

Fall 12-1-2005

Sioux Valley Intensive Air Web-Based Scheduling Application

Chad Breidenbach
Dakota State University

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

Recommended Citation

Breidenbach, Chad, "Sioux Valley Intensive Air Web-Based Scheduling Application" (2005). *Masters Theses*. 64.
<https://scholar.dsu.edu/theses/64>

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.

SIOUX VALLEY INTENSIVE AIR WEB-BASED SCHEDULING APPLICATION

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

Master of Science

in

Information Systems

December, 2005

By

Chad Breidenbach

Project Committee:

Dr. Omar El-Gayar

Dr. Ronghua Shan

Dr. Stephen Krebsbach

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.

Project Committee

Faculty supervisor:  Date: _____

Committee member:  Date: May 31, 2006

Committee member: _____ Date: _____

ACKNOWLEDGMENT

A Web-Based Scheduling Application was identified by the Sioux Valley Hospital Intensive Air Program as a business need. This request became part of the Sioux Valley Information Technology Department Project List and was identified as a priority project.

I would like to acknowledge the contributions of the Web Development Team of the Sioux Valley Information Technology Department: Jeff Leat-Webmaster, Tim Rusten-Web Developer, Kendra Schnabel-Web Developer/Layout and Design.

This project is proprietary of Sioux Valley Hospitals and Health System. There was no additional compensation for this project for myself or any contributing member of the programming team.

I would also like to acknowledge the support of my wife, Elizabeth, and family who supported my efforts while working on this project. Without their support this project would not have been possible.

ABSTRACT

The pilots of the Sioux Valley Intensive Air Program currently maintain their schedule on a static calendar. They cannot directly update the schedule and upon taking the shift of another pilot, the pilot taking a shift must notify the pilot whose shift he or she took. The change must also be reported as a courtesy to the Program Manager.

The pilots of the Sioux Valley Intensive Air Program wish to implement a Web-Based Scheduling Application that will give them up-to-date scheduling information and allow them to make shift changes online while providing automatic notification of a shift change to the pilot whose shift is affected and the Program Manager.

The process of addressing this business need began by identifying this as a priority project on the Project List maintained by the Sioux Valley Information Technology Web Development Team (Project Team). Upon selecting the project, a Project Plan, Work Breakdown Structure and Gantt chart were created to organize the project and identify the activities necessary to accomplish successful application development.

End-user requirements were then determined through an initial meeting with a designated representative of the Sioux Valley Intensive Air Program. A Project Charter was developed from those requirements to ensure that the program representative and the Web Development Team had a clear understanding of the project requirements. Once the Project Charter was reviewed and agreed upon, the Project Team met to determine Project Specifications for addressing user needs. Use Case Descriptions and a Use Case Diagram were created from those specifications to model the “to-be” application I wished to create.

Application Development began by establishing a Plan of Action to summarize the approach needed to develop the application. Data requirements were identified for the application to ensure that the proper information would be maintained for the application. HTML and JavaScript code was used to develop the user interface for the application while ASP code was used so that the application would dynamically interact with the MS SQL Server 2000 database.

Once the application was developed, it was demonstrated to the representative of the Sioux Valley Intensive Air Program. The application received favorable review by the representative indicating that it would provide great value to their operation. We also discussed future enhancements to the application, but overall he stated that the application provided the functionality initially desired by the Intensive Air Program. After his approval of the application, a User's Guide was created to help the pilots use the application for their scheduling needs.

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,



Chad Breidenbach

TABLE OF CONTENTS

ACKNOWLEDGMENT	III
ABSTRACT	IV
DECLARATION	VI
TABLE OF CONTENTS	VII
LIST OF TABLES	VIII
LIST OF FIGURES	IX
INTRODUCTION	1
BACKGROUND OF THE PROBLEM.....	1
STATEMENT OF THE PROBLEM.....	2
OBJECTIVES OF THE PROJECT.....	3
LITERATURE REVIEW	7
REQUIREMENTS ANALYSIS	12
SYSTEM DESIGN	21
IMPLEMENTATION AND RESULTS	32
CONCLUSIONS	34
REFERENCES	37
APPENDIX A: PROJECT PLANNING	38
APPENDIX B: USER'S GUIDE	41
APPENDIX C: APPLICATION DEVELOPMENT	48

LIST OF TABLES

Table 1. Summary of Interview-eFlightSchedule.com	8
Table 2. Summary of Interview-SkyScheduler.....	9
Table 3. Summary of Interview-MyFBO.com.....	10
Table 4. Interview Report	12
Table 5. Project Charter Attachment.....	14
Table 6. Project Specifications.....	15
Table 7. Use Case Descriptions	17
Table 8. Data Tables	22

LIST OF FIGURES

Figure 1. Calendar Creator sample schedule.....	2
Figure 2. Network diagram of DMZ server	4
Figure 3. Work Breakdown Structure (tabular form).....	6
Figure 4. Work Breakdown Structure (graphical form).....	6
Figure 5. Project Charter	14
Figure 6. Use Case Diagram	20
Figure 7. Window Navigation Diagram.....	21
Figure 8. Data Tables created with MS SQL Server 2000.....	23
Figure 9. Database Schema	24
Figure 10. Entity Relationship (E-R) Diagram	24
Figure 11. Home page (weekly schedule).....	25
Figure 12. Monthly schedule.....	26
Figure 13. Pilot shift change window	26
Figure 14. Changed pilot schedule.....	27
Figure 15. Enter Pilot Info window.....	28
Figure 16. Schedule Admin window.....	29
Figure 17. Calendar grid	29
Figure 18. Schedule with drop-down menus.....	30

CHAPTER 1

INTRODUCTION

Background of the Problem

Sioux Valley Hospital opened in 1894 as the first hospital in Sioux Falls, SD. Now known as Sioux Valley Hospital USD Medical Center, the hospital has evolved and is now a part of Sioux Valley Hospitals and Health System. Part of their evolution included the region's first air ambulance service which is now called the Sioux Valley Intensive Air Program (Sioux Valley Hospitals and Health System-History, 2005).

The Sioux Valley Intensive Air Program has been operating since 1977 and has flown more than 15,000 patients. Intensive Air was the first CAMTS (Commission on Accreditation of Medical Transport Services) accredited flight program in South Dakota. Operations include a Bell 230 helicopter and two King Air 200 fixed-wing aircraft. Intensive Air is staffed with four specialized flight teams (Sioux Valley Hospitals and Health System-Intensive Air, 2005).

Currently, there are 12 pilots who fly for the fixed-wing portion of the Sioux Valley Hospital Intensive Air Program. The fixed-wing portion of the program has recently averaged close to 900 patient flights per year. Flights are primarily conducted within a 300 mile radius of Sioux Falls, but lately many flights have been made to points beyond said radius. They hold an FAA Part 135 Certificate (Commercial Air Carrier Certificate) that allows them to conduct business in all of the United States and Canada.

The schedule of the pilots of the Sioux Valley Hospital Intensive Air Program is currently maintained in the flight office on an application called Calendar Creator that creates a static listing of shifts on which the pilots are on duty.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1 Sherwin Shane Mark Dick Mike Tim	2 Rick Gary Mark Steve Dick Sherwin	3 Rick Gary Jeff Steve Dick Sherwin	4 Tim Kelly Rick Arlen Mike Shane	5 Tim Kelly Rick Arlen Mike Shane
6 Tim Kelly Rick Arlen Mike Shane	7 Jeff Dick Mark Tim Steve Gary	8 Sherwin Dick Mark Tim Steve Gary	9 Rick Shane Mike Sherwin Tim Dick	10 Rick Shane Mike Sherwin Kelly Dick	11 Mark Steve Jeff Gary Sherwin Rick	12 Mark Steve Jeff Gary Sherwin Rick
13 Mark Steve Jeff Gary	14 Tim Arlen Steve Shane	15 Tim Mike Steve Shane	16 Sherwin Mark Tim Rick	17 Sherwin Mark Tim Rick	18 Shane Gary Dick Arlen	19 Shane Gary Dick Arlen

Figure 1. Sample schedule on Calendar Creator

Statement of the Problem

If a pilot wishes to obtain the schedule for the current month, they must obtain a copy via fax, in paper form from the airport office or through a screen shot that would have to be emailed to them. Since the pilots do not have real-time access to the most recent schedule and cannot view recent changes that have been made to the schedule, they would not have knowledge of which pilots are available for a certain shift on a certain day. If a pilot has recently taken a shift (e.g. earlier in the same week), they may not be eligible for a shift exchange because they may be limited by FAA or company policy to the number of hours they may work that week.

If a change is to be made to a schedule, a pilot must contact the pilot whose shift they wish to take. If the change occurs, the pilot requesting the shift must also notify the Program Manager of the change and the airport office to ensure that the schedule is updated or edited. The Program Manager must also reconcile pilot hours at the end of the month to determine hours worked and report to Human Resources. The Intensive Air Pilots would not have immediate access to the most recent schedule and would have to obtain an updated schedule as previously mentioned.

Objectives of the Project

The pilots of the Sioux Valley Hospital Intensive Air Program wish to implement a Web-Based Scheduling Application that allows them to view and change scheduling information online, thereby increasing the efficiency of their off-hours labor.

The major deliverables of the project include the following: The schedule is established over a predetermined number of days and will be repeated in the next schedule (e.g. the first scheduled 13 weeks will be repeated over the next 13 week period). Any pilot affected by a change in the schedule will be notified by email of that change. The schedule will be updated (or refreshed) every ten minutes. The application will allow all pilots to view and edit schedules for all shifts.

The scope of this project will include the development of a Web-Based Scheduling Application. The interface and functionality of the application will be constructed using ASP-VB Script. A IIS 6.0 web server in an internet environment (DMZ) with access to a SQL Server database will be utilized. The server must allow ASP pages to be served.

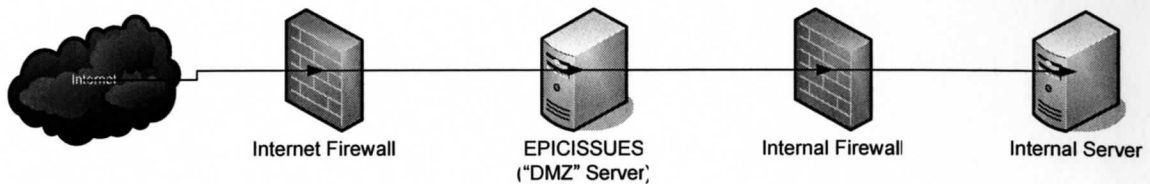


Figure 2. *Network diagram of the DMZ server*

This project was of limited scope and scope creep was not evident during the development of the application. Any additions or advanced functionality desired by the Sioux Valley Intensive Air Program will be addressed as requested.

Since I was unfamiliar with the use of ASP and its integration with HTML and JavaScript programming, I utilized the expertise of the Web Development Team to guide me through that process. Within the Project Specifications, I identified that the weekly and monthly schedules could be viewed interchangeably by clicking a toggle “switch”. This functionality was replaced by having different links within the main menu of “Weekly” and “Monthly”. Functionality not originally identified but subsequently added was a toggle switch included within both the weekly and monthly schedules which allows notes to be presented and hidden at the user’s discretion. Also, an additional email will be sent back to the person making the change as a reminder to themselves of the schedule change.

I will be involved in implementation of the schedule for Sioux Valley Intensive Air Program. The pilots will need to enter scheduling information based on the upcoming schedule. Monitoring to ensure that the schedule is copied forward as requested will be done as needed. End-user training will be offered if requested. Since the application is very intuitive and doesn’t involve elements beyond assigning pilots to a

specific shift in a one-to-one relationship, user acceptance of the application is not anticipated to be a significant factor.

Baseline numbers on the amount of time being spent by pilots and support staff on scheduling-related activities were not readily available. Efficiency gained through this application will need to be obtained from the Intensive Air Program after implementation.

A Work Breakdown Structure has been created to organize the tasks essential for successful completion of the project. Figure 3 depicts the Work Breakdown Structure in tabular form while Figure 4 depicts the Work Breakdown Structure in graphical form.

