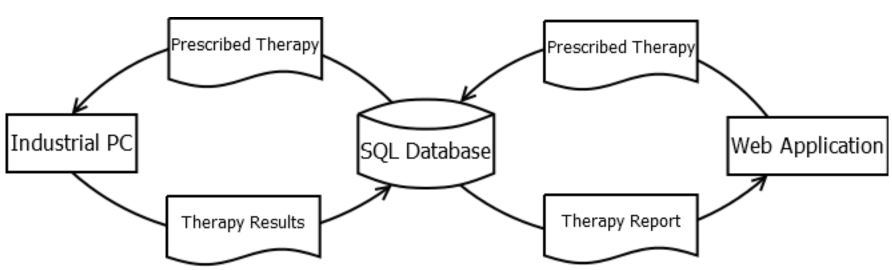
In general, the hardware subsystem's main goal was to provide a monitoring and reporting system, to keep the patient from incorrect form and to communicate with all components. The halting of incorrect form is implemented through proximity sensors and a brake solenoid. When applicable, depending on the need per exercise, the brake is triggered by the input of the proximity sensors. Furthermore, the adjustable weight of the machine is controlled by a servo drive, servo motor, and mechanical weight system. The patient's progress (reps and sets) is also necessary to track; this is fulfilled by an encoder. In order to tie in the input and output communications, an industrial PC, IP switches, and digital I/O module are utilized.

Figure 2.1: Hardware Diagram



In continuation, the desired functions of the software subsystem are to report and monitor the services, as well as provide a hardware interface. An online website allows for creation of new patients, submission of individual and custom reports, and reception of completed patient reports. The data that includes all patient and therapist information is stored in a secure SQL database. To provide an interface to the shoulder exercise machine, an on-machine application is available to the patients.

Figure 2.2: Software Diagram



In regards to the specific approach chosen, the sponsors had laid out a finely-detailed approach to follow, including the specific requirements for physical components and exactly what the software applications should display. Therefore, this team project was slightly abnormal in that the conceptual design was clearly chosen without desire from the sponsors to stray. It was important to follow through with the requested design, as Panther Global was custom-building the physical exercise machine during the semester with the understanding that the team would deliver exactly what was needed to form a functioning exercise system. In continuation, the specific approach of how the project was created is a new concept since general exercise machines are mass-produced without a feedback system. A great deal of thought and consideration has been placed into the project to combine a custom machine with new feedback ideas. Evidence of the system being able to work can stem from the fact that once the entire feedback system is able to communicate, the goal is fulfilled of giving therapists and doctors an incredibly greater method to monitor patients.

2.2 Budget

Design Team One was allotted a budget from the sponsors of $5,000. After thorough consideration of the necessary hardware and software components, the complete collection of materials stayed within the budget. Below is the bill of materials needed to implement the prototype. In order to put the design into production, it must also be reminded that the below bill is strictly towards the feedback system, completely excluding the physical structure of the mechanical machine. Therefore, it would cost under $4,500 to reproduce the specific feedback system, given that the physical machine is a separate entity.

Table 2.1: Project Budget

Part Description	**Part Number**	**Quantity**	**Unit Price**	**Purpose**
Panel PC [1]	CV-112-P1001	1	$1,247.00	User Interface and control hub
Ethernet Switch [2]	SE-SW8M	1	$595.00	I/O hub
Proximity Sensor [3]	AES-AN-1A	2	$16.00	Sensors to ensure correct arm location
Brake Solenoid [4]	TP12X19-C	1	$29.91	Disengage/engage machine
Ice Cube Control Relay [5]	781-1C-24D	1	$4.50	Switching for the solenoid
Relay Socket [6]	781-1C-SKT	1	$4.00	Connector
24 V DC Power Supply [7]	PSP24-240S	1	$180.00	Power for various components
Absolute Encoder [8]	AFM60A	1	$927.00	Track position of weight system
Encoder Accessories	6025901, 6048245	1	$131.00	Communication between encoder and control hub
24 V DC Power Supply (stepper) [9]	PSB24-240	1	$115.00	Power for stepper motor
Stepper Motor w/ Encoder [10]	HT23-603D-ZAA	1	$245.00	Adjusts weight
Stepper's Encoder Extension Cable [11]	3004-195-10	1	$42.00	Connection to Ethernet switch for encoder
Stepper Drive [12]	ST10-Q-EE	1	$726.00	Controls stepper motor
Brainbox Ethernet to Digital I/O [13]	ED-588	1	$197.77	Connect components to IPC
Database [14]	Single Database - S0	1	$15.03/ month	Store patient information
Total (w/out monthly cost)			$4,476.18	

(a) Please refer to the references sections in chapter 7.2 for itemized list of sources for each component

2.3 Gantt Chart

During week 4, a Gantt chart was created which presented a documented plan of the team's projected design approach. Due to changes throughout the semester, the chart was immensely edited to reflect the team's plan that was carried out. See figures 2.3 and 2.4 for comparisons between the original and final Gantt chart schedules and time lines.

Figure 2.3: Project Gantt Charts

TIMELINE

Start Fri 9/16/16 | Sep 18, '16 | Sep 25, '16 | Oct 2, '16

GANTT CHART

	Task Name	Durat	Start	Finish	Prede
1	**Intial Meeting and Project Description**	**5 days**	**Fri 9/16/16**	**Thu 9/22/16**	
2	Project meeting with advisor	1 day	Fri 9/16/16	Fri 9/16/16	
3	Project meeting with sponsor	1 day	Thu 9/22/16	Thu 9/22/16	
4	Project definition milestone	0 days	Thu 9/22/16	Thu 9/22/16	
5	Research and order parts	5 days	Fri 9/23/16	Thu 9/29/16	3
6	**Initial Setup**	**10 days**	**Fri 9/30/16**	**Thu 10/13/1(**	
7	Basic set up of industrial PC (Beckoff) to use for all the other testing	5 days	Fri 9/30/16	Thu 10/6/16	5
8	Set up other components and program interface between PC and components	5 days	Fri 10/7/16	Thu 10/13/16	7
9	**Software Setup**	**20 days**	**Fri 10/14/16**	**Thu 11/10/1(**	
10	Create APP for user interface on PC	5 days	Fri 10/14/16	Thu 10/20/16	8
11	Create set up mode for device. Make sure range of motion is set properly and exercises are displayed.	5 days	Fri 10/21/16	Thu 10/27/16	10
12	Create manual operation mode	5 days	Fri 10/28/16	Thu 11/3/16	11
13	Create run mode and make sure alarms are working properly and that exercises get "grayed out". Be able to track data.	5 days	Fri 11/4/16	Thu 11/10/16	12
14	Concurrently create secure database. Must have secure login for patients and doctors.	15 days	Fri 9/23/16	Thu 10/13/16	3
15	Figure physical layout and placement of components including wires.	5 days	Fri 11/11/16	Thu 11/17/16	13
16	Set up data transfer in between industrial PC and database	5 days	Fri 11/18/16	Thu 11/24/16	15
17	**Finalization**	**10 days**	**Fri 11/25/16**	**Thu 12/8/16**	
18	Extra features and optimization	5 days	Fri 11/25/16	Thu 12/1/16	16
19	Final presentation and document prep	5 days	Fri 12/2/16	Thu 12/8/16	18