Microsoft Excel WBS and Gantt chart for SV Web-based application - INFS 788, Chad Breidenbach

File Edit View Insert Format Tools Data Window Help Type a question for help

100%

Arial

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Work Breakdown Structure															
2	for the Sioux Valley Intensive Air Web-Based Scheduling Application															
3	INFS 788-Project Planning															
4	Chad Breidenbach															
5																
6	1.0 Concept															
7	1.1 Evaluate current systems															
8	1.2 Define requirements															
9	1.2.1 Define user requirements															
10	1.2.2 Define content requirements															
11	1.2.3 Define system requirements															
12	1.3 Define specific functionality															
13	1.3.1 Pilots will go to a specific URL to access the application with a specific user name and password.															
14	1.3.2 The pilots will be able to view, print and submit updates to the schedule.															
15	1.3.3 Upon submitting an update request, an email will be generated and sent to selected pilots and administrator to alert them to the scheduling request.															
16																
17	1.4 Develop project plan															
18	1.5 Brief other member of the Sioux Valley Web Development and Programming team															
19	2.0 Web Site Design															
20	2.1 Collaborate with a graphics specialist and end users to create a user-friendly look and feel for the application.															
21	3.0 Web Site Development															
22	3.1 The site will be developed using ASP pages written in VBScript.															
23	3.2 JavaScript will be used as the client-side scripting language															
24	3.3 Utilize IIS 6.0 web server that serves ASP pages in an internet environment (DMZ) with access to a SQL Server database.															
25	4.0 Roll Out															
26	4.1 Get feedback from end users															
27	5.0 Support															
28	5.1 Technical support of the site will be provided by the Sioux Valley Information Technology Department.															
29																
30																
31																
32																
33																
34																

WBS / Gantt Chart /

Ready

Figure 3. The Work Breakdown Structure in tabular form is used to organize the project into smaller “work packages” or manageable tasks.

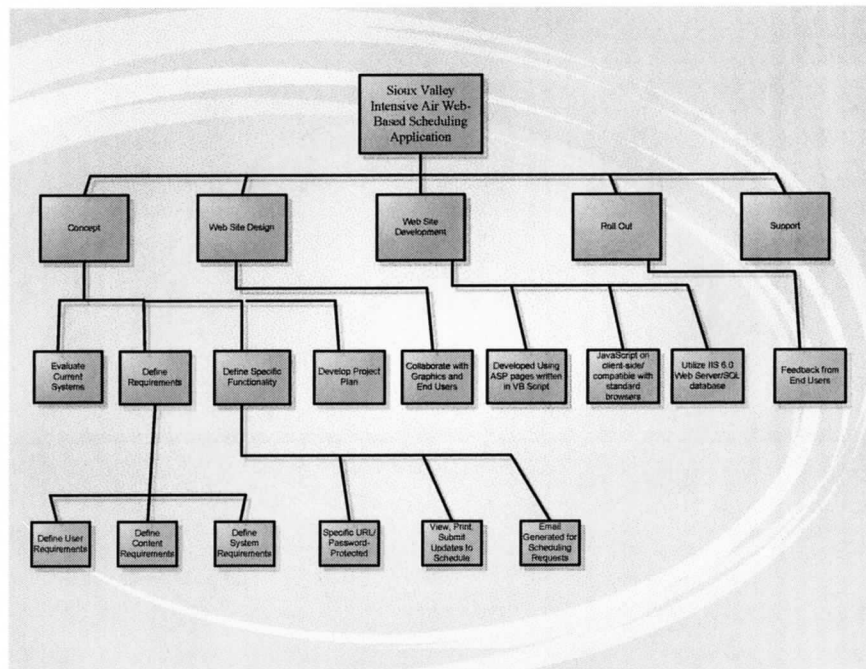


Figure 4. Diagram of the Work Breakdown Structure for the Sioux Valley Intensive Air Web-Based Scheduling Application.

CHAPTER 2

LITERATURE REVIEW

In undertaking the project, I thought it was prudent to analyze how the identified problem is typically addressed. Through an internet search, I identified three external vendors who offered flight scheduling applications. I contacted them to determine if the functionality of their applications was similar to what was being requested by the Sioux Valley Intensive Air Program. Once appropriate functionality was determined, I gathered information such as price, length of obligation, etc.

eFlightSchedule.com offers a flight scheduling program intended primarily for flight instruction programs. Pilots are assigned to specific aircraft at specific times. In order for pilot scheduling to occur, a pilot must be designated as a CFI (Certified Flight Instructor) to a specific aircraft and each specific aircraft would have to be designated as ground instruction. This would require that each pilot on the same shift be assigned to one of the two aircraft within the "fleet". Since the shifts of pilots overlap (e.g. the first shift has two pilots scheduled from 4 a.m. to 4 p.m., the second has two pilots scheduled from 10 a.m. to 10 p.m.) there would be an inherent conflict in scheduling aircraft to pilots in the overlapping time frames. The application does offer email notification of flights, custom color coding of schedules and use of a custom logo for a specific organization.

The cost to use the eFlightSchedule.com scheduling application is a base rate of \$20 per month with a charge of \$5 per aircraft per month. Since the Sioux Valley

Intensive Air Program has two fixed-wing aircraft, the cost per month would total \$30 to use them as an external vendor. The commitment to use the application is on a month-to-month basis. Table 1 is a summary of the interview held with eFlightSchedule.com

President Eddie Rohwedder.

Table 1. *Summary of the interview with President Eddie Rohwedder.*

eFlightSchedule.com
 President: Eddie Rohwedder
 (949) 302-8726
 September 13, 2005

Flight scheduling program intended primarily for flight instruction programs. The pilots must be entered as CFI (Certified Flight Instructor). Two pilots can't be assigned to the same plane and need to be designated as ground instruction. The application offers email notification of flights, custom color coding of schedules and use of a custom logo for the organization. No annual commitment for this; on a month-to-month basis. The cost is a base rate of \$20 per month with a charge of \$5 per aircraft per month on a month-to-month basis. Monthly cost: \$30

SkyScheduler of Columbus, Ohio, offers an application that is also geared toward use in flight schools for flight training programs. As long as the pilots are entered in the SkyScheduler application as instructors, the system would work in scheduling pilots. This application offers more flexibility than the eFlightSchedule.com application in that pilots can be assigned to an aircraft group instead of having to designate specific pilots to specific aircraft. The SkyScheduler application also offers last-minute cancellation tracking, real-time dispatching and other features beyond what has been requested by the Sioux Valley Intensive Air Program. The application offers functionality similar to what

was requested such as email notification of scheduled flights and also a summary of a pilot's upcoming schedule.

The basic package would offer the requested functionality at the rate of \$10 per aircraft per month or \$20 for the program. This arrangement is also on a month-to-month basis and with a 5% discount for annual prepayment. Table 2 is a summary of the interview held with SkyScheduler President Mike Carr.

Table 2. *Summary of the interview with President Mike Carr.*

SkyScheduler

President: Mike Carr

Columbus, OH

(877) 759-7483

September 7, 2005

The online application is geared toward flight schools; as long as the pilots were entered in the system as instructors, the system would work in scheduling pilots.

The service is offered (and billed) month-to-month, and no long-term commitment is necessary.

Pilots can be assigned to an aircraft group instead of having to designate specific pilots to specific aircraft.

The application offers last-minute cancellation tracking, real-time dispatching and other features beyond what has been requested by the Sioux Valley Intensive Air Program.

The application offers email notification of scheduled flights and also a summary of a pilot's upcoming schedule.

The application is an online service with nothing to install or maintain.

The price for the basic package (which would function as a scheduler) is \$10/aircraft per month (or \$20/month for us).

The step-up program would cost \$20 per aircraft per month (\$40) and also include maintenance tracking.

The basic package would offer the requested functionality at the rate of \$10 per aircraft per month. Agreement is month-to-month with a 5% discount for annual prepayment.

Monthly cost: \$20

MyFBO.com of Danville, Virginia, offers an online scheduling application that is intended to schedule individual flights and not intended to be a flight scheduling application. As with the eFlightSchedule.com application, the system could be "tricked" to allow pilots to be scheduled to individual aircraft for individual flights. This

application also offers functionality beyond what has been requested such as maintaining pilot information, email notification of flights and tracking of aircraft maintenance data.

The package that would include the basic functionality needed would cost \$27 per month. Additional requirements can be accounted for and quoted via an online quote. Based on those additional requirements, the total cost per month would be \$46 with a month-to-month commitment. Table 3 is a summary of the interview held with MyFBO.com President Paul Liepe.

Table 3. *Summary of the interview with President Paul Liepe.*

MyFBO.com
 President: Paul Liepe
 Danville, VA
 (434) 793-6800
 September 12, 2005

The online scheduling application that is intended to schedule individual flights and not intended to be a flight scheduling application.

The system can be "tricked" to allow pilots to be scheduled to individual aircraft for individual flights.

This application also offers functionality beyond what has been requested such as maintaining pilot information and tracking of aircraft maintenance data, but does offer email notification of flights.

The software is an online service. There is nothing to install or maintain. The service is offered (and billed) month-to-month, and no long-term commitment is necessary.

You do not need to have a web site to use the online system. It can be used without any web site at all, or link to it from an existing web site.

The monthly base charge for the application is \$27. The total amount depends on a number of variables -- how many aircraft, how many instructors / captains, and whether the number of instructors exceeds the number of aircraft.

Quotes can be obtained online from <http://myfbo.com/myfbo/quote.asp>. Click "Classic Edition" and then enter the number of resources (aircraft and flight staff).

The quote obtained online based on the needs of the Sioux Valley Intensive Air Program was \$46 per month with a month-to-month commitment.

Build Instead of Buy

The applications that are available through external vendors would meet many of the requirements identified by the Sioux Valley Intensive Air Program with only slight

adjustments needed to accommodate pilot scheduling. With these applications available at a very reasonable cost with no lengthy commitment, it would seem that outsourcing the scheduling function would be a desirable option. However, the decision to create the Web-Based Scheduling Application in-house instead of purchasing this service was due to the following criteria:

The application was relatively straightforward and offered the functionality desired without having to purchase additional hardware or software. The resources to develop the application were already in place and the expense of developing and maintaining the application would be minimal to the organization.

The in-house application has a much simpler and intuitive design and will be more user-friendly, increasing the likelihood of end-user acceptance. The representative for the Sioux Valley Intensive Air Program indicated that some of the pilots are not technically skilled with computers and would need a very simple design. Editing pilot and shift information with the in-house application is less cumbersome to the end-users without having to worry about “assigning” aircraft to specific pilots, pilots to specific a group of aircraft, etc., that would be necessary with the functionality of the external applications.

Building this application in-house gives us the flexibility to develop functionality specific to the needs of our user group. The in-house application is customizable to work processes. Having the application in-house allows the Web Development Team to build any enhancements as needed to meet business requirements without having to restructure or adapt operations.

CHAPTER 3

REQUIREMENTS ANALYSIS

As mentioned in the beginning of this report, a scheduling application was identified as a priority project from the Project List maintained by the Sioux Valley Information Technology Web Development Team. Based on the documentation that they had gathered, I began the planning process by creating a Project Plan (Appendix A) and Work Breakdown Structure of the project. From the Work Breakdown Structure, I created a Gantt chart to establish timelines for completion of tasks necessary for completion of the Sioux Valley Intensive Air Web-Based Scheduling Application (Appendix A).

The next step in the process was to gather information on the end-user requirements. On August 30, 2005, I met with Sherwin Bolks, representative of the Sioux Valley Intensive Air Program, to determine what functionality was needed by the pilots. Table 4 is the Interview Report that lists the requirements for the application as requested by Sherwin Bolks.

Table 4. *The Interview Report with representative of the Sioux Valley Intensive Air Program Sherwin Bolks.*

Interview Report

Interview Notes Approved By: Chad Breidenbach

Person Interviewed:	Sherwin Bolks
Interviewer:	Chad Breidenbach
Date:	August 30, 2005

Table 4. *The Interview Report with representative of the Sioux Valley Intensive Air Program Sherwin Bolks.*

Primary Purpose: To determine current scheduling policy, confirm end-user requirements and content required for a web-based scheduling application. I also confirmed the expected functionality of the application.