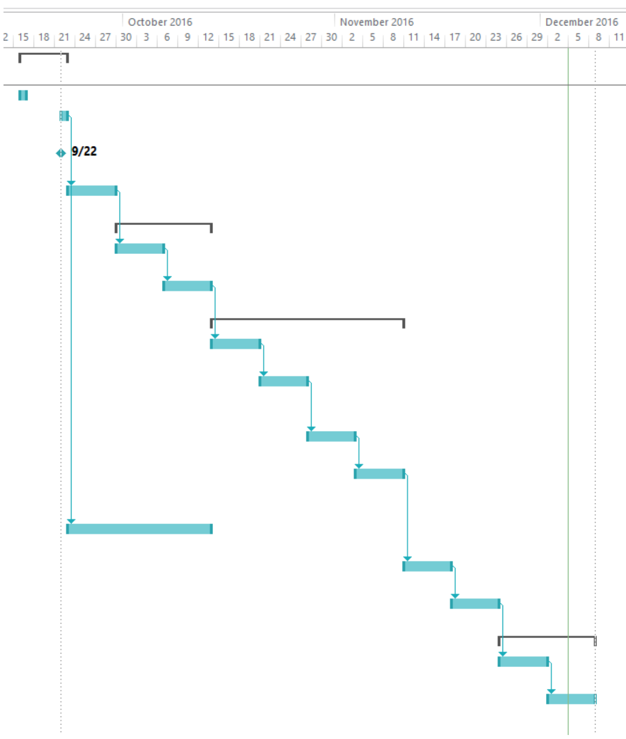
(a) Week Four Gantt Chart

GANTT CHART

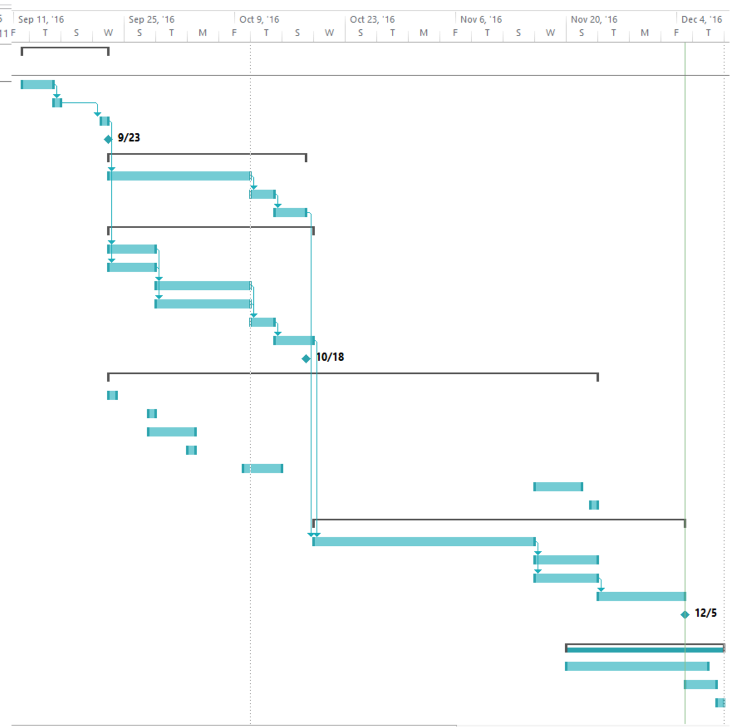
	Task Name	Durati	Start	Finish	Predecessors
1	**Initial Meetings and Brainstorming**	**9 days**	**Mon 9/12/16**	**Fri 9/23/16**	
2	First meeting with team	4 days	Mon 9/12/16	Thu 9/15/16	
3	First meeting with facilitator	1 day	Fri 9/16/16	Fri 9/16/16	2
4	First meeting with sponsor	1 day	Thu 9/22/16	Thu 9/22/16	3
5	Project definition milestone	0 days	Fri 9/23/16	Fri 9/23/16	
6	**Hardware Components**	**17 days**	**Fri 9/23/16**	**Mon 10/17/16**	
7	Discuss hardware options with teammates	12 days	Fri 9/23/16	Mon 10/10/16	4
8	Confirm choices with sponsor	3 days	Tue 10/11/16	Thu 10/13/16	7
9	Place orders for components	2 days	Fri 10/14/16	Mon 10/17/16	8
10	**Software Components**	**18 days**	**Fri 9/23/16**	**Tue 10/18/16**	
11	Discuss available programs for both apps	4 days	Fri 9/23/16	Wed 9/28/16	4
12	Discuss secure database options	4 days	Fri 9/23/16	Wed 9/28/16	4
13	Create basic patient/therapist apps	8 days	Thu 9/29/16	Mon 10/10/16	11
14	Start beginning layout for database	8 days	Thu 9/29/16	Mon 10/10/16	12
15	Incorporate database into applications	3 days	Tue 10/11/16	Thu 10/13/16	13,14
16	Create patient reports	3 days	Fri 10/14/16	Tue 10/18/16	15
17	Have working, basic applications	0 days	Tue 10/18/16	Tue 10/18/16	
18	**Documentation/Presentations**	**44 days**	**Fri 9/23/16**	**Wed 11/23/16**	
19	Gantt chart first draft	1 day	Fri 9/23/16	Fri 9/23/16	
20	Engineering Design Review presentation	1 day	Wed 9/28/16	Wed 9/28/16	
21	FAST diagram	4 days	Wed 9/28/16	Mon 10/3/16	
22	Proposal presentation	1 day	Mon 10/3/16	Mon 10/3/16	
23	Final proposal document	5 days	Mon 10/10/16	Fri 10/14/16	
24	Business canvas	4 days	Wed 11/16/16	Mon 11/21/16	
25	Demonstration of prototype	1 day	Wed 11/23/16	Wed 11/23/16	
26	**Prototyping**	**33 days**	**Wed 10/19/16**	**Mon 12/5/16**	
27	Begin hardware/software integration	20 days	Wed 10/19/16	Tue 11/15/16	9,16
28	Test (adjust) application modes	6 days	Wed 11/16/16	Wed 11/23/16	27
29	Determine physical layout and placement	6 days	Wed 11/16/16	Wed 11/23/16	27
30	Setup Industrial PC / database transfer	8 days	Thu 11/24/16	Sun 12/4/16	29
31	Have interactive, communicating control system	0 days	Mon 12/5/16	Mon 12/5/16	
32	**Finalization**	**16 days**	**Sun 11/20/16**	**Fri 12/9/16**	
33	Final Report	14 days	Sun 11/20/16	Wed 12/7/16	
34	Optimizations	4 days	Mon 12/5/16	Thu 12/8/16	
35	Design Day	1 day	Fri 12/9/16	Fri 12/9/16	

(b) Final Gantt Chart

Figure 2.4: Project Gantt Charts Time Line



(a) Week Four Gantt Chart Time Line



(b) Final Gantt Chart Time Line

Chapter 3

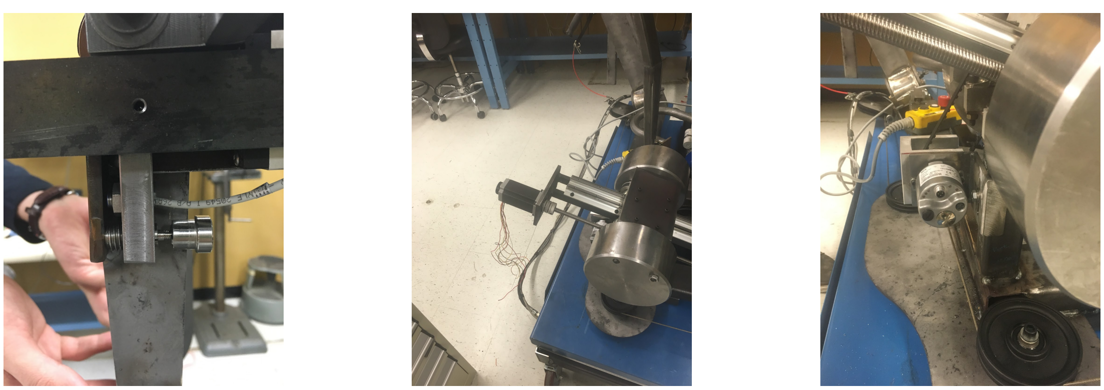
3.1 Hardware Design Requirements

An array of hardware requirements were laid out to the team at the beginning of the design process. A main need was that of a brake system to halt a patient if incorrect movement was sensed. Also, to adjust the weight on the pulleys, a motor had to be implemented. Another requirement was to be able to track the patient's range of motion that is prescribed by his or her therapist. The system needed to have a device that could be used as a display for patients at the machine, and also control all of the other electrical components in the system. It was crucial that all of the hardware components were Ethernet compatible in order to have full communicative capabilities.

3.2 Hardware Implementation

To implement a brake system, proximity sensors were mounted on the machine to sense metal from the gull wings. The team decided on using inductive sensors, because they are rated to sense metal better than capacitive sensors. The proximity sensors are connected to a brake solenoid that is set to lock once the sensors no longer read metal. These components are connected using a digital I/O module with the two sensors as inputs and the brake as an output. The module is Ethernet capable to allow the system to be programmed via Visual Basic. Furthermore, to control the exercise weight, the team chose a stepper motor that is controlled by a stepper drive. The stepper motor is able to increase or decrease the weight resistance for the user. To increase the weight, the stepper motor moves the weight away from the pivot point of the weight beam on the back of the machine, which increases the tension on the cables. To decrease the weight resistance, the stepper motor moves the weight away from the pivot point. To track the amount of reps and sets, an absolute, multiturn encoder was intended to be utilized. The encoder has the ability to track the amount of rotation of the weight beam on the back of the machine. That information, along with the starting position, can be used to count the number of reps. Once the patient begins a rep, the encoder tracks the position and considers it as a completed rep as it returns to the initial position. Another use of the encoder can be to control the range of motion of the patient, implemented by the same method as counting reps. While in setup mode, the patient can perform an exercise to his or her full range of motion; the position is recorded and used to monitor and control the range of motion during an exercise routine in run mode. Unfortunately, the encoder was not able to be fully integrated into the system as the physical machine was completed the weekend before design day, leaving the encoder to be mounted and calibrated after that time. In addition, the team decided to use a touch screen industrial PC at the machine to control all of the hardware and also as a display for the patients, Ethernet and Wi-Fi capable. In order for all of these components to be able to communicate safely, the team is using a managed Ethernet switch with eight ports to connect the hardware to the industrial PC via one Ethernet cable.

Figure 3.1: Mounted Hardware



(a) Mounted Proximity Switch (b) Mounted Servo Motor (c) Mounted Encoder

3.3 Software and Interface Design Requirements

The software portion of the shoulder exercise system is composed of three main units: patient application, therapist web browser or application, and database.

The patient application was requested to be implemented on a touch screen industrial PC, to be mounted onto the machine itself. Upon use of the PC, the sponsors wished to see setup mode, run mode, and manual mode available to the patient. Setup mode is intended to train the range of motion for new users, with a secure login to link to their specific data. Run mode is the most elaborate option, as it needed to fetch the prescribed workout for the user who logs in (out of nine exercises available). After the initial login, only the exercises that had been prescribed were to be visible, and the individual exercises were to have a separate display. The display was requested to show a moving animation of the selected exercise, along with the running tally of reps and sets. After the patient completes one exercise, the app needed the ability to return to the original list to select another exercise, until completion. Once the assigned exercises are successfully finished, a report was to be sent to the therapist's account to present the updated statistics. Manual mode was to allow for any user to sit down and immediately choose any of the nine exercises, without needing a user ID or workout prescribed.

To incorporate the therapist side, it was requested to create some type of web browser or application to hold therapist accounts. The decision was left for the team as to whether a website or down-loadable application would best suit the project. After the therapist securely logs in, it was required to allow for creation of new patient accounts, and assignment of new workout plans. The other important point was to enable therapists to view completed reports.

Implementing a database was an important standard to meet. From the patients' perspectives, the database would hold the assigned workout and fetch it upon logon. Then after completing the prescribed workout, the information would need to be uploaded to the database. From the therapist's perspectives, the database needed to allow for uploaded workouts and pulling in information of completed reports. Since this database will contain patient data, it had to adhere to all HIPAA regulations. The decision was left to the team as to the type and details of which database to use.

3.4 Software Implementation

3.4.1 Patient Interface

To best implement the patient application, it was decided to use Visual Basic (via Visual Studio) to represent the interface. The software designers strived to deliver a robust application that was pleasant to the eye, as well as easy to navigate. Since one of the key requirements for this project was reproducibility, the application was programmed with a dynamic design that could adapt to a wide range of screens without causing error or an undesirable appearance. A sample of this code can be seen in figure 3.2. For more project code, please see Appendix 7.3.2. This feature was implemented by utilizing software to measure the current screen size, then sizing and placing elements and controls of the application based on the discovered screen size. The dynamic design was tested on multiple screens ranging from 12-inches to 40-inches diagonally. By implementing the resizing capability, future machines can be built with varying PC sizes, all capable of running the patient interface without need to change any of the application code.

Figure 3.2: Sample of Dynamic Location and Sizing Code

```
Exercise_Label.Location = New Point((My.Computer.Screen.Bounds.Size.Width / 2)
           - (Exercise_Label.Size.Width / 2), BorderPadding)
Me.WinMediaPlayer.Width = My.Computer.Screen.Bounds.Size.Width * (2 / 3)
Me.WinMediaPlayer.Height = My.Computer.Screen.Bounds.Size.Height * (1 / 2)
ProgressBar1.Height = WinMediaPlayer.Height
```

The main screen of the application allows the patient to enter the three modes as requested. Setup and run modes are password-protected, whereas manual mode allows free range of therapy. Upon a patient's request to a password-protected section of the application, a log-in message box will appear prompting for the user's credentials. Since this application was developed to be used on a touch screen PC, an instance of the Windows On-Screen Keyboard (OSK) process is started when either the user name or password text boxes are clicked to allow the patient an input method. Once the patient has entered his or her credentials, the application checks the information by connecting to the SQL database [14] and running a query to ensure that the entered information is valid. Since patients and other users will have physical access to the PCs running the patient interface, no account data is stored locally on the PC. Through this feature, unwarranted access to patient login credentials is prevented.

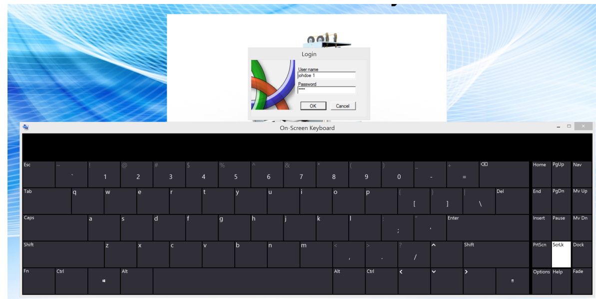


Figure 3.3: Login Form and OSK

Once the patient enters his or password into the setup mode login prompt, a greeting displays with a personalized message. The screen then presents a list of all nine available exercises. Upon selecting one, the user will then view a progress bar, as well as a note to complete three reps of the selected routine. By fulfilling three reps, the application creates an average of the patient's unique range of motion for each exercise, which is stored and put into perspective in the prescribed exercises. The patient is able to return to the page of nine exercises, and continue to set up the range of motion for the remaining.

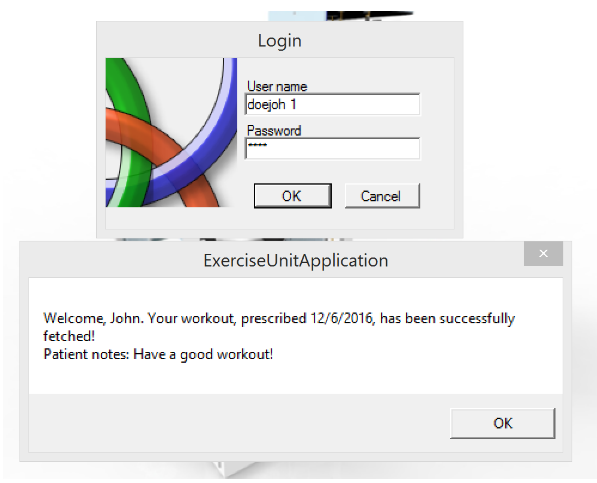


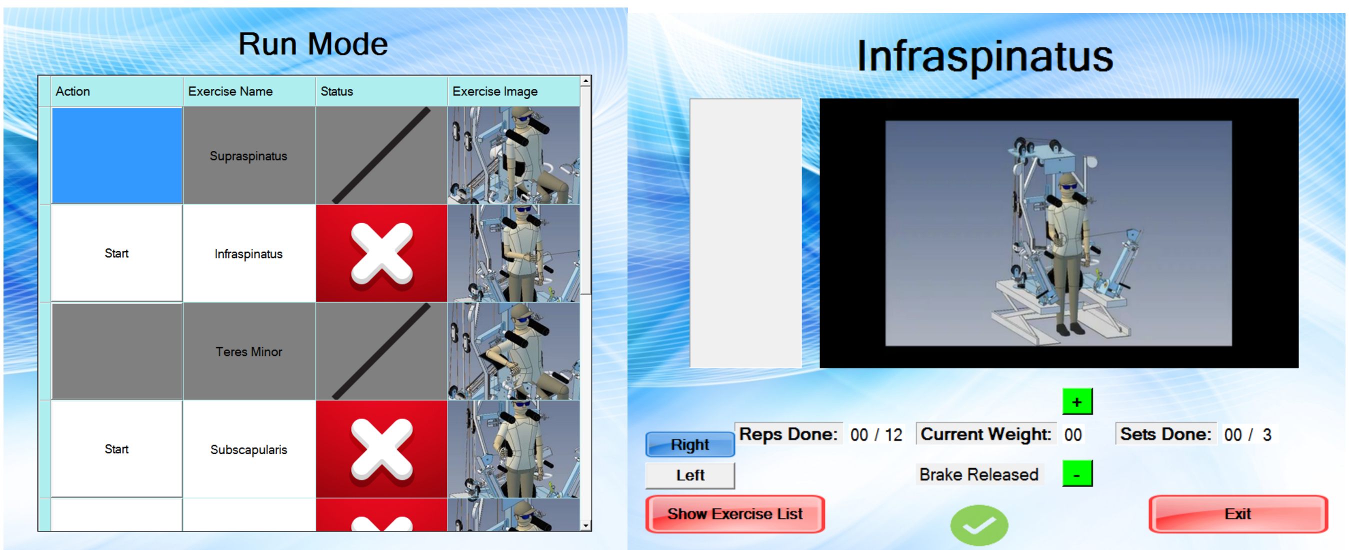
Figure 3.4: Run Mode Login

Run mode is the core of the software requirements, as this is the central point of therapy operation. After the patient securely logs in, they are greeted with a personalized message, including any notes that the prescribing therapist had for the patient, and the application fetches the latest prescribed plan. Then, only those exercises prescribed will be available to select. The exercises in which the therapist does not desire his or her patient to complete are represented as "grayed-out"; unavailable to select.

Once the patient selects an available routine, he or she will then view an animation of the selected routine, as well as the status of completed reps and sets. Also visible is a progress bar that represents the user's range of motion, and shows the range of motion that is needed to complete each rep. The intent is to track the status of the encoder and show a visual representation of the current position in the motion of the exercise against what the patient's max range of motion is. It also serves as an indicator to represent what the prescribed range of motion is for the selected workout. Unfortunately, these features where not completed in the final project due to the sponsor completing the physical device the weekend before Design Day. In result, ample time was not left for the team to complete the calibrations needed for this feature. Furthermore, an icon is near the bottom of the screen acting as a visual indicator of the brake status. While in the exercise display mode, the brake is released at all times, except when one of the two monitored exercises is being preformed. As per the sponsors request, while the Infraspinatus or Subscapularis exercise is being preformed, the status of the gull wing is constantly being monitored via the signal received from the attached proximity sensor. The two named exercises require the user to select which arm they are exercising, and the application will ensure the exercise can only be performed if the corresponding gull wing is fully down. In effect, the patient must maintain the proper form of having their elbow tucked into their body. If the elbow comes away from the body, the gull wing raises and breaks the connection with the proximity sensor. The application will see the change in signal and enable the mechanic brake gently to prevent further movement on the pulleys. A visual indicator will also appear on the screen telling the patient that the brake has been engaged, along with a message requesting the gull wing to be lowered back to position. Once proper position is regained, the brake is disabled and the patient can once again perform the exercise.