Summary of Interview:

We discussed the essential elements that should be included in a web-based scheduling application. We also discussed the environment in which the pilots operate in relation to regulatory policy (FAA, etc.) and company policy (hours that a pilot may be on duty, Human Resources policies, etc.) that may affect the schedules of the pilots. I shared with him the timelines of the project and exchanged contact information so that we could maintain open lines of communication so that I could receive timely feedback on questions that arose during application development.

Open Items:

Listing of on-call pilots and “Notes section” for each day of the schedule; different colors for the different pilot shifts.

Detailed Notes:

- ✓ Currently, only pilots should have read/write access to the schedule.
 - ✓ Security concerns if there is an event that requires confidentiality (e.g. a plane crash, etc.).
 - ✓ There will be one or two super users, but the application will allow all pilots to view and edit schedules for all three shifts.
 - ✓ Any pilot affected by a change and the Program Manager will be notified of by email of that change. In the subject line, have the standard text that displays w/ date?
 - ✓ The change in the schedule does not need to be approved by anyone; the email is sent as a courtesy to the Program Manager to make him aware of any activity that would affect pay, vacation, sick leave, and all payroll/HR-related items.
 - ✓ The schedule established over a predetermined number of days will be repeated in the next schedule (e.g. the first scheduled 90 days will be repeated over the next 90 day schedule.) The subsequent schedule will be based on the initial preceding 90 days prior to any edits.
 - ✓ The web page containing the schedule will be refreshed every 10-60 minutes. (This is very flexible per Sherwin.)
 - ✓ The schedule should have flexibility to eventually edit pilots, shift change times, etc.
-

Those requirements were used to create a Project Charter that was presented to Sherwin Bolks and members of the Web Development Team to sign-off and ensure that all parties had a clear understanding of the project requirements. Figure 5 is a sample of the Project Charter. Table 5 is the Project Charter Attachment.

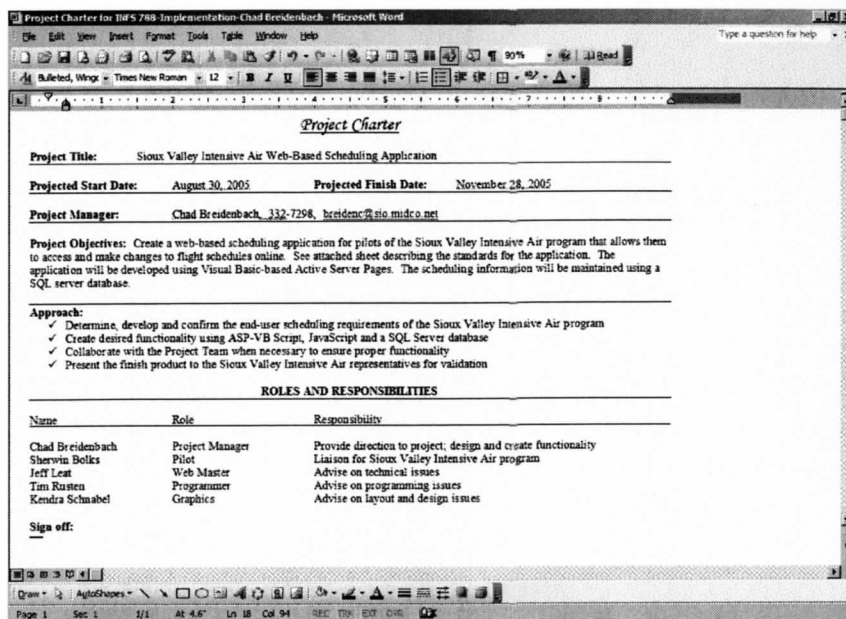


Figure 5. *The Project Charter was used to ensure that all parties had a clear understanding of the project requirements.*

Table 5. *Project Charter Attachment.*

Project Charter Attachment

Below are the requirements (standards) as determined in a validation session on August 30, 2005 with Sherwin Bolks, representative for the Sioux Valley Intensive Air program:

- ✓ Only pilots will have read/write access to the schedule.
- ✓ Security/privacy concerns exist in the event of an incident that requires confidentiality (e.g. a plane crash, etc.).
- ✓ There will be one or two super users, but the application will allow all pilots to view and edit schedules for all three shifts.
- ✓ Any pilot affected by a change and the Program Manager will be notified by email of that change.
- ✓ A change in the schedule does not need to be approved by anyone. An email would be sent as a courtesy to the Sioux Valley Intensive Air Program Manager to make him aware of any activity that would affect pay, vacation, sick leave, and all payroll/HR-related items.
- ✓ The schedule established over a predetermined number of days will be repeated in the next schedule (e.g. the first scheduled 90 days will be repeated over the next 90 day schedule.) The subsequent schedule will be based on the initial preceding 90 days prior to any edits.
- ✓ The web page containing the schedule should be refreshed every 10-60 minutes.
- ✓ The schedule should have flexibility to eventually edit pilots, shift change times, etc.
- ✓ Pilots would like to see a listing of on-call pilots and a notes section for each day of the schedule.

Once the requirements were agreed upon, I met with the Web Development Team to understand how they normally approach a project. After sharing the specific requirements given to me by the Intensive Air Program, the team and I began to brainstorm on how we would address the functionality needed for the application. Table 6 highlights the results of that meeting in the Project Specifications document.

Table 6. *The Project Specifications were established in a meeting with the Web Development Team to establish how to address the functionality needed for the application.*

*Project Specifications
for the Sioux Valley Intensive Air Web-Based Scheduling Application*

Banner name (to be the header on each page):

"Sioux Valley Fixed-Wing Schedule"

The user is presented with the Log In page.

The user logs in. The next page the user is guided to is a calendar which displays the next 7-8 days of the schedule (today's schedule and the next seven days.) A toggle link (switch) is available to allow the user to view the full schedule for the current month.

The user has an option to view future months using a drop-down menu with an onChange event. Each shift will be indicated by a different color.

When you click on a person's name, a window will appear allowing pilots to trade with one another.

The window should display the current pilot and a substitution pilot chosen by the user. This window will allow the user to swap pilots and shifts. The window will also include a field named: "Reason for swap"; with a drop-down menu and free text option.

Upon onClick of the submit button, the change identified in window will occur.

An email will then be sent to both pilots affected by a change and the Program Manager to notify them of that change. The subject line of the email will read:

"Schedule change as of (*today's date*)."

The body of email will default the following text:

Ex. "(Pilot requesting change) is rescheduled to (new date requested) at (schedule time requested)."

"(2nd pilot) is rescheduled to (scheduled date of the requesting pilot) at (scheduled shift time of the requesting pilot)."

(The pilot requesting change will be allowed to free-text any additional information.)

An option somewhere in the page above the calendar (link) will allow the user the option to print the schedule by week for the next seven days or by month. There will be a notes section by each day on the schedule.

Having more specifically identified the functionality of the application, Use Case Descriptions were written and a Use Case Diagram was created to address the main scenarios that would occur as a user interacts with the application. The Use Case Descriptions were shared with the project team and the representative from the Intensive Air Program to verify that we had addressed the essential elements needed for the application. Table 7 is a sample of the Use Case Descriptions. Figure 6 is the Use Case Diagram that was developed from the Use Case Descriptions.

Table 7. *Use Case Descriptions were developed to address the main scenarios that would occur as a user interacts with the application.*

USE CASE 1 – Enter pilot information

CHARACTERISTIC INFORMATION

Brief description

Only users with Administrator (superuser) privileges will be allowed to go into the Scheduling Application and enter/edit pilot information such as username, password, first name, last name, email address and security level.

Primary Actor

A superuser.

Stakeholders

Sioux Valley Intensive Air Pilots.

Trigger

Need to enter or edit pilot information for new and/or existing pilots.

Preconditions

The user needs to be entered in the Scheduling Application as a superuser having administrator privileges.

Guarantees

Success End Condition

Applicable pilot information is entered and the pilot is available in the system and can be scheduled.

Failed End Condition

Pilot information is not saved in the Scheduling Application and the pilot cannot be scheduled.

MAIN SUCCESS SCENARIO

1. Superuser provides username and password to the Scheduling Application.
2. Scheduling Application validates student according to log on rules.
- 3a. Scheduling Application displays the menu options to the superuser.
4. Superuser selects option from the menu to add a new user.
5. Superuser enters pilot-specific information and saves information.

EXTENSIONS

- *a. Superuser exits the Application:
- a1. Scheduling Application does not save entered information.
- 3b. Invalid log on:
- 3b.1. Scheduling Application notifies the pilot [repeat at 1]

Pilot Edit Rules

NUMBER	Rule Description: The Scheduling Application shall
LOG1	Verify if the pilot username is invalid
LOG2	Verify if the password for a pilot username is invalid

Table 7. *Use Case Descriptions were developed to address the main scenarios that would occur as a user interacts with the application. (Continued)*

USE CASE 2 - Add pilot to the schedule
CHARACTERISTIC INFORMATION

Brief description

A pilot with Administrator privileges (superuser) logs onto the Scheduling Application and enters a pilot into the system.

Primary Actor

A superuser.

Stakeholders

Sioux Valley Intensive Air Pilots.

Trigger

Superuser needs to add a pilot to the flight schedule.

Preconditions

The user needs to be entered in the system as superuser having Administrator privileges.

Guarantees

Success End Condition

The superuser has entered the pilot into the schedule.

Failed End Condition

The superuser doesn't enter pilot into the schedule and the Scheduling Application is unchanged.

MAIN SUCCESS SCENARIO

3. Superuser provides username and password to the Scheduling Application.
4. Scheduling Application validates user according to log on rules.
- 3a. Scheduling Application displays the menu options to the superuser.
5. Superuser selects option from the Main Menu to add a pilot to the schedule.
5. Superuser selects the desired date for scheduling the pilot from a pop-up calendar.
6. Superuser enters pilot-specific information and saves information.

EXTENSIONS

*a. Superuser exits the application:

- a1. Scheduling Application does not save entered information.

3b. Invalid log on:

- 3b.1. Scheduling Application notifies the pilot [repeat at 1]

Pilot Edit Rules

NUMBER	Rule Description: The Scheduling Application shall
LOG1	Verify if the pilot username is invalid
LOG2	Verify if the password for a pilot username is invalid

Table 7. *Use Case Descriptions were developed to address the main scenarios that would occur as a user interacts with the application. (Continued)*

USE CASE 3 – Change a pilot shift
CHARACTERISTIC INFORMATION

Brief description

A pilot logs onto the system and takes the shift of another pilot.

Primary Actor

A pilot.

Stakeholders

Sioux Valley Intensive Air Program.

Trigger

Pilot needs to take a shift of another pilot on the schedule.

Preconditions

The pilot who is executing the change and the pilot being replaced are in the system.

Guarantees

Success End Condition

The pilot has taken the shift of another pilot and the schedule is updated.

Failed End Condition

The pilot is unable to complete the change and the system is unchanged.

MAIN SUCCESS SCENARIO

5. Pilot provides username and password to the scheduling application.
6. Scheduling Application validates user according to log on rules.
- 3a. Scheduling Application displays the menu options to the pilot.
6. Pilot selects either the 'Weekly' or 'Monthly' option from the menu.
5. Pilot selects the name of the pilot on the shift desired.
7. Scheduling Application presents a pop-up window that includes scheduling information and the pilots involved in the schedule change.
8. Pilot saves information entered including the pilot taking the shift and notes.
9. Scheduling Application displays the schedule with updated shift information.
10. Scheduling Application emails notification of the change to the pilot whose schedule was affected and the Program Manager.

EXTENSIONS

- *a. Superuser exits the application:
- a1. Scheduling Application does not save entered information.
- 3b. Invalid log on:
- 3b.1. Scheduling Application notifies the pilot [repeat at 1]

Pilot Edit Rules

NUMBER	Rule Description: The Scheduling Application shall
LOG1	Verify if the pilot username is invalid
LOG2	Verify if the password for a pilot username is invalid

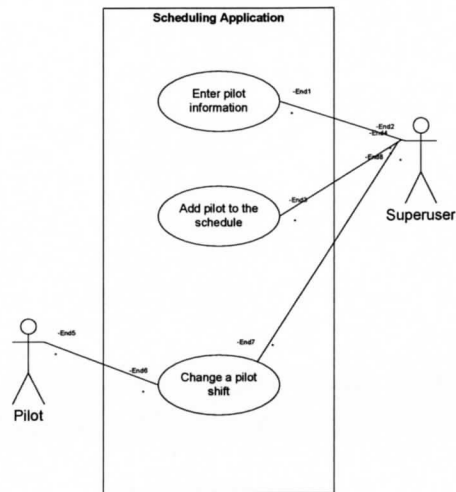


Figure 6. *The Use Case Diagram is a graphical depiction derived from the Use Case Descriptions.*

After the Use Case Descriptions were written and the Use Case Diagrams were created, a Plan of Action was developed to summarize the approach needed to develop the application (Appendix C).