From this same screen, the patient can monitor the current weight felt on the pulleys. An option also exists to increase or decrease the weight at the push of a button, carried out by the application sending the corresponding increase or decrease signals to the servo motor. The motor then turns, changing the displacement of the weight from the pivot point as well as changing the weight on the pulleys. The patient interface then updates to display the new weight on the pulleys. This feature is import, because it enables the patients using this machine to preform all exercises at different weights without having to get up at any point during their therapy session. In result, the exercise is ideal for patients who have spinal cord injuries or are movement impaired. From this screen, the patient is also allowed to return to the prescribed exercise list to continue the progression of therapy.

Figure 3.5: Patient Interface Run Mode



(a) Exercise Selection

(b) Exercise Display

Additionally, manual mode is demonstrated much like run mode but without the monitoring feature. All nine exercises are available to select at all times, as this is intended for free roam. After selecting one, the screen will show the animation as well as track the number of reps being completed. The user is able to return to the list of nine exercises to complete any desired in a similar fashion. This allows the exercise machine to be used by any gym patrons that are not in physical therapy and thus do not have a log in.

3.4.2 Therapist Web Application

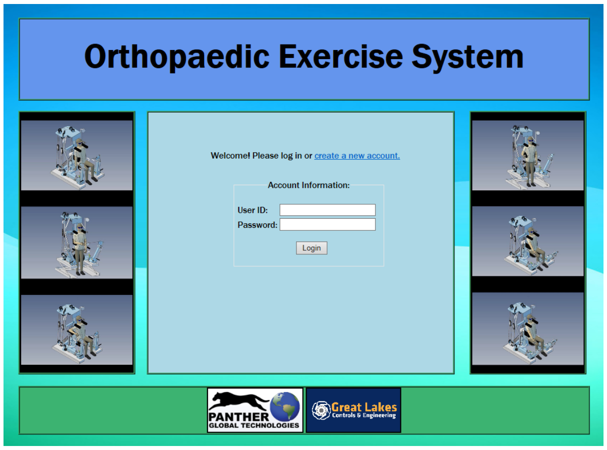


Figure 3.6: Web Application Login

In regards to the website for therapists, it was decided to manually build the website via Visual Studio (HTML and Javascript) with Visual Basic running the server code. Upon loading the URL *http://projectbaymax.azurewebsites.net/* a welcome screen is presented which allows for secure therapist logon. After the user enters in credentials, the website connects to the SQL database and validates the information. The connection is built on the server end and only harvests the data from the client's browser. This feature is important because it hides all of the connection and authentication code from the browser, making the log in methods indeterminable if the user reads the source code for the website viewable via the browser. The server login function also checks to make sure that the login is set up with therapist permissions, and also checks which organization or health care group they are associated with. This prevents patients from logging into the website and allows therapist access to only data belonging to patients registered with their organization or health care group. The role-based authentication protects patient's data from other patients as well as from unauthorized therapists acting on behalf of different organizations. After logon, the user is able to choose from multiple options: "Create New Patient", "Assign Workout Plan", "View Reports", and "Sign Out".

Through "Create New Patient", the therapist assigns the patient-specific credentials to be used at the machine. To create a patient, only four fields are required: first name, last name, password, and confirm password, though additional information for the patient may also be entered at this screen. Once the required fields are filled out and the submit button is clicked, a processing request is sent to the server. All of the code to handle creating patients is handled on the server, thereby, hiding the implementation methods from the website source code. The "create patient" function first ensures that both passwords provided match then it generates a unique user name for this new patient. Unique user names are generated by taking up to the first five characters of the last name and then appending up to the first three characters of the first name. If this user name is already present in the database, integer values are then appended to the end of the user name, counting up starting at 1, until a user name is generated that is not present in the database. For example, if there happened to be three patients named John Doe, their user names would be: "doejoh", "doejoh1", and "doejoh2". Once a unique user name is generated, the server appends records to the "Users" and "Demographics" database tables with the group that the patient belongs to set to the same as the therapist that created the patient. At this point the patient can now log into one of the exercise machines using set up mode. They must have a workout prescribed to them before they are capable of using run mode.

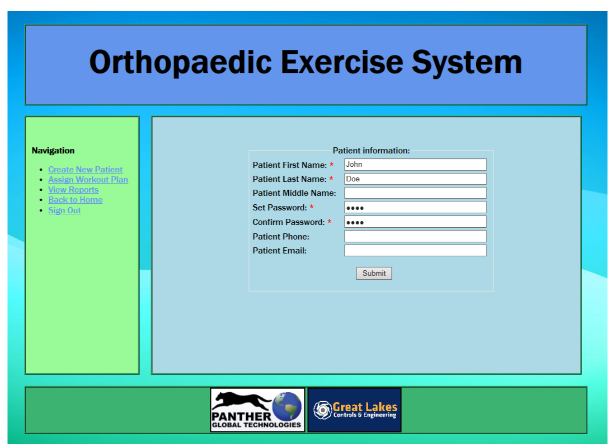


Figure 3.7: Web Application Create Patient

Via the website, therapist can also prescribe custom workout plans to any of the patients belonging to their organization or health care group. By default, all patients belonging to the organization are displayed in the drop down selection box, however, this list can be filtered by first name. The therapist selects the recipient for this workout from the drop down selection box and picks a date for the workout. Upon clicking on the date box a calendar appears, allowing the selection of the date to be done at the click of a button. If no enter is entered, the current date is entered automatically. Next, the therapist must check off each of the exercises that they would like the patient to perform. Once an exercise is selected, four option boxes appear allowing the therapist to set the number of sets and reps that they would like the patient to complete, along with a recommended max range of motion and weight that should be tried for each particular exercise. At the bottom of the page, the therapist may also enter notes for the patient to see upon login at an exercise machine or notes for themselves for future reference. All of the code to handle the submission of prescribed workouts is handled on the server end. Each workout that is posted is created with a unique workout ID that allows data corresponding to prescribed workouts to be measured against the completed workouts counter part. This feature was vital when creating the generated reports that are available to the therapist via the website.

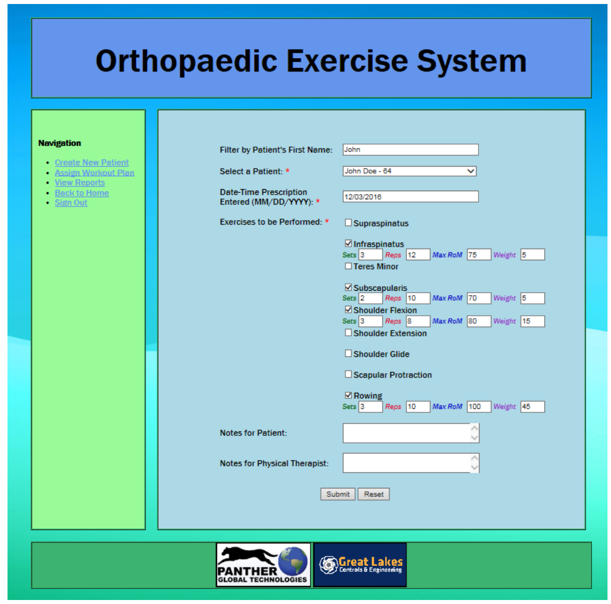
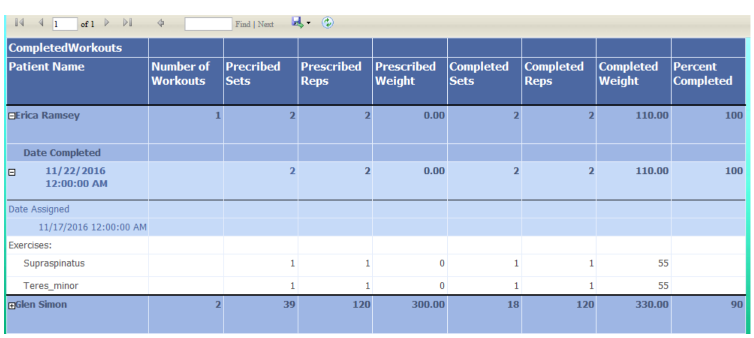


Figure 3.8: Web Application Prescribe Workout

Therapists can access reports for patients in their organization using the "View Reports" tab. This tab displays auto-generated reports aggregating patient data for the therapist's organization. These reports feed directly from the database and provide real-time feedback from what data is currently available, so as soon as patients complete their workouts, the reports will be able to display the results automatically and instantaneously. Currently there are three reports available. One stepped report presents details of all completed workouts, which can be drilled down to display details of individual patient progress. Each individual patient can be drilled down further to display all of the completed works and the details for each exercise during each workout. Completed workouts can also be displayed side by side with what was prescribed. A column of the report also indicates how much of the workout the patient was able to complete to allow a quick indication if patients are fully completing their prescribed workouts. There are two reports available that relay data about prescribed workouts. The first follows the same format as the completed workout report previously described. The other is a wide style report containing all of the same data in columns rather than having to drill down to display it. All reports can be exported to a Excel, Word, or PDF format at the click of a button.

Figure 3.9: Completed Workouts Report



1 of 1 Find | Next

CompletedWorkouts								
Patient Name	Number of Workouts	Precribed Sets	Prescribed Reps	Prescribed Weight	Completed Sets	Completed Reps	Completed Weight	Percent Completed
Erica Ramsey	1	2	2	0.00	2	2	110.00	100
Date Completed								
11/22/2016 12:00:00 AM		2	2	0.00	2	2	110.00	100
Date Assigned								
11/17/2016 12:00:00 AM								
Exercises:								
Supraspinatus		1	1	0	1	1	55	
Teres_minor		1	1	0	1	1	55	
Glen Simon	2	39	120	300.00	18	120	330.00	90

3.4.3 SQL Database

The database was implemented using Microsoft Azure for several reasons. First, the sponsor wanted as little upkeep as possible. Microsoft Azure offers a near hands-off maintenance option. They will manage how and where the database is hosted, even handling crashing and restoring to make the database consistently operational with all of the details transparent to the developer. Azure also offers a highly customizable and high-level GUI for developers. This will allow the sponsor to check the status and utilization of the database without having to know any of the details about how it is structured or maintained. Azure also offers transferable subscriptions and resource groups with no down time. This will allow the sponsor to take control of the database at the completion of the project without interface to the running of the patient interface application or website. Microsoft Azure databases are also easily upgradeable, therefore the team could go with the cheapest option during development, but more storage and resources can be acquired as needed with no down time to the service. Most importantly Microsoft Azure is secure, with top of the line firewalls and policies in place, which is important when handling medical data.

One of the criterias for the database was for it to be HIPAA compliant, mainly targeted at the distribution of medically protected data. To satisfy HIPAA, the database and applications were developed to operate using role-based authentication. Specifically each user is assigned a role: patient, therapist, or developer. Patients only have access to their own data, therapist will have access to all data corresponding to the organization that they are a part of, and developers have access to all data for maintenance purposes. At the completion of this project, the only developer users will be that of the sponsor.

For the protection of the users, all passwords are encrypted by being passed through a hash function before being stored. The data base then only stores the binary data of the result of this hash function. This makes it near impossible for any one, including developers, to determine a user's password even if they have access to the database, unless they also have the specific hash function used during the encryption.

3.5 Problems

The servo motor and drive that were initially selected could not be obtained within a reasonable time frame. Several servo motor and drive combinations were considered as substitutes; however, none of these options were available or would not be available within the project timeframe. The project sponsor suggested a stepper motor and drive be used as a substitute. A stepper motor and drive had not been considered because the project sponsor had initially requested a servo motor. The stepper motor and drive that was chosen as a substitute met most of the requirements that were laid out for the original servo motor and drive. However, the stepper motor and drive were not EtherCAT compatible. The drive, however, had onboard program memory and could be controlled with a terminal program provided by the manufacturer. While this terminal would allow the IPC to control the motor, the application would have no way of communicating with the motor. A communication subroutine would have to be developed and integrated into the application. Luckily through coordination with the project application designers, communication with the stepper motor via the patient interface was established and the patients could control the weight system with the press of a button.

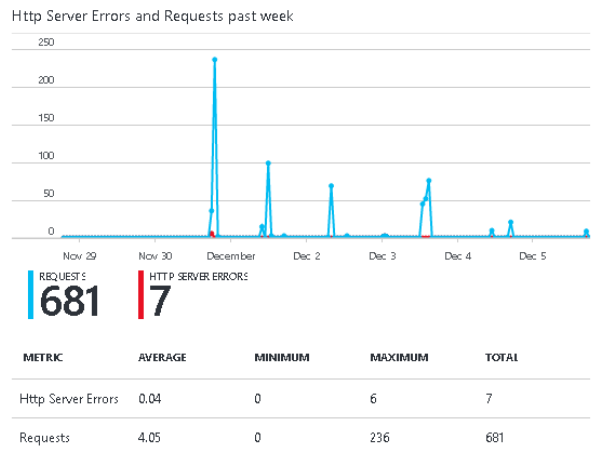
Another issue that existed during the design process was of properly programming the proximity sensors and brake solenoid. The proximity sensors are connected to two of the inputs of the digital I/O module, and the brake solenoid connected to one of the outputs. Originally, the sensors were programmed to each have a high output when no metal was sensed, and a low output conversely. As an outcome, the brake would always be engaged, and only disengage upon detection of metal. The problem arose when attempting to incorporate both left and right proximity sensors at the same time. As the group introduced the second sensor, the only instance that the brake would release was if both sensors detected metal. Design specification required that if the gull wing is released at any point during monitored exercises, the mechanical brake is to be engaged and only the proximity sensor on the side that is actively being exercised will effect the brake. To have this functionality, interrupt handlers had to implemented in the patient interface to continuously monitor the status of both proximity sensors. On either a raising or falling edge of the signal, the input was checked to determine which proximity sensor was being tripped. If it was the same as the side being exercised, the mechanical brake would be either engaged or released.

Chapter 4

4.1 Software Performance

The web application is set up so that performance can be easily managed via a dashboard provided by Microsoft Azure. One of the capabilities of this dashboard is to allow the user to monitor the requests and errors of the web application at any given time. For this project, the web application experienced only a total of seven errors. Six of these seven errors were during the initial posting of the web application and the seventh was during the first functionality test. Modifications were made to the web application after each experienced error and no errors were encountered in following tests.

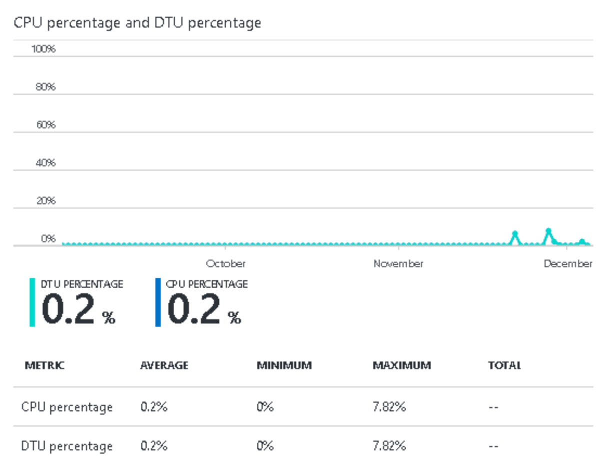
Figure 4.1: Web Application Performance



METRIC	AVERAGE	MINIMUM	MAXIMUM	TOTAL
Http Server Errors	0.04	0	6	7
Requests	4.05	0	236	681

For the ease of the sponsors, the database had monitoring set up to be available via the same Microsoft Azure dashboard as the web application. The performance of the database can be measured by observing the CPU usage and the DTU (database throughput units) percentage. The database for this project has been running since mid-September, and was tested in late November and early December once the patient interface and web application communications were implemented. Throughout the semester, the peak usage that the database experienced was 7.82%, even during the testing of the web application. This is ideal for the sponsor since they would like to have multiple patient interfaces and web applications accessing this database simultaneously. The database was also implemented so that its performance and storage capabilities can easily be expanded.

Figure 4.2: SQL Database Performance



METRIC	AVERAGE	MINIMUM	MAXIMUM	TOTAL
CPU percentage	0.2%	0%	7.82%	--
DTU percentage	0.2%	0%	7.82%	--

4.2 Overall Performance

The Design Team's project was not as data-driven as the typical ECE 480 assignment; therefore, usual testing methods and data analysis was not applicable to the exercise machine. The majority of the design timeline dealt with experimenting with different methods of integration until communication was successful. Due to this approach, quantitative data does not show whether or not the project was successful. Instead, the functionality of each hardware and software component dictates success. For proof of the operational application and website, reference Chapter 3. Below is a table listing the specifications named in the design proposal, compared to the final prototype performance. Refer to the other chapters for more detailed information.

Table 4.1: Overall Performance

Proposed Specifications	**Final Prototype Performance**
User interface for patient access	Industrial PC mounted onto machine for patient interaction
Three modes of operation	Setup, run, and manual modes have been implemented, each capable of performing specified functions
Display exercise demonstration	In all modes, application presents animated video of selected exercise
Send reports to therapist	Upon patient logout, updated report is sent to therapist via database
Password protected	Both therapist website and patient applications have unique, secure logins
Custom range of motion	Setup mode interacts with encoder to create an average, custom, range of motion for each individual patient
Control weight	Stepper drive and motor adjusts weight by user interaction at industrial PC
Monitor patient progress	Encoder tracks number of sets and reps; website allows therapist to monitor progress
Sense incorrect movement	Proximity sensors detect position of gull wing and sends signal to industrial PC
Inhibit incorrect movement	Brake solenoid engages when proximity sensors detect incorrect movement
Website for therapist access	HTML, Javascript, and Visual Basic was combined to create website to meet all design specifications
Secure database	Microsoft Azure database paid monthly for database services
Remain in budget	Teams expenditures remained under $5,000 budget

Chapter 5

5.1 Design Issues

As the prototype was critiqued for possible complications of standards and usability, two main design issues arose. To best ensure that the prototype does not fail in the future, these issues have been analyzed in order to create prospective solutions.

The feedback system is intended to monitor patients while they are sitting or standing, adjustable via a scissor lift. All of the commands for operating the exercise machine are located on the touchscreen interface, which unfortunately does not accommodate for a patient who is visually impaired. Due to the lack of certain accommodations, the prototype is not universally designed for a broad range of patients. However, it is possible to modify the design to expand the user capability. For a user that is visually impaired, a strong adaptation to the interface would be that of a keyboard, voice feedback, and initial short training session. The keyboard can be standard, with the intent to have the keypad control the exercise selection, and the “Esc” button used as a return key to the previous screen. Voice feedback can be implemented for selection knowledge, and to narrate the progress of each exercise. (For instance, a pleasant ding when a rep is completed, a harsh buzz in result of incorrect movement, the word “back” sounding when the Esc key is depressed, name of the exercise title read, welcome message upon logon, etc.) In order to train a visually impaired patient as to how each exercise is performed, a more thorough training session would be necessary, as opposed to the much simpler session required for use of this machine.