CHAPTER 4

SYSTEM DESIGN

The first major task of system design was to develop a window navigation diagram (WND) based on the Use Case Descriptions. Figure 7 is a sample of the diagram used to illustrate the expected functionality of the application.

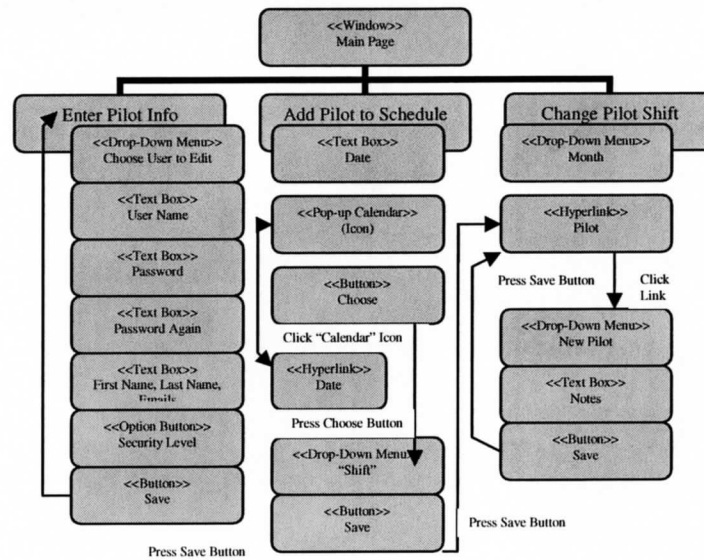


Figure 7. *Window Navigation Diagram.*

Once the window navigation diagram was completed, data requirements for the application were established. After I identified the essential data requirements for the application, I reviewed these with Jeff Leat to determine if all attributes needed for the database tables were appropriate. Table 8 shows the essential data elements needed for the application.

Table 8. *Data tables for the scheduling application.*

<u>Configuration</u>		
<u>Attribute</u>	<u>Data Type</u>	
Item	VARCHAR(50)	NOT NULL,
Value	VARCHAR(500);	
<u>Schedule Information</u>		
<u>Attribute</u>	<u>Data Type</u>	
ScheduleID	INTEGER(4)	NOT NULL,
User ID	INTEGER(4),	
Shift	DATETIME(8);	
<u>Schedule Modifications</u>		
<u>Attribute</u>	<u>Data Type</u>	
ModID	INTEGER(4)	NOT NULL,
ScheduleID	INTEGER(4),	
UserID	INTEGER(4),	
NewPilot	INTEGER (4),	
Changed	DATETIME(8),	
Notes	VARCHAR(1000);	
<u>User Information</u>		
<u>Attribute</u>	<u>Data Type</u>	
UserID	INTEGER(4)	NOT NULL,
UserName	VARCHAR(50),	
Pass	VARCHAR(50),	
FName	VARCHAR(50),	
LName	VARCHAR(50),	
LastOn	DATETIME(8),	
SecurityLevel	INTEGER(4);	

Once the data requirements were identified, database tables were created using MS SQL Server 2000. Figure 8 is a sample of those tables.

2:Design Table 'Configuration' in 'FixedWingSchedul...				
Column Name	Data Type	Length	Allow Nulls	
sItem	varchar	50		
sValue	varchar	500		✓

3:Design Table 'Schedule' in 'FixedWingSchedule' on '(L				
Column Name	Data Type	Length	Allow Nulls	
iScheduleID	int	4		
iUserID	int	4		✓
dShift	datetime	8		✓

4:Design Table 'ScheduleModifications' in 'FixedWingSc				
Column Name	Data Type	Length	Allow Nulls	
iModID	int	4		
iScheduleID	int	4		✓
iUserID	int	4		✓
iNewPilot	int	4		✓
dChanged	datetime	8		✓
sNotes	varchar	1000		✓

5:Design Table 'Users' in 'FixedWingSchedule' on '(LOC				
Column Name	Data Type	Length	Allow Nulls	
iUserID	int	4		
sUserName	varchar	50		✓
sPass	varchar	50		✓
sFName	varchar	50		✓
sLName	varchar	50		✓
dLastOn	datetime	8		✓
iSecurityLevel	int	4		✓

Figure 8. Database tables in MS SQL Server 2000.

A database schema and Entity-Relationship (E-R) Diagram were created to show the interrelationships of the database tables. They are shown in Figure 9 and Figure 10, respectively.

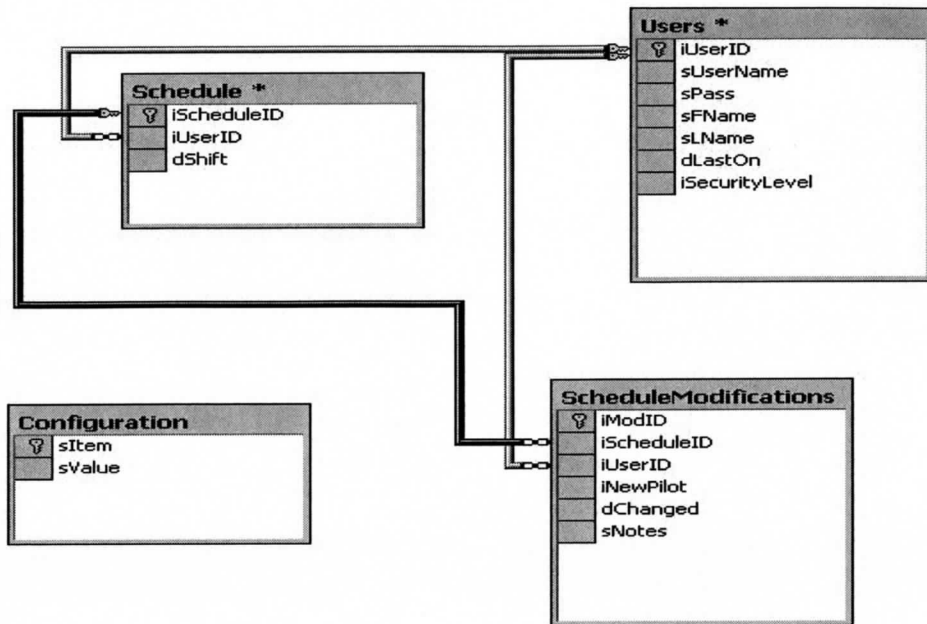


Figure 9. Database Schema.

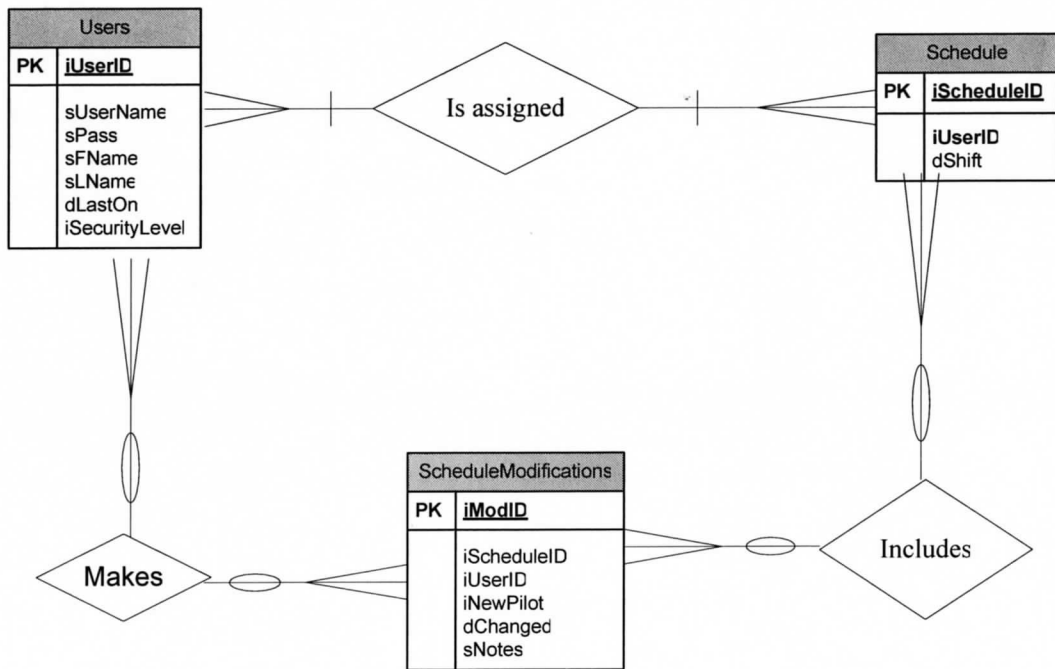


Figure 10. Entity-Relationship (E-R) Diagram

The next step was to develop the user interface (GUI) using HTML and JavaScript (Appendix C). ASP code was used so that the application could dynamically interact with the MS SQL Server 2000 database.

VB.net and ASP.net are the most current versions of the technology that was used to develop this application. However, the Sioux Valley Web Development Team is still using ASP 3.0 as their main development platform.

The Sioux Valley Web Development Team will begin using VB.net and ASP.net in the spring or summer of 2006. The high current volume of projects, the team's familiarity with legacy technology, and the implementation of higher priority projects such as the EPIC Information System, has slowed the team's transition to the newer technology.

Once the user has logged in, the application defaults to the weekly view of the pilot schedule for the current week. Figure 11 is a screenshot of the home page (weekly view) for the Web-Based Scheduling Application.

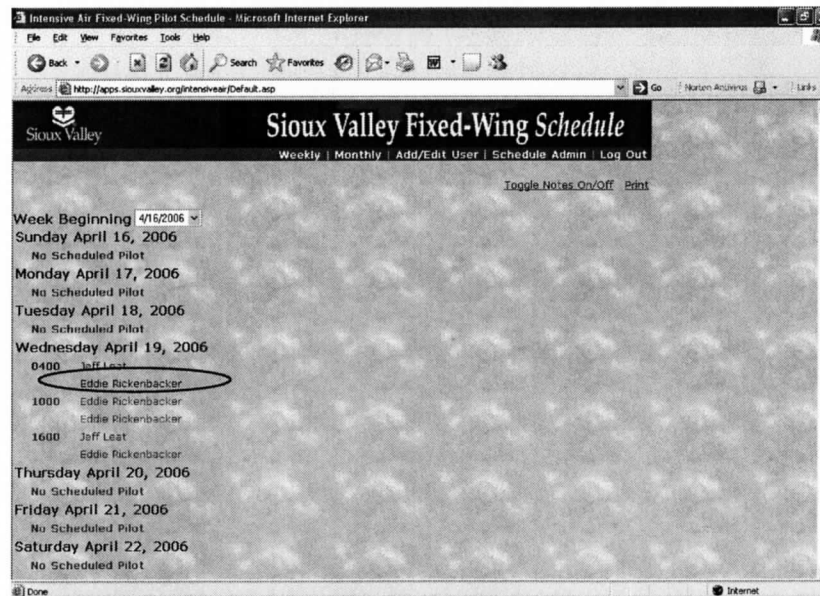


Figure 11. Home page of the Web-Based Scheduling Application.

The schedule may also be viewed as a monthly schedule. Figure 12 is a screenshot of the monthly schedule.

Sioux Valley Fixed-Wing Schedule

Weekly | Monthly | Add/Edit User | Schedule Admin | Log Out

Toggle Notes On/Off Print

May 2006

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
30	1 0400: Steve 1105 Chris 1117 1000: Rick 1107 Mark 1113 1600: Jeff 1114 Shane 1111	2 0400: Steve 1105 Tim 1112 1000: Rick 1107 Mark 1113 1600: Mike 1110 Shane 1111	3 0400: Dick 1106 Arlen 1116 1000: Gary 1109 Arlen 1116 1600: Shane 1111 Steve 1105	4 0400: Dick 1106 Sherwin 1103 1000: Gary 1109 Arlen 1116 1600: Mike 1110 Steve 1105	5 0400: Dick 1106 Sherwin 1103 1000: Arlen 1116 Tim 1112 1600: Shane 1111 Chris 1117	6 0400: Mark 1113 Sherwin 1103 1000: Arlen 1116 Tim 1112 1600: Shane 1111 Chris 1117
7 0400: Dick 1106 Shane 1111 1000: Arlen 1116 Tim 1112 1600: Shane 1111 Chris 1117	8 0400: Mike 1110 Gary 1109 1000: Dick 1106 Jeff 1114 1600: Mark 1113 Shane 1111	9 0400: Mike 1110 Gary 1109 1000: Dick 1106 Steve 1105 1600: Mark 1113 Shane 1111	10 0400: Jeff 1114 Sherwin 1103 1000: Chrs 1117 Steve 1105 1600: Mark 1113 Shane 1111	11 0400: Mike 1110 Sherwin 1103 1000: Chris 1117 Steve 1105 1600: Arlen 1116 Tim 1112	12 0400: Jeff 1114 Sherwin 1103 1000: Gary 1109 Dick 1106 1600: Steve 1105 Mark 1113	13 0400: Jeff 1114 Sherwin 1103 1000: Gary 1109 Dick 1106 1600: Steve 1105 Mark 1113
14 0400: Jeff 1114 Sherwin 1103 1000: Chris 1117 Dick 1106 1600: Steve 1105 Mark 1113	15 0400: Mike 1110 Arlen 1116 1000: Dick 1106 Gary 1109 1600: Jeff 1114 Tim 1112	16 0400: Mike 1110 Rick 1107 1000: Dick 1106 Gary 1109 1600: Shane 1111 Tim 1112	17 0400: Rick 1107 Mark 1113 1000: Gary 1109 Sherwin 1103 1600: Steve 1105 Shane 1111	18 0400: Rick 1107 Mark 1113 1000: Jeff 1114 Sherwin 1103 1600: Steve 1105 Shane 1111	19 0400: Dick 1106 Tim 1112 1000: Arlen 1116 Mike 1110 1600: Chris 1117 Shane 1111	20 0400: Dick 1106 Tim 1112 1000: Arlen 1116 Mike 1110 1600: Chris 1117 Shane 1111
21 0400: Dick 1106 Tim 1112	22 0400: Dick 1106 Jeff 1114	23 0400: Dick 1106 Rick 1107	24 0400: Sherwin 1103 Rick 1107	25 0400: Sherwin 1103 Rick 1107	26 0400: Steve 1105 Sherwin 1103	27 0400: Steve 1105 Sherwin 1103

Figure 12. Monthly schedule.

The HTML coding used to create the calendar can be viewed in Appendix C under the title “Calendar Code”. When a schedule change is to be made, the user must simply click on the name of the pilot whose shift is to be changed. When this occurs, the following window will appear as in Figure 13.

http://apps.siouxvalley.org - Intensive Air Fixed-Wing Pilot Sch...

Schedule Date: 4/19/2006
 Schedule Time: 4:00 AM
 Scheduled Pilot: Eddie Rickenbacker
 New Pilot: b, chad
 Notes
 SAVE

Figure 13. The window used to complete a shift change.

At this point, any information that is entered in the New Pilot field and the Notes section is retained. The processing of this information is done by the *modifySchedule.asp* (Appendix C) module. Once the ASP code has processed this information, it appears on the HTML-based schedule. Figure 14 shows the schedule with the appropriate change made.

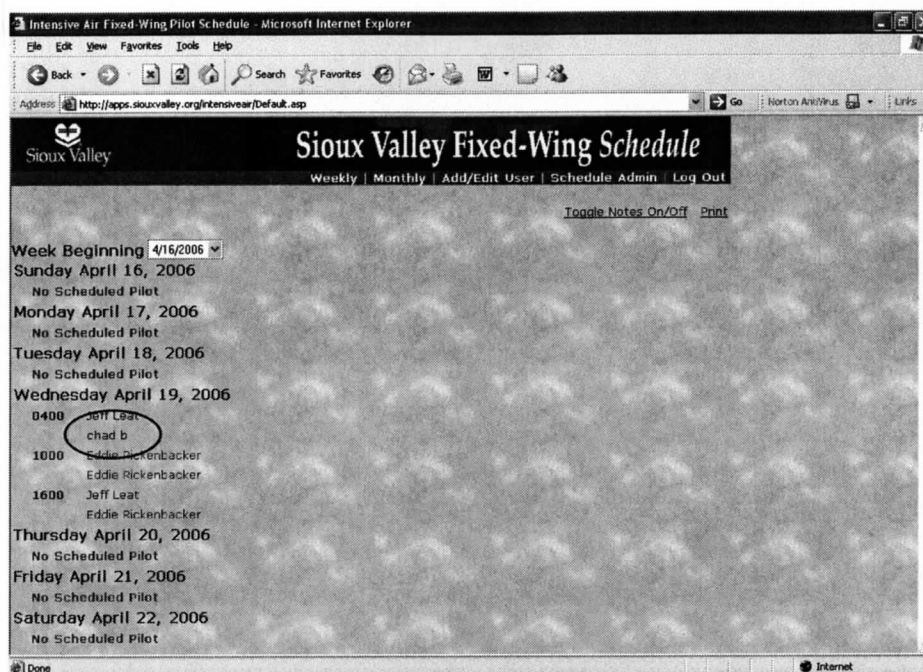


Figure 14. Weekly schedule showing a completed shift change.

After the change is being processed, an email is also being created and sent by the *functions.asp* (Appendix C) module. The information related to the schedule change is placed on an email and sent out to the pilots involved in the shift change and the Program Manager.

For a user to enter pilot information and create a new pilot in the system, the “Add/Edit User” hyperlink must be selected. When a pilot is entered into the system, the information pertaining to that pilot is entered on the form depicted in Figure 15.

Intensive Air Fixed-Wing Pilot Schedule - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Print Mail

Address http://apps.siouxvalley.org/intensiveair/admin.asp

Sioux Valley

Sioux Valley Fixed-Wing Schedule

Weekly | Monthly | Add/Edit User | Schedule Admin | Log Out

Choose User to Edit: New User

Username:

Password:

Password Again:

First Name:

Last Name:

Emails:

separate by semicolon(,)

Security Level: Normal User Administrator

SAVE

Figure 15. The form used to enter pilot information.

Only users assigned the security level of Administrator will be able to enter information on this HTML screen. When the user presses the Save button, the information is passed to the *admin.asp* (Appendix C) module for processing and entered into the database.

To add pilots onto a schedule, the user must click the “Schedule Admin” hyperlink. When selected, the following HTML screen will appear as depicted in Figure 16.

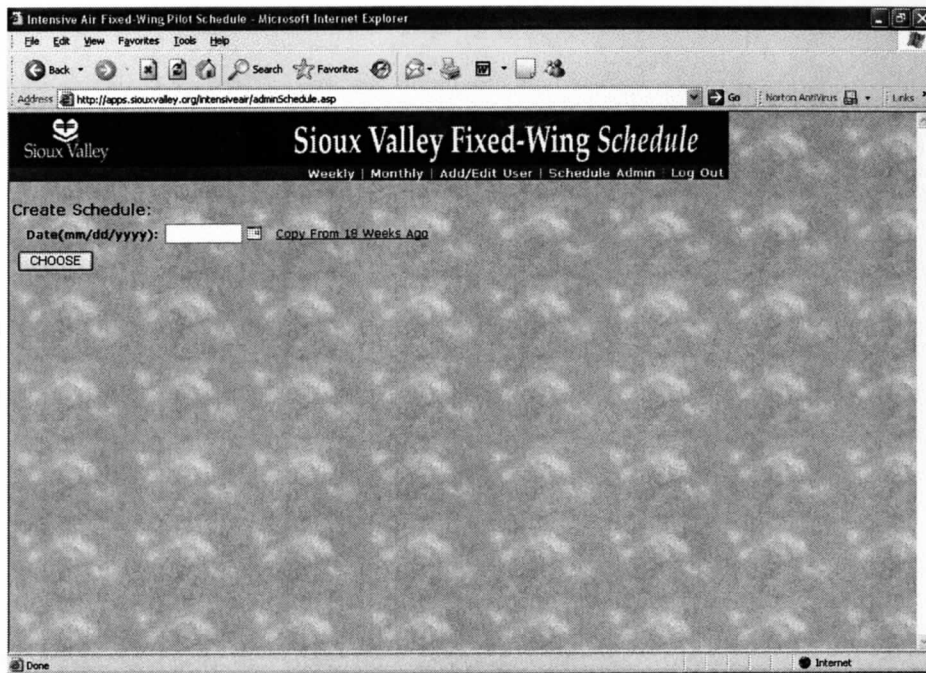


Figure 16. *HTML screen used to enter pilots onto a schedule.*

The user must then either type the date of the schedule to be populated or click the calendar icon to be presented with a calendar with a grid of dates from which the user may choose. The grid is will appear as seen in Figure 17.



Figure 17. *Calendar used to select a schedule date.*

The specific schedule date to be entered from either option and the Choose button is pressed to enter that value in the text box. When the specific date of the schedule to be

changed is selected, the user will be presented with a window that has a series of fields with drop-down menus (Figure 18).

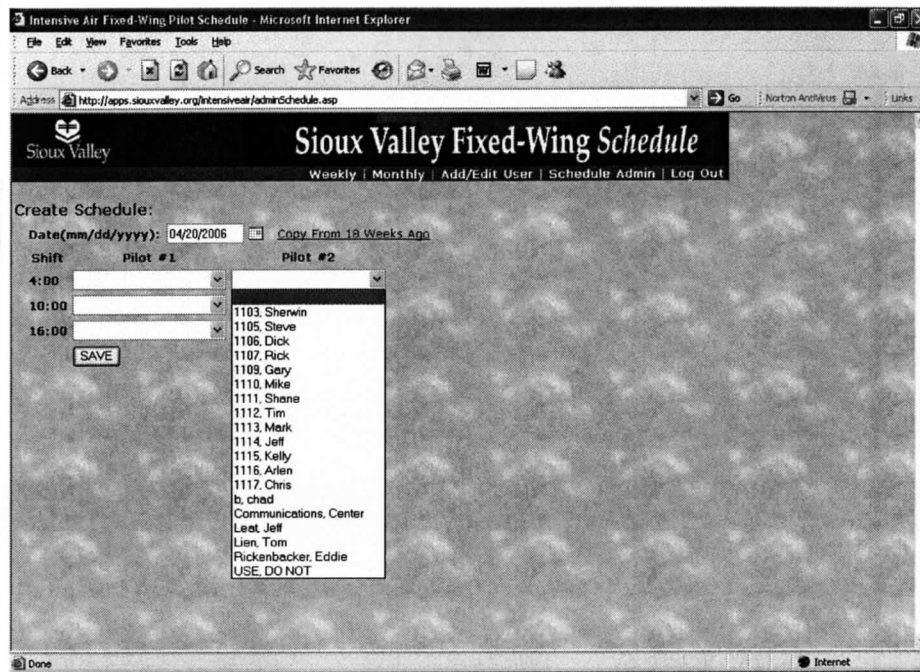


Figure 18. *The window that contains a series of fields with drop-down menus.*

If the schedule has previously been populated, that day's schedule will appear, but edits cannot be completed on the pilots that appear. When all shifts of the schedule have been filled, the user presses the Save button and the entered information will be processed by the *adminSchedule.asp* (Appendix C) module on the server.

Upon completion of the application, testing was undertaken to verify that the application provided the functionality initially desired. The application was subsequently demonstrated to the representative of the Sioux Valley Intensive Air Program. He indicated that the application provided the precise functionality they requested in a format that would greatly enhance their operation.

Upon approval of the Web-Based Scheduling Application by the program representative, I proceeded to create a User's Guide to assist the pilots in using the

application. Upon rollout of the application, he indicated that he will demonstrate the application to the pilots and share the User's Guide (Appendix B) with them at that time.

The final step that needs to occur before the application can be put into a "go-live" environment, is approval by the Sioux Valley Information Technology Department Change Advisory Board. It is anticipated that the approval will occur in January or February of 2006.