Additionally, the security of the product should be brought into consideration. With the implementation of a database to store and send patient and therapist information, the team has attempted to create HIPAA compliant storage. However, if the exercise machine is placed into a public gym, the possibility exists of running on an unsecure network. Most gyms have a public wireless network that the machines would likely be communicating through. This leaves the chance that someone could connect to this network and harvest any and all data that passes through it. Since the patient interface application does not encrypt any of its communications before transmitting, it is possible for patient data to be potentially collected by unauthorized sources. It is recommended that future work on this project involve the encryption of all patient data before being transmitted over any network.

In continuation, an important constraint that should be checked into is the safety of this product. A prototype needs to go through several safety tests before being reproduced; therefore, it is important to critique the safety measures of this machine. The component that should be optimized, in regards to safety, is that of the brake solenoid and programming thereof. Currently, if the brake is engaged and slack remains on the pulleys tension, a possibility exists of a jerking movement on the patients shoulder after the arm is moved to the correct position that is recognized by the proximity sensors. With slack on the connection, the brake will disengage and jerk until taut once again. The faulty jerk can be removed with further programming of the brakes function. One method to remove the jerk would be, after brake activation, to only respond to the proximity sensor and disengage the brake if the patient creates tautness on the line. This could be detected through software by reading positive movement from the encoder before sending any signals to the brake. The design team was unable to complete this safety functionality with the time constraint of only having the physical machine for four days before Design Day. Not enough time remained to configure the encoder readings properly and it was decided to cut this feature from the final project. It is recommended (if possible) that continuation of this project should include a refined braking feature.

Chapter 6

6.1 Conclusion

All in all, MSU's Design Team One has strived to deliver a finished prototype to Great Lakes Controls & Engineering and Panther Global Technologies. The project is not absolutely completed and ready for reproduction at this point, yet in time, the sponsors will be able to incorporate the feedback system into routine shoulder therapy exercises. It would be ideal if this task could be extended for the next class to finalize.

After much trial-and-error, many aspects of the feedback system were successes. Communication between the software application, website, and database was able to come together and have a fully linked data system. The database stores tables of therapist and patient information, including patient data, prescribed workouts, and completed workouts. Due to the success of the software portion, the functionality exists to allow therapists to prescribe workout plans, patients to fetch these plans, and the system to automatically submit reports. Also, the team strived to ensure a secure log-on, as patient data should always be on protected systems. Not every attempt at the project resulted in success, since the majority of the progression was experimental. One failure that hindered the project was the difficulty of obtaining the wide range of hardware components in a timely manner. The required components were such that they needed to be custom-ordered; therefore, it may have been underestimated from the beginning as to how long it would take to communicate with the manufacturers. Due to this hindrance, the members working on the hardware tasks had a late start. Another significant issue was that of integrating the hardware components with the patient application. Ideally, the team attempted to purchase components that were compatible with TwinCat, which would allow for hardware/software integration. However, most of the purchased parts resulted in an incompatibility. The TwinCat failure was solved by the team hard-coding the hardware commands into the patient application code.

Design Team One recommends the feedback system to be continued into another semester, as the scope of this project is significantly large with much room for growth. Since the integration of hardware and software with the machine itself is not fully completed, it is worthwhile to allow continuation of the project. Specifically, it is suggested to locate hardware components that are all compatible with the same application (such as TwinCat). By ensuring this development, it is possible to simplify the way that the hardware is implemented. The therapist website is mostly sound, although it could benefit from a redesign of layout if desired from the sponsor. The patient application, likewise, is sound but has room for very minor tweaks.

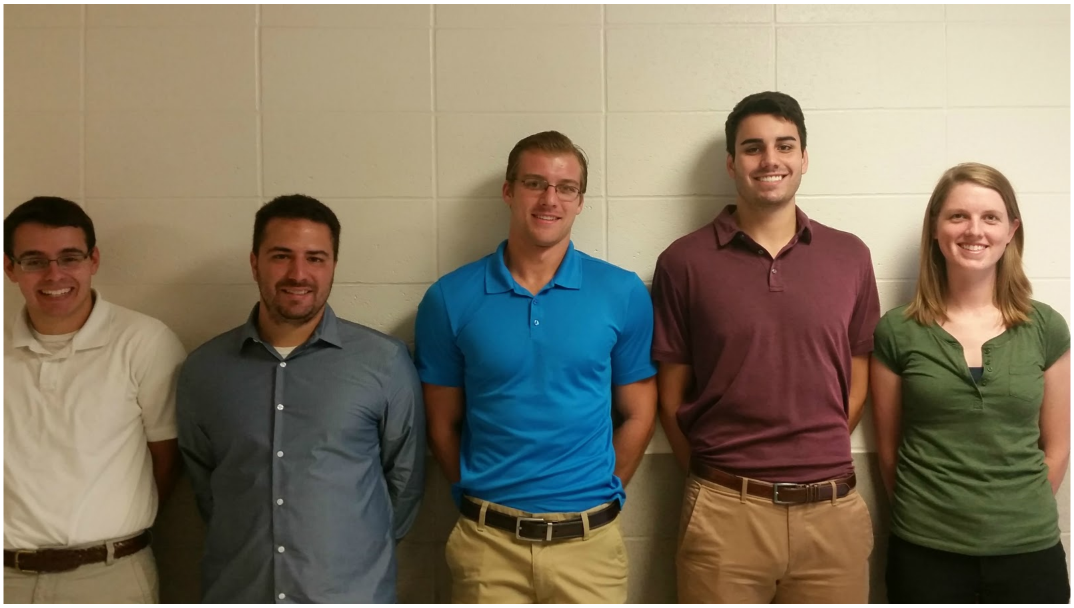
At the time of the proposal, the team's initial servo unit and sponsor's recommended industrial PC came to a total of over $8,500. Through careful deliberation and research into the overall design of the feedback system, it was decided that a change of industrial PC and servo unit would result in the final cost being half of the projected. Both of the discovered replacements met the specifications of the project, and were therefore determined to be acceptable. Due to the replacement of these components, the team was able to drop the original proposed cost of roughly $10,720 to $4,476, saving a total of $6,244. Though the sponsor had agreed to the original proposed cost of the project, the described changes allowed for the overall cost of the machine to be dropped dramatically, thus making the final product marketable and reproducible.

The beginning stages of creating the project consisted of the initial meetings and brainstorming. The team met with the facilitator and also visited the sponsors to discuss the fine details of what was projected. After that, it was decided that the team would designate three members to the hardware portion, and two to the software, allowing for concurrent progress. The hardware members began ordering the custom components, and the software members began the creation of application, website, and database. Integration was the key to the projects success, so the database designer focused on incorporating the communication throughout the software pieces. Once that had a general foundation, it was then possible to integrate the hardware. At that point, a sizeable amount of time was designated to experimenting and troubleshooting, as the most difficult part was found to be the hardware communication. Once the sponsor had delivered the custom-built, physical machine, the final steps were carried out to incorporate the feedback system into the device.

Appendix

7.1 Technical Roles

Figure 7.1: Design Team One



(a) Pictured left to right: Joe Fabbo, Michael Juricny, Glen Simon, Rafee Muhmud, and Erica Ramsey

7.1.1 Joe Fabbo - Servo Unit Specialist

The initial design called for an EtherCAT compatible AC servo motor and drive to control the position of the weight beam. The parts that were initially selected to fill this role were the ABB microflex e150 servo motor and a Baldor servo motor. Sourcing these parts proved to be quite a challenge. Every distributor that was contacted either did not carry these parts or had an unacceptably long lead time. The search for substitutes for these parts did not yield a satisfactory option. After several weeks of searching the project sponsor suggested using the Applied Motion ST10-Q-EE stepper motor drive and the Applied Motion HT23-603D stepper motor. While these components met the mechanical criteria, they were not EtherCAT compatible. The stepper drive could be controlled with analog pulses, an onboard program, or controlled using commands sent via the Ethernet or serial port. Analog control was not considered because it would require additional hardware. While the onboard programming capability would allow the control of the motor without additional hardware, the application would not be able to communicate with the drive. It was possible to send commands directly to the drive over a tcp connection. This required the creation of a communication subroutine in the application to send commands to the drive. The ability to send commands to the drive made it possible to send and receive data from the application. The application could call for the execution of subroutines stored in the drive's memory. Furthermore, data such as current position and motor speed could be read from the drive. This data could be used to determine the current weight setting. Because of limited time it was decided that the best course of action would be to send commands directly to the drive from the application and not utilize the drive's program memory.