CHAPTER 5

IMPLEMENTATION AND RESULTS

The implementation of the Web-Based Scheduling Application was installed into a production environment on March 1, 2006. To gain some familiarity with the application, pilots began testing the application as a prototype the following week on Monday, March 6. About a month later, the pilots received training on the application. Upon receiving training, they were asked what modifications should be made to improve the application. The following suggestions were received and subsequently added to the application:

- Change displayed time from regular time to military time.

- Change the font of the printed monthly schedule to fit on one printed page.

- Configure the automated "Copy Schedule" link that copies the existing schedule from 13 weeks in the past forward so that it copies the schedule from 18 weeks in past forward.

- Hide the "Copy Schedule" link so that the Administrator may not be able copy forward an existing schedule if there are already pilots scheduled in those time slots.

On May 1, 2006, the Web-Based Scheduling Application was put into use by the pilots of the Sioux Valley Intensive Air Program.

Users of the application have been very pleased with the new web-based schedule and have remarked that it has saved them a great deal of time by having the ability to

view and make changes to the existing schedule via the Internet. They are unable to quantify the exact amount of time it has saved them and no monetary numbers have been established to determine cost savings.

CHAPTER 6

CONCLUSIONS

Was the Objective Achieved?

I believe that the deliverables established at the beginning of this project were achieved. The project could be qualified as small and limited in scope. The project was well-defined by the customers and enhancements were not requested by them. Scope creep did not occur during this process and did not necessitate any work beyond what was needed for the basic functionality of the application

Anticipated Deliverable

The anticipated deliverable was achieved and created to the satisfaction of the end-users. When I met with the representative of the Sioux Valley Intensive Air Program, Sherwin Bolks, on Friday, December 2, 2005, to demonstrate the Web-Based Scheduling Application. He expressed his satisfaction with the application and stated that it was exactly what they were looking for. His response was the following: "Thanks a ton for your efforts. I am very impressed with the product. Not only will this application be very usable for our operation but for the employees that are computer-challenged, it will be very easy for them to use as well. I am very excited to get this product in use, eliminating the need for paper schedules with changes written on them in

several locations". Subsequent feedback noted in Chapter Five verifies that the application has significantly met the expectation of the end-users of the application.

Lessons Learned

This project was a review of HTML and JavaScript programming. However, the use of these languages was limited in relation to the total scope of the build. The majority of the coding was done using ASP code. ASP uses a VBScript variation of Visual Basic used for scripting. The Web Development Team was very accommodating with their time in helping me learn and understand how ASP code is written and in the development of an interactive web application.

Future Work

The representative for the Sioux Valley Intensive Air Program mentioned two future enhancements to the Web-Based Scheduling Application. He indicated that he would like to see the mechanic on-call on the schedule. He also mentioned that he would like to see a shift on the schedule for one pilot that runs from 10 p.m. to 10 a.m.

Other possible enhancements would be the addition of a constraint which would not allow a pilot to be scheduled more than once in the same day or within a certain period of time from another shift in which the pilot is already scheduled. This was intentionally left out of this phase of development at the verbal request of the representative of the Intensive Air Program. He and I both agreed that we should initially configure the application with fewer constraints to allow more flexibility and if scheduling issues arose, we would address them as needed.

In our follow-up meeting where I demonstrated the application, the representative mentioned that the Program Manager is responsible for reporting pilot hours worked at regular intervals to Human Resources. Functionality that would tabulate pilot hours worked for certain intervals of time and put that information into a report format could be proposed and addressed in the future.

As indicated in Chapter Five, specific features have been added to the application to make it more user-friendly and adaptable to their work environment.

Any requested enhancements will be addressed in the next phase by the Web Development Team. The current application addresses the immediate needs of the pilots of the Sioux Valley Intensive Air Program.

REFERENCES

Sioux Valley Hospitals and Health System-History. (n.d.). Retrieved September 12, 2005 from <http://www.siouxvalley.org/AboutSiouxValley/Hospital%5CHistory.cfm>

Sioux Valley Hospitals and Health System-Intensive Air. (n.d.). Retrieved September 12, 2005 from <http://www.siouxvalley.org/CentersofExcellence/Trauma/IntensiveAir/Index.cfm>

APPENDICES

Appendix A: Project Planning

The Project Plan was developed in INFS 788: Project Planning and was used to identify a project that addressed a business need for the organization.

*Project Plan for
Sioux Valley Hospital Intensive Air Web-Based
Scheduling Application
Submitted by
Chad Breidenbach
INFS 788-Project Planning
August 12, 2005*

Introduction

The Sioux Valley Intensive Air program has been operating since 1977 and has flown more than 15,000 patients. Intensive Air was the first CAMTS (Commission on Accreditation of Medical Transport Services) accredited flight program in South Dakota. Operations include a Bell 230 helicopter and two King Air 200 fixed-wing aircraft. Intensive Air is staffed with four specialized flight teams. Currently, there are 12 pilots who fly for the Sioux Valley Hospital Intensive Air Program.

Problem Statement

The pilots of the Sioux Valley Hospital Intensive Air Program currently schedule manually on a scheduling sheet. When pilots are in a remote location they must call in to verify when they are on duty. This makes it very cumbersome to view or adjust current schedules. They are seeking a more efficient way of scheduling, updating existing schedules and having immediate access to scheduling information.

Objective

The pilots of the Sioux Valley Hospital Intensive Air Program wish to implement a web-based scheduling application that allows them to view and update scheduling information online, thereby increasing the efficiency of their off-hours labor.

The scope of this project will include the development of a web-based scheduling application. The interface and functionality of the application will be constructed using

ASP-VB Script. A IIS 6.0 web server in an internet environment (DMZ) with access to a SQL Server database will be utilized. The server must allow ASP pages to be served.

The Plan of Action will be to complete the programming for the programming staff at Sioux Valley to provide the Intensive Air pilots the requested functionality. I will be the primary provider of Layout, Programming and Database development for this application. I will also collaborate with a designated graphics person within the Sioux Valley Information Technology department.

Specific functionality of the web-based application will include the following:

- ✓ The schedule established over a predetermined number of days will be repeated in the next schedule (e.g. the first scheduled 90 days will be repeated over the next 90 day schedule).
- ✓ Any pilot affected by a change in the schedule will be notified by email of that change.
- ✓ The web page containing the schedule will be refreshed every 10 minutes.
- ✓ There will be one or two super users, but the application will allow all pilots to view and edit schedules for all three shifts.

The end-result will be a very user-friendly and efficient scheduling application, thus increasing efficiency and job satisfaction for the pilots of the Sioux Valley Hospital Intensive Air Program.

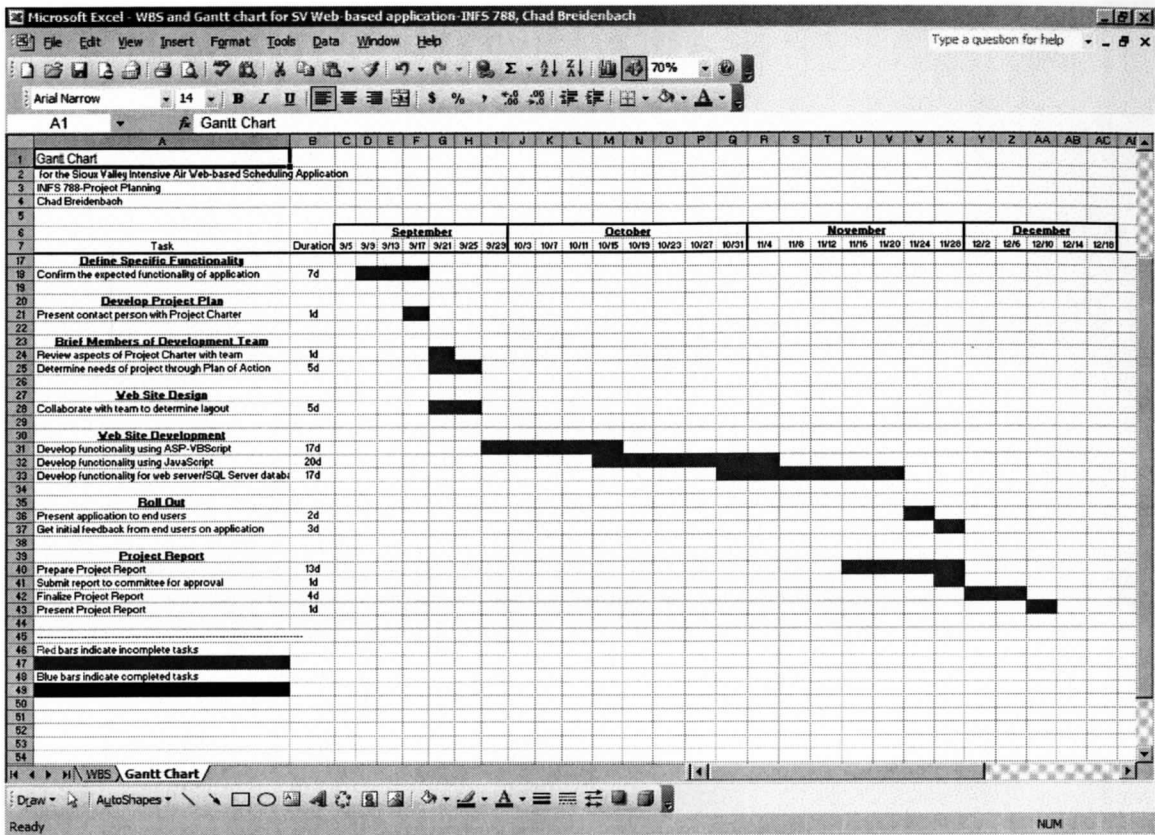
Work Breakdown Structure

See Visio and Excel attachment

Gantt Chart

See Excel attachment

The Gantt chart was created to establish timelines for the completion of tasks.



Appendix B:

USER'S GUIDE

A User's Guide was developed for the pilots of the Sioux Valley Intensive Air Program as a quick reference guide to the Web-Based Scheduling Application.

User's Guide



Sioux Valley Hospital
USD Medical Center

*Sioux Valley Fixed-Wing Intensive Air Program
Web-Based Scheduling Application*

Table of Contents



Introduction	3
How to Log On	4
How to Change Shifts.....	5

Introduction

Welcome!!!

This User's Guide includes instructions which will assist you in utilizing the Web-Based Scheduling Application developed for the Fixed-Wing Sioux Valley Intensive Air Program.

The application was designed to be very easy to use. However, if you should run into any problems, this guide will assist you by giving you some insight on how it should function and additional resources available.

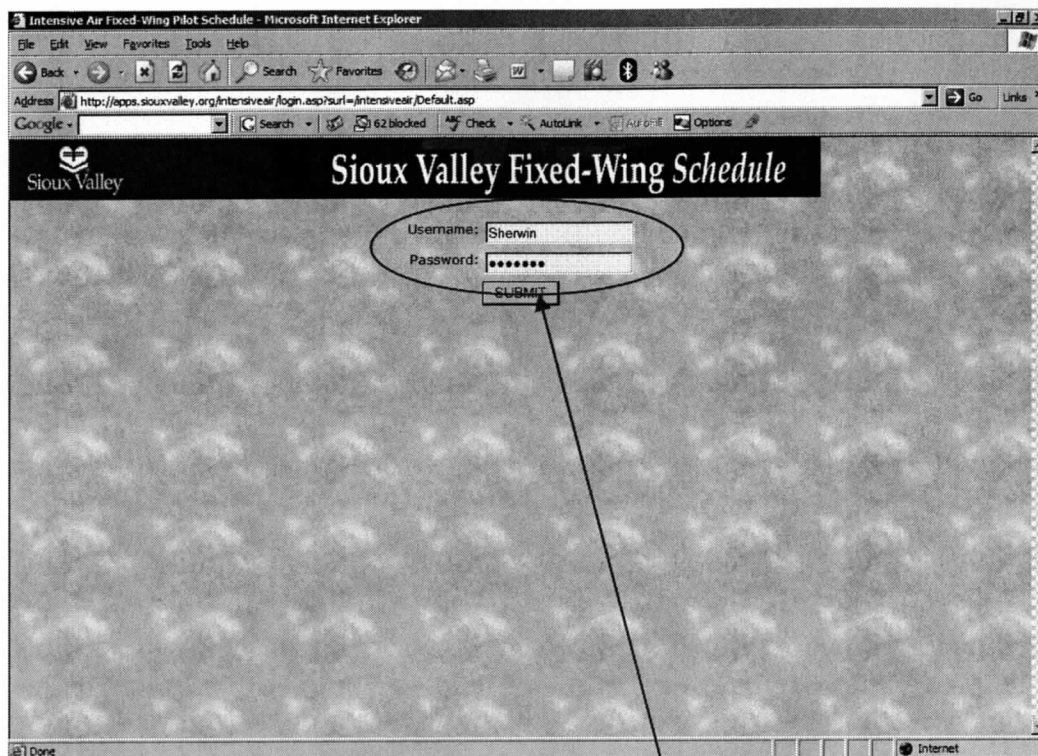
Good luck with the application!