7.1.2 Michael Juricny - Brake Solenoid and Proximity Sensor Specialist

It was essential for the project to have two proximity sensors that would trip the brake solenoid. This is essential for the infraspinatus workout. The proximity sensors were placed on the gull wings of the machine so it could read either the left side or the right side of the machine. Once the workout is selected and which arm is being worked out the brake is inactivated until the sensor can no longer read the gull wing. This is done to protect the patient from re-injuring the surgery repaired shoulder. To have the proximity sensors to communicate to the brake solenoid a digital I/O module is needed and a relay is needed for the brake solenoid. The sensors are connected to two of the inputs on the I/O module and the brake is connected one of the outputs. Once these components were connected Visual Basic was used to program the sensors to trip the brake at the right time. This was done by first programming one sensor to have a high output when there was no metal to sense, and a low output when the sensor was sensing metal. This was done because the brake will turn on when the output of the sensor is high and the brake will turn off when the output of the sensor is low. Once the one sensor was working like expected, the second sensor was then implemented. There was a problem with the system when the second sensor was first introduced. When trying to run one program the sensors would rely on each other, when for the brake system to work properly they needed to be individual. This is so when one of the sensors is supposed to be activated the other sensor is not also activated. The easiest way to fix this problem was just to have two programs each indicating a different sensor. Now when the exercise is selected the arm is also selected and only one of the sensors will be turned on.

7.1.3 Glen Simon - Application Designer and Database Developer

Since the project was software intensive and consisted of two separate applications being able to communicate with a shared database, early planning and development were key to success. After receiving the details of this project, a database design was developed and tested in the early stages of this project to ensure that it was capable of containing all needed information while still being expandable. Having a strong background as a database developer and working with HIPAA compliant databases aided in the foresight required to complete this task. It also gave familiarity with key applications, such as Visual Studios and Microsoft Management Studios, and programming languages, such as Transact SQL and Visual Basic, needed to complete the software and database. This previous experience allowed for a functioning database to be set up and hosted using Microsoft Azure in the first couple weeks of the project, while having a design that only needed minor changes during the entirety of the project.

Having a functioning database in the first couple weeks of the project allowed for early development of the patient interface. Having developed form based applications using Microsoft Access before, it was decided that developing the patient interface in a similar environment, using a Windows Forms Project running Visual Basic, would be beneficial in an expedited build of the application. While the other application designer for this project was becoming familiar with the languages and environments used for this project, this member was able to build the foundation for the patient interface. This involved coming up with the design for the interface and deciding on how best complete all of the requirements for this project. By the completion of this foundation, the other developer was ready to take over responsibility for the front-end of the interface. This allowed for personal efforts to be focused on the back-end functionality. Over the course of the project communications with the database, demonstration video playing capabilities, interaction with hardware, and some front-end aspects were all completed.

After the other developer produced the foundation for the web application, efforts were made by this developer to take charge of the server-end code that would be responsible for handling any interactions with the database. This involved developing methods responsible for therapist signing in and out of the website, restricting access of the website from patient accounts, creating new patients, prescribing workout plans to patients, viewing reports and implementing the actual hosting of the web application. A reporting service was developed to help auto generate reports based off the real-time data present in the database. This allowed for the displayed reports to never have to be manually updated when new data became available. Instead available reports are instantaneously updated whenever the page is refreshed.

7.1.4 Rafee Mahmund - Encoder Specialist

One of the main hardware components for the exercise machine was the encoder. The encoder is located at the pivot point of the weight beam on the back of the machine. Initially, research had to be done to make sure that the right encoder was selected for the project. Encoders can be either incremental or absolute, and within those two categories can be conductive, optical, or magnetic. After research, contacting different companies, and speaking to technical support, it was decided that an absolute, optical, multi-turn encoder with a 12mm blind hollow shaft would be used. It also was important for the encoder to connect to the industrial PC via EtherCAT so real time monitoring of the outputs of the encoder was possible. Next, an encoder with the proper specifications had to be found with a lead time that worked with the design team's timetable. After speaking with multiple companies again, the best option was to order an encoder from SICK Sensor Intelligence with a lead time of 3-4 weeks. While waiting for the encoder this group member helped with work on the other components (team members in charge of other components helped with the encoder when it arrived as well). This included working in Visual Studio to program the brake solenoid to lock when the proximity sensors detected metal, and working on the stepper drive and motor so the weight can be increased and decreased from the industrial PC. Once the encoder arrived work began on programming it through Visual Studio. EtherCAT devices can be controlled through a program called TwinCAT 3. It was necessary to integrate TwinCAT 3 with Visual Basic through the use of a DLL. After that, the encoder was programmed so the step count output was displayed. The step count shows how much the encoder has been rotated from its initial position and can be used for tracking the reps and sets of the user's workout.

7.1.5 Erica Ramsey - Application Designer

As one of the two application designers, it was pertinent to begin the developing stages as soon as the scope of the project specifications was laid out. The first step was to focus on the patient application, as the interface was necessary to set up because the hardware pieces were dependent on integration with the app. As this member had not used Visual Basic or Visual Studio prior to the project, the first two weeks were designated to a crash-course in the new language. In the meantime of these learning stages, the other designer had created the foundation to the app, so then it was possible to build off of that strong foundation from the other designer. The layout and user-friendly design was a main focus during those design phases, so that the rest of the design process would be allotted to create integrated functionality. After the patient application had the basic functionality, the designer began to create the therapist website. HTML was also a new language for this member, so it involved several experimental websites in order to create a solid knowledge-base. To begin with the web development, basic blocks were placed into the page and backgrounds, as well as themes/colors. As each new page was created, the designer communicated closely with the database developer, who would implement the database into each new function. The page that seemed to consume the most time was that of assigning workout plans. Multiple types of tools had to be implemented to allow for completely customary plans. A certain struggle was that of using a popup calendar into the form, which may have been basic for a skilled developer, but seemed to take some time to understand. Then, a goal that this designer wished to achieve was that to hide the individual textboxes for reps, sets, weight, and range of motion, unless the applicable exercise was check-marked. After some experimenting, the desired function was executed. Once the main goals for the patient application and therapist website were fulfilled, the rest of the time spent went towards testing and further optimizations.

7.2 References

1. Panel PC
Logic Supply. (2016). Cincoze Crystal 12.1" Industrial Panel PC. Logic Supply. Retrieved from http://www.logicsupply.com/cv-112-p1001/.

2. Ethernet Switch
AutomationDirect. (2016). SE-SW8M. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Communications/Ethernet/Ethernet_Switches/Managed_Ethernet_Switches_-a-_Accessories/SE-SW8M.

3. Proximity Sensor
AutomationDirect. (2016). AES-AN-1A. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Sensors_-z-_Encoders/Inductive_Proximity_Sensors_-z-_Proximity_Switches/8mm_Round_Industrial_Automation/Standard_Sensing_(8mm)/AES-AN-1A.

4. Brake Solenoid
Guardian Engineering and Manufacturing. (2011). Model-TP12x19-Push-Type-DC-Tubular-Solenoid. Guardian Electric Manufacturing Co. Retrieved from http://electromechanicalcomponents.guardian-electric.com/item/dc-tubular-solenoids/push-type-dc-tubular-solenoids/a422-064104-02-2.

5. Ice Cube Control Relay
AutomationDirect. (2016). 781-1C-24D. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Relays_-z-_Timers/Electro-Mechanical_Relays/Square_-z-_Cube_Relays,_Plug-In,_3A_-_15A_(78x-z-_Qxx-z-H78x_Series)/General_Purpose,_15A_(781_-z-_782_-z-_783_-z-_784_Series)/781-1C-24D.

6. Relay Socket
AutomationDirect. (2016). 781-1C-SKT. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Relays_-z-_Timers/Relay_Sockets_-a-_Accessories/Relay_Sockets/781-1C-SKT.

7. 24 V DC Power Supply
AutomationDirect. (2016). PSP24-240S. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Power_Products_%28Electrical%29/DC_Power_Supplies/24VDC,_DIN_Rail_Mount/PSP24-240S.

8. Absolute Encoder
SICK AG. (2016). Absolute encoders AFS/AFM60 EtherCAT. SICK Sensor Intelligence. Retrieved from https://www.sick.com/us/en/encoders-and-inclination-sensors/absolute-encoders/afsafm60-ethercat/afm60a-beeb018x12/p/p301974.

9. 24 V DC Power Supply (Stepper)
AutomationDirect. (2016). PSB24-240. AutomationDirect.com. Retrieved from https://www.automationdirect.com/adc/Shopping/Catalog/Power_Products_(Electrical)/DC_Power_Supplies/24VDC,_DIN_Rail_Mount/PSB24-240.

10. Stepper Motor w/ Encoder
Applied Motion Products, Inc. (2016). HT23-603 - NEMA 23 High Torque Step Motor, 354 oz-in. Applied Motion Products: A Moons Company. Retrieved from http://www.applied-motion.com/products/stepper-motors/ht23-603.

11. Encoder Extension Cable
Applied Motion Products, Inc. (2016). 3004-195-10 - Encoder Extension Cable, 10 ft. Applied Motion Products: A Moons Company. Retrieved from http://www.applied-motion.com/products/accessories/3004-195-10.

12. Stepper Drive
Applied Motion Products, Inc. (2016). ST10-Q-EE - DC Advanced Microstep Drive w/ Ethernet & Encoder Input. Applied Motion Products: A Moons Company. Retrieved from http://www.applied-motion.com/products/stepper-drives/st10-q-ee.

13. Brainbox Ethernet to Digital I/O
NextWarehouse.com. (2016). BRAINBOXES Brainboxes ED-588 Ethernet to Digital IO 8 Inputs + 8 Outputs. NextWarehouse. Retrieved from http://www.nextwarehouse.com/item/?1470557.