Sioux Valley Web Development Team

How to Log On



The link to the log on screen of the scheduling application is <http://apps.sioxvalley.org/intensiveair/login.asp?surl=/intensiveair/Default.asp>

Once you select the link, the Home Page of the application should appear as follows:



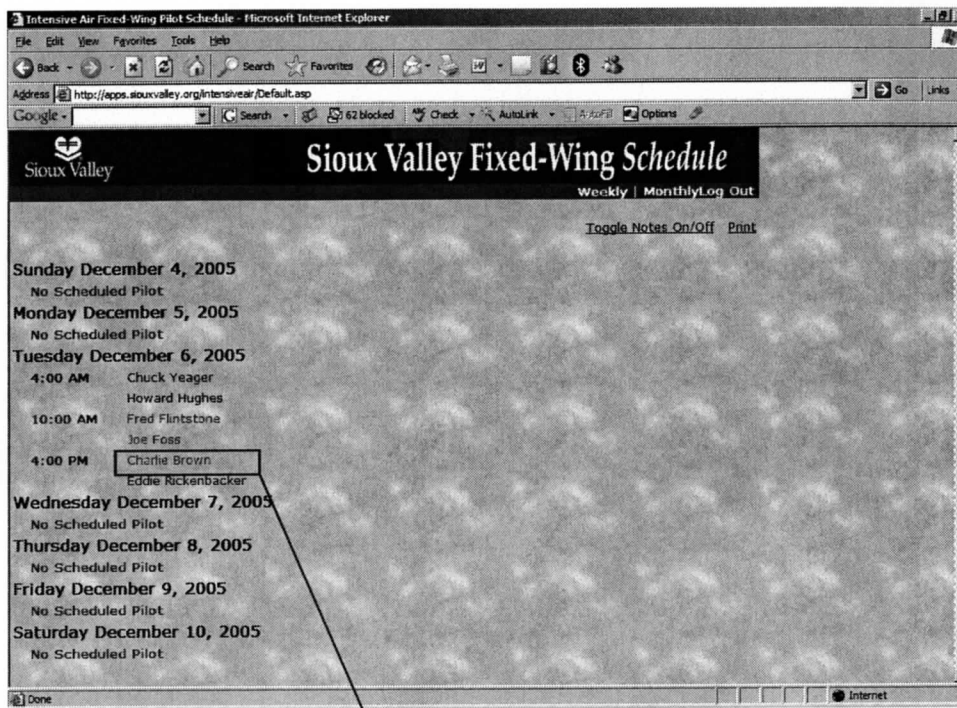
Once you have entered your user Username and Password, click the "Submit" button and you will be taken to the home page.

Note:

-  *Username and Password are not case-sensitive (upper-case or lower-case letters may be used).*
-  *Your Username and Password will be provided to you by Sherwin Bolks.*

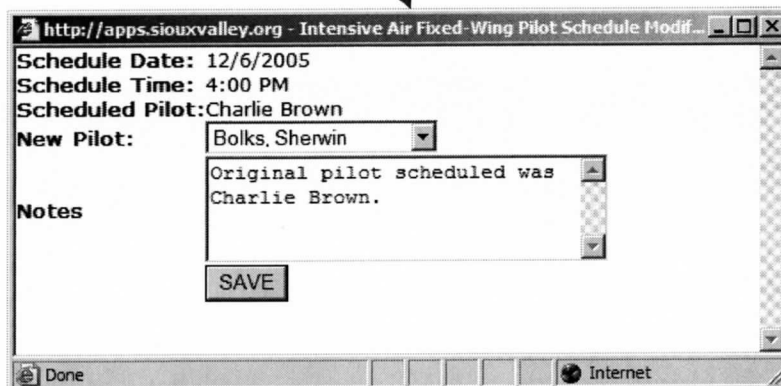
How to Change a Pilot Shift

The Home Page of the application will appear as follows:



You will first see a 7-day (weekly) starting with current day. To select a shift that you wish to take, simply click on the name of the pilot that occupies that shift.

A pop-up window will appear with the Schedule Date, Schedule Time, Scheduled Pilot and New Pilot.



Note:



The logged-in user's name will appear in the New Pilot field. However, any pilot can make a change for any other pilot on the schedule.

In the Notes section, make sure to identify the name of the pilot who originally occupied the shift you took for record-keeping purposes.

You may also type in any additional information that may be helpful to you, the pilot whose shift you took and the Program Manager. Once you have completed this, click the "Save" button and the schedule will be updated.

Sioux Valley Fixed-Wing Schedule

Weekly | Monthly | Log Out

Toggle Notes On/Off Print

Sunday December 4, 2005
No Scheduled Pilot

Monday December 5, 2005
No Scheduled Pilot

Tuesday December 6, 2005

4:00 AM	Chuck Yeager	Notes:
	Howard Hughes	Notes:
10:00 AM	Fred Flintstone	Notes:
	Joe Foss	Notes:
4:00 PM	Sherwin Bolks	Notes: Original pilot scheduled was Charlie Brown.
	Eddie Rickenbacker	Notes:

Wednesday December 7, 2005
No Scheduled Pilot

Thursday December 8, 2005
No Scheduled Pilot

Friday December 9, 2005
No Scheduled Pilot

Saturday December 10, 2005
No Scheduled Pilot

If you wish to view the Notes section of the schedule, select the "Toggle Notes On/Off" link in the upper right-hand corner of the page.

The same functionality is available with the monthly schedule. To access the monthly schedule, simply click on the "Monthly" link available in the Main Menu.

Sioux Valley Fixed-Wing Schedule

Weekly | Monthly | Log Out

Toggle Notes On/Off | Print

December 2005

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
27	28	29	30 4:00 AM: Joe Foss 4:00 AM: Charlie Brown 10:00 AM: Tim Rusten 10:00 AM: Fred Flintstone 4:00 PM: Jeff Leat 4:00 PM: Chad Breidenbach	1 4:00 AM: Sherwin Bolks Charlie Brown 10:00 AM: Fred Flintstone Joe Foss 4:00 PM: test ea;lkldj Sherwin Bolks	2	3 4:00 AM: Chad Breidenbach Charlie Brown 10:00 AM: Jeff Leat Fred Flintstone 4:00 PM: Chad Breidenbach Mike Christianson
4	5	6 4:00 AM: Chuck Yeager Howard Hughes 10:00 AM: Fred Flintstone Joe Foss 4:00 PM: Sherwin Bolks Eddie Rickenbacker	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

If a printed schedule is needed, you may select the "Print" link available with either the weekly or monthly schedule. Printing the monthly schedule may require you to select the "Options" tab from the pop-up window and choose the "Landscape" option.

Note:

Once you save a schedule change, email notification of the change will automatically be sent to the pilot whose schedule was affected, the new pilot on that shift and the Program Manager.

If you have any questions regarding the scheduling application, contact Sherwin Bolks at the Airport Office or via email (bolks@siouxvalley.org) or Chad Breidenbach, Sioux Valley Information Technology Department, at extension 89908 or email (breidenc@siouxvalley.org).

APPENDIX C:

Application Development

A Plan of Action was developed to summarize the approach needed to develop the application.

*Plan of Action
for the Sioux Valley Intensive Air Web-Based Scheduling Application*

Determine Data Requirements for the Web-Based Scheduling Application.

Determine data requirements by identifying attributes needed for requested functionality. Verify the integrity of the database schema by ensuring that the primary and foreign keys are appropriate.

Build the User Interface

The pages that are required for viewing by the pilots of the Sioux Valley Intensive Air Program will be determined with the layout of those pages completed in HTML. Design and graphic requirements will also be established at this time.

Integrate GUI with data

The data management function will be accomplished by using Visual Interdev 6.0 to write the ASP code. ASP code will be created to interact with the HTML pages to achieve the requested functionality and appropriate interaction between the web pages.

Test and ensure proper functionality and data integrity

Testing of the application will be conducted to ensure proper functionality before demonstrating to the representative of the Sioux Valley Intensive Air Program.

The following are the copies of the Program Code used in the development of the Web-Based Scheduling Application.

Calendar Code

(The JavaScript code in this page is used to generate a calendar for a specific month and display to the user. Once the user selects a date, the date will be passed back to the parent page.)

```

<!--
calendar code
-->
<html>
<head>
<title>Select Date, Please.</title>
<style>
    td {font-family: Tahoma, Verdana, sans-serif; font-size: 12px;}
</style>
<script language="JavaScript">

// months as they appear in the calendar's title
var ARR_MONTHS = ["January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"];
// week day titles as they appear on the calendar
var ARR_WEEKDAYS = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"];
// day week starts from (normally 0-Su or 1-Mo)
var NUM_WEEKSTART = 1;
// path to the directory where calendar images are stored. trailing slash req.
var STR_ICONPATH = 'img/';

var re_url = new RegExp('datetime=(\\-?\\d+)');
var dt_current = (re_url.exec(String(window.location))
    ? new Date(new Number(RegExp.$1)) : new Date());
var re_id = new RegExp('id=(\\d+)');
var num_id = (re_id.exec(String(window.location))
    ? new Number(RegExp.$1) : 0);
var obj_caller = (window.opener ? window.opener.calendars[num_id] : null);

if (obj_caller && obj_caller.year_scroll) {
    // get same date in the previous year
    var dt_prev_year = new Date(dt_current);
    dt_prev_year.setFullYear(dt_prev_year.getFullYear() - 1);
    if (dt_prev_year.getDate() != dt_current.getDate())
        dt_prev_year.setDate(0);

    // get same date in the next year
    var dt_next_year = new Date(dt_current);
    dt_next_year.setFullYear(dt_next_year.getFullYear() + 1);
    if (dt_next_year.getDate() != dt_current.getDate())
        dt_next_year.setDate(0);
}

// get same date in the previous month
var dt_prev_month = new Date(dt_current);
dt_prev_month.setMonth(dt_prev_month.getMonth() - 1);
if (dt_prev_month.getDate() != dt_current.getDate())
    dt_prev_month.setDate(0);

// get same date in the next month
var dt_next_month = new Date(dt_current);
dt_next_month.setMonth(dt_next_month.getMonth() + 1);
if (dt_next_month.getDate() != dt_current.getDate())
    dt_next_month.setDate(0);

// get first day to display in the grid for current month

```

```

var dt_firstday = new Date(dt_current);
dt_firstday.setDate(1);
dt_firstday.setDate(1 - (7 + dt_firstday.getDay() - NUM_WEEKSTART) % 7);

// function passing selected date to calling window
function set_datetime(n_datetime, b_close) {
    if (!obj_caller) return;

    var dt_datetime = obj_caller.prs_time(
        (document.cal ? document.cal.time.value : ''),
        new Date(n_datetime)
    );

    if (!dt_datetime) return;
    if (b_close) {
        window.close();
        obj_caller.target.value = (document.cal
            ? obj_caller.gen_tsmp(dt_datetime)
            : obj_caller.gen_date(dt_datetime)
        );
        window.opener.document.frmForm1.blnPass.value='1';
        window.opener.document.frmForm1.submit();
    }
    else obj_caller.popup(dt_datetime.valueOf());
}

</script>
</head>
<body bgcolor="#FFFFFF" marginheight="5" marginwidth="5" topmargin="5" leftmargin="5"
rightmargin="5">
<table class="clsOTable" cellspacing="0" border="0" width="100%">
<tr><td bgcolor="#4682B4">
<table cellspacing="1" cellpadding="3" border="0" width="100%">
<tr><td colspan="7"><table cellspacing="0" cellpadding="0" border="0" width="100%">
<tr>
<script language="JavaScript">
//var compCurrMonth = dt_current.getMonth();
//document.write ('current month = '+compCurrMonth);
document.write(
'<td>'+(obj_caller&&obj_caller.year_scroll?'<a
href="javascript:set_datetime('+dt_prev_year.valueOf()+')"></a>&nbsp;': '')+'<a
href="javascript:set_datetime('+dt_prev_month.valueOf()+')"></a></td>'+
'<td align="center" width="100%"><font
color="#ffffff">'+ARR_MONTHS[dt_current.getMonth()]+' '+dt_current.getFullYear() +
'</font></td>'+
'<td><a href="javascript:set_datetime('+dt_next_month.valueOf()+')"></a>'+(obj_caller && obj_caller.year_scroll?'&nbsp;:<a
href="javascript:set_datetime('+dt_next_year.valueOf()+')"></a>': '')+'</td>';
);
</script>
</tr>
</table></td></tr>
<tr>
<script language="JavaScript">

// print weekdays titles
for (var n=0; n<7; n++)
    document.write('<td bgcolor="#87cefa" align="center"><font
color="#ffffff">'+ARR_WEEKDAYS[(NUM_WEEKSTART+n)%7]+'</font></td>');
document.write('</tr>');

// print calendar table
var dt_current_day = new Date(dt_firstday);
while (dt_current_day.getMonth() == dt_current.getMonth() ||

```



```

dt_current_day.getMonth() == dt_firstday.getMonth()) {
// print row heder
document.write('<tr>');
for (var n_current_wday=0; n_current_wday<7; n_current_wday++) {
    if (dt_current_day.getDate() == new Date().getDate() &&
        dt_current_day.getMonth() == new Date().getMonth() &&
        dt_current_day.getFullYear() == new Date().getFullYear())
        // print current date
        document.write('<td bgcolor="#ffb6c1" align="center"
width="14%">');
    else if (dt_current_day.getDay() == 0 || dt_current_day.getDay() == 6)
        // weekend days
        document.write('<td bgcolor="#d9ead3" align="center"
width="14%">');
    else
        // print working days of current month
        document.write('<td bgcolor="#ffffff" align="center"
width="14%">');

        document.write('<a
href="javascript:set_datetime('+dt_current_day.valueOf()+', true);">');

        if (dt_current_day.getMonth() == this.dt_current.getMonth())
            // print days of current month
            document.write('<font color="#000000">');
        else
            // print days of other months
            document.write('<font color="#606060">');

        document.write(dt_current_day.getDate()+</font></a></td>');
        dt_current_day.setDate(dt_current_day.getDate()+1);
    }
    // print row footer
    document.write('</tr>');
}
if (obj_caller && obj_caller.time_comp)
    document.write('<form onsubmit="javascript:set_datetime('+dt_current.valueOf()+',
true)" name="cal"><tr><td colspan="7" bgcolor="#87CEFA"><font color="White" face="tahoma,
verdana" size="2">Time: <input type="text" name="time"
value="'+obj_caller.gen_time(this.dt_current)+'" size="8"
maxlength="8"></font></td></tr></form>');
</script>
</table></tr></td>
</table>
</body>
</html>

```

adminSchedule.asp

(The JavaScript code embedded in this section will open a pop-up window with copyDate.asp when selecting the "Copy from 13 weeks prior" link.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->
<%
    'If they don't have rights deny them
    If Session("intensiveair_loggedin") <> "2" Then
        Response.Redirect "denied.asp"
    End If

    'Initialize the arrShift array and the arrErrorPilot array

```

```

Dim arrShift()
Dim arrErrorPilot()

'Get the shift start times from the db
sql = "SELECT * FROM Configuration WHERE sItem LIKE 'Shift%' ORDER BY sItem"
Set oShift = Server.CreateObject("ADODB.RecordSet")
oShift.Open sql, oConn

iCnt = 0

'Loop thru the recordset
While NOT oShift.EOF
    sThisShift = Trim(oShift("sValue"))
    sThisShiftMinute = Minute(sThisShift)
    If Len(sThisShiftMinute) < 2 Then
        sThisShiftMinute = "0" & sThisShiftMinute
    End If
    'Bump the size of the array by 1 and enter the value in the array
    ReDim Preserve arrShift(iCnt)
    ReDim Preserve arrErrorPilot(iCnt)
    arrShift(iCnt) = Hour(sThisShift) & ":" & sThisShiftMinute
    arrErrorPilot(iCnt) = ""
    iCnt = iCnt + 1
    'go to the next record
    oShift.MoveNext
Wend

'Object cleanup
Set oShift = nothing

'Grab the variables
sDate = Trim(Request.Form("sDate"))

'If the form was submitted run this code
If Trim(Request.Form("blnPass")) = "true" Then
    'Error Checking
    If sDate = "" OR (sDate <> "" AND NOT IsDate(sDate)) Then
        blnError = true
        sErrDate = " error"
    End If
    For intX = 0 to ubound(arrShift)
        For intY = 1 to 2
            sThisPilot = Trim(Request.Form("shiftpilot" & intX & "-" &
intY))

            If sThisPilot = "" Then
                arrErrorPilot(intX) = " error"
                blnError = true
            End If
        Next
    Next

    'If there were no errors do the db stuff
    If NOT blnError Then
        'Loop thru all members of the array
        For intX = 0 to ubound(arrShift)
            For intY = 1 to 2
                sThisPilot = Trim(Request.Form("shiftpilot" & intX &
intY))

                'If this is a valid pilot id continue
                If sThisPilot <> "" AND sThisPilot <> "xxxx" Then
                    'INsert into the db
                    sql = "INSERT INTO Schedule (iUserID,
dShift) " & _
                        "VALUES (" & sThisPilot &
",'" & sDate & " " & arrShift(intX) & "')"
                    oConn.execute sql
                End If
            Next
        Next

        'Reload the page

```



```

                                If Trim(sID) = Trim(Request.Form("shiftpilot" &
intX)) AND Trim(Request.Form("blnPass")) <> "1" Then
                                    Response.Write " selected"
                                End If
                                Response.Write ">" & sName & "</option>" & VbCrLf
                                oPilots.MoveNext
                            Wend

                                Set oPilots = nothing
%>
                                </select></td>
<%
                                Next

                                Set oCheck = nothing
%>
</tr>
<%
                                Next
                                btnSave = "SAVE"
                                End If
%>
                                <tr>
                                    <td colspan=2></td>
                                    <td align="left" colspan=2><input type="submit" name="btn<%=btnSave%>"
id="btn<%=btnSave%>" value="<%=btnSave%>"></td>
                                </tr>
</table>

<script language="javascript">
if (document.frmForm1.sDate) document.frmForm1.sDate.focus();

var call = new calendar2(document.frmForm1.sDate);
call.year_scroll = true;
call.time_comp = false;
</script>
<!-- #include file="include/footer.asp" -->
```

denied.asp

```

<!-- #include file="include/header.asp" -->
<table border=0 cellpadding=3 cellspacing=0>
  <tr>
    <td class="error">You do not have access to this area.</td>
  </tr>
</table>
<!-- #include file="include/footer.asp" -->

```

modifySchedule.asp

(The JavaScript code embedded in this section will automatically close the popup if the copy date function is successful.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->
<%
  'Grab the schedule id
  iID = Trim(request("id"))
  sNewPilot = Trim(Request.Form("newpilot"))
  sNotes = Trim(Request.Form("notes"))

  'Set new pilot id to the logged in user if it is not already set
  If sNewPilot = "" AND Trim(Request.Form("blnPass")) = "" Then
    sNewPilot = Session("intensiveair_User_ID")
  End If

  'If the schedule id is sent then continue
  If iID <> "" Then
    'Select schedule info from db
    sql = "SELECT u.*, s.dShift, s.iScheduleID " & _
          "FROM Schedule s " & _
          "INNER JOIN Users u ON u.iUserID=s.iUserID " & _
          "WHERE s.iScheduleID=" & iID
    Set oRS = Server.CreateObject("ADODB.RecordSet")
    oRS.Open sql, oConn

    'Loop thru recordset
    If NOT oRS.EOF Then
      sOldPilot = Trim(oRS("sFName")) & " " & Trim(oRS("sLName"))
      sOldPilotID = Trim(oRS("iUserID"))
      sSchedID = Trim(oRS("iScheduleID"))

      'Get Modification Info
      sql = "SELECT * " & _
            "FROM ScheduleModifications sm " & _
            "INNER JOIN Users u ON sm.iNewPilot=u.iUserID " & _
            "WHERE iScheduleID=" & sSchedID & _
            " ORDER BY dChanged DESC"
      Set oMod = server.CreateObject("ADODB.RecordSet")
      oMod.Open sql, oConn

      If NOT oMod.eof Then
        sOldPilot = Trim(oMod("sFName")) & " " &
Trim(oMod("sLName"))

```

```

                                sOldPilotID = Trim(oMod("iUserID"))
End If

Set oMod = nothing

sDate = Trim(oRS("dShift"))
sSchedDate = Month(sDate) & "/" & Day(sDate) & "/" & Year(sDate)
sHour = Hour(sDate)
sMinute = Minute(sDate)
sAMPM = "AM"
If Len(sMinute) < 2 Then
    sMinute = "0" & sMinute
End If
If sHour > 11 Then
    If sHour > 12 Then
        sHour = sHour - 12
    End If
    sAMPM = "PM"
End If
sSchedTime = sHour & ":" & sMinute & " " & sAMPM
End If

Set oRS = nothing
End If

'If form was submitted run through this code
If Trim(Request.Form("blnPass")) <> "" Then
    'Error Checking
    If sNewPilot = "" Then
        blnError = true
        sErrNewPilot = " error"
    End If
    If Len(sNotes) > 1000 Then
        sNotes = left(sNotes,1000)
    End If

    'If no errors enter into database
    If NOT blnError Then
        sql = "INSERT INTO ScheduleModifications (iScheduleID, iUserID,
iNewPilot, sNotes) " & _
                "VALUES (" & IID & ", " &
Session("intensiveair_User_ID") & ", " & _
                sNewPilot & ", " & sNotes & " )"
        oConn.execute sql

        'Send an email
        sBody = "<b>Schedule Change:</b><br>" &
getPilotNameFromID(sNewPilot) & _
                " is replacing " & getPilotNameFromID(sOldPilotID) &
-
                " on " & sSchedDate & " " & sSchedTime & ".<p>" &
sNotes & _
                "<p><a
href=""https://apps.siouxvalley.org/intensiveair/"">Click " & _
                "Here</a> to go to the schedule."
        sSubject = "Schedule change as of " & Date

        sOldPilotEmail = getEmailFromID(sOldPilotID)
        If Trim(sOldPilotEmail) <> "" AND NOT IsNull(sOldPilotEmail) Then
            arrOldPilotEmail = Split(Trim(sOldPilotEmail))
            For intX = 0 to ubound(arrOldPilotEmail)
                If IsEmail(arrOldPilotEmail(intX)) AND
arrOldPilotEmail(intX) <> "" Then
                    sendMail arrOldPilotEmail(intX),
"webmaster@siouxvalley.org", sSubject, sBody
                End If
            Next
        End If

        sNewPilotEmail = getEmailFromID(sNewPilot)
        If Trim(sNewPilotEmail) <> "" AND NOT IsNull(sNewPilotEmail) Then

```

```

arrNewPilotEmail = Split(Trim(sNewPilotEmail))
For intX = 0 to ubound(arrNewPilotEmail)
    If IsEmail(arrNewPilotEmail(intX)) AND
arrNewPilotEmail(intX) <> "" Then
        sendMail arrNewPilotEmail(intX),
"webmaster@siouxvalley.org", sSubject, sBody
        End If
    Next
End If

sendMail "chadbreidenbach@yahoo.com", "webmaster@siouxvalley.org",
sSubject, sBody

blnDone = true

End If
End If
%>

<html>
<head>
<script language="javascript">
function endIt()
{
    sHref = window.opener.document.location.href;
    window.opener.document.location.href=sHref;
    this.window.close();
}
</script>
<title>Intensive Air Fixed