14. Database
Microsoft. (2016). SQL Database Pricing. Microsoft Azure. Retrieved from https://azure.microsoft.com/en-us/pricing/details/sql-database/.

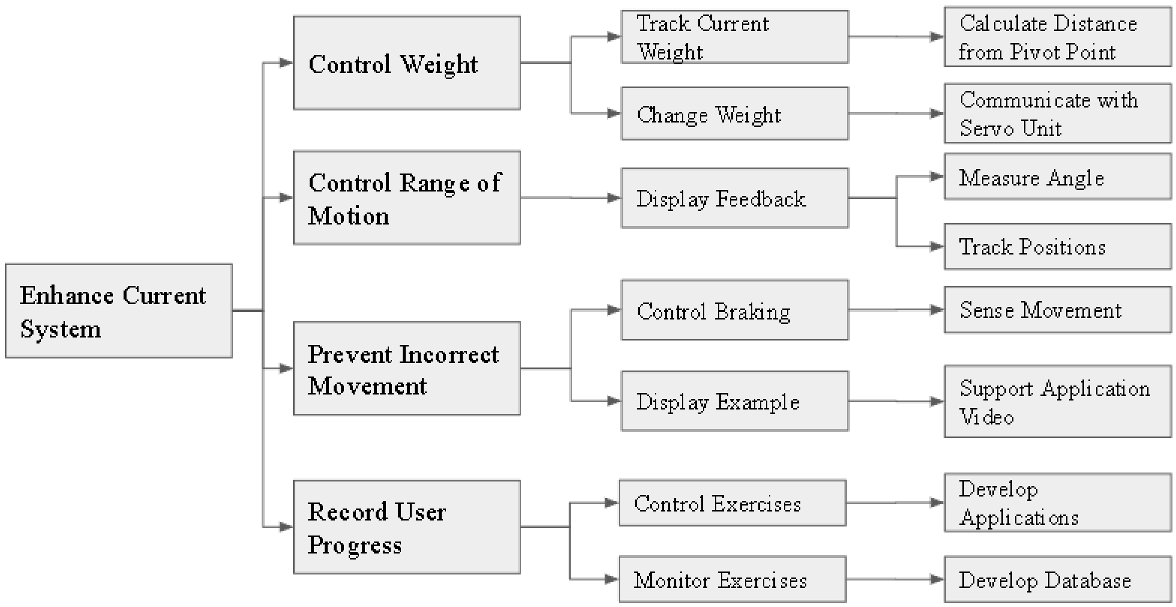
15. Visual Basic
Microsoft. (2016). Developing Applications with Visual Basic. Microsoft: Developer Network. Retrieved from https://msdn.microsoft.com/en-us/library/k6a6etxs.aspx.

16. Images and Animations from Sponsors Presentation
Groat, R. (2016, May). Abridged Shoulder Machine Final.pptx. Presentation.

7.3 Technical Attachments

7.3.1 FAST Diagram

Figure 7.2: FAST Diagram



7.3.2 Software Code

As the software code to the project is immensely long, snippets of significant importance are shown below. The complete and commented software code will be given to Panther Global Technologies as well as to Great Lakes Controls and Engineering.

```
'Resize necessary objects based off of which computer is used
If My.Computer.Screen.Bounds.Size.Height < 900 Then
Exercise_Label.Location = New Point((My.Computer.Screen.Bounds.Size.Width / 2) -
    (Exercise_Label.Size.Width / 2), BorderPadding)
Me.WinMediaPlayer.Width = My.Computer.Screen.Bounds.Size.Width * (2 / 3)
Me.WinMediaPlayer.Height = My.Computer.Screen.Bounds.Size.Height * (1 / 2)
ProgressBar1.Height = WinMediaPlayer.Height
Else
Exercise_Label.Location = New Point((My.Computer.Screen.Bounds.Size.Width / 2) -
    (Exercise_Label.Size.Width / 2), Exercise_Label.Size.Height + BorderPadding)
WinMediaPlayer.Height = 404 '(478, 279)
WinMediaPlayer.Width = 540
End If
-----------------------------


'Load in prescribed sets and reps
If GlobalVars.DestinationForm = GlobalVars.Forms.RunMode Then
setsMax.Text = GlobalVars.GetPrescribedSets(GlobalVars.CurrentExercise)
repsMax.Text = GlobalVars.GetPrescribedReps(GlobalVars.CurrentExercise)
ElseIf GlobalVars.DestinationForm = GlobalVars.Forms.SetUpMode Then
repsMax.Text = 3
End If


-----------------------------


'Toggle active proximity sensor side from left to right
Private Sub Right_Arm_Button_Click(sender As Object, e As EventArgs) Handles Right_Arm_Button.Click
If GlobalVars.ActiveSide = 1 Then
GlobalVars.ActiveSide = 0
Right_Arm_Button.FlatStyle = FlatStyle.Flat
Left_Arm_Button.FlatStyle = FlatStyle.System
Me.Refresh()
End If
End Sub


'Toggle active proximity sensor side from right to left
Private Sub Left_Arm_Button_Click(sender As Object, e As EventArgs) Handles Left_Arm_Button.Click
If GlobalVars.ActiveSide = 0 Then
GlobalVars.ActiveSide = 1
Right_Arm_Button.FlatStyle = FlatStyle.System
Left_Arm_Button.FlatStyle = FlatStyle.Flat
Me.Refresh()
End If
End Sub
```

```
-----------------------------

'Log in Form
'Connection to SQL Server
con.ConnectionString = strCon
cmd.Connection = con
con.Open()
cmd.CommandText = "SELECT * FROM Users WHERE UserName = '" &
UsernameTextBox.Text & "' AND UserPasswordHash = HASHBYTES('SHA1', '" &
 PasswordTextBox.Text & "')"
rd = cmd.ExecuteReader

'Check to see if entered username and password are valid
If rd.HasRows Then

'Harvest User Info - UserID from Users table
rd.Read()
GlobalVars.CurrentUserID = rd.GetValue(2).ToString()
'MsgBox(GlobalVars.CurrentUserID)
rd.Close()

'Harvest User Info - UserFirstName from Demographics table
cmd.CommandText = "SELECT * FROM Demographics WHERE UserID = '" & GlobalVars.CurrentUserID & "'"
rd = cmd.ExecuteReader
If rd.HasRows Then
rd.Read()
GlobalVars.CurrentUserFirstName = rd.GetValue(1).ToString()
'MsgBox(GlobalVars.CurrentUserFirstName)
Else
MsgBox("No Demographics record found for " & UsernameTextBox.Text & "!")
GoTo OK_Click_Exit
End If
rd.Close()


-----------------------------


'Allows control of bypassing braking system for testing purposes
Private Sub HelloWorld_Click() Handles HelloWorld_Button.Click
If GlobalVars.BrakeSystemConnected = True Then
GlobalVars.BrakeSystemConnected = False
HelloWorld_Button.Text = "Brake System Not Connected"
Else
GlobalVars.BrakeSystemConnected = True
HelloWorld_Button.Text = "Brake System Connected"
End If
End Sub


'Allows control of bypassing servo unit system for testing purposes
Private Sub ServoConnection_Click(sender As Object, e As EventArgs) Handles ServoConnection.Click
If GlobalVars.ServoSystemConnected = True Then
```

```
GlobalVars.ServoSystemConnected = False
ServoConnection.Text = "Servo System Not Connected"
Else
GlobalVars.ServoSystemConnected = True
ServoConnection.Text = "Servo System Connected"
End If
End Sub
```

```
 ''''''''''''''''''''''''''''' Start Servo System         ''''''''''''''''''''''''''''''''''''
 If GlobalVars.ServoSystemConnected = True Then
    If GlobalVars.tcpClientGlobal.Connected = False Then
       Dim tcpClient As New System.Net.Sockets.TcpClient()
       tcpClient.Connect("192.168.0.40", 7776)
       GlobalVars.tcpClientGlobal = tcpClient
    End If
 End If
 ''''''''''''''''''''''''''''' End Servo System           ''''''''''''''''''''''''''''''''''''
```

```
   'Proxy sensors and brake
   Dim ed As EDDevice = New ED588(New TCPConnection("192.168.137.175"))
   ed.Label = "Prox sensors IO"
   Dim prox_sensor As IOList(Of IOLine) = ed.Inputs.Take(2).AsIOList
   prox_sensor.Label = "Prox. sensors"
   Dim Relay As IOList(Of IOLine) = ed.Outputs.Take(1).AsIOList
   Relay.Label = "Relays"

   If (GlobalVars.CurrentExercise = GlobalVars.Exercises.Infraspinatus) Or
    (GlobalVars.CurrentExercise = GlobalVars.Exercises.Subscapularis) Then
   'Proxy sensor high (connection established) - release break
   AddHandler prox_sensor.IOLineFallingEdge, Sub(line, Device, changetype)

   If line.IONumber = GlobalVars.ActiveSide Then
   BrakeDisabled()
   Relay(0).Value = 0
   End If


   End Sub

   'Proxy sensor low (connection broke) - activate break
   AddHandler prox_sensor.IOLineRisingEdge, Sub(line, device, changeType)

   If line.IONumber = GlobalVars.ActiveSide Then
   BrakeEnabled()
   Relay(0).Value = 1
   End If
```

```
End Sub
ed.Connect()
'Enable brake
ed.SendCommand("#0100FF")
BrakeEnabled()
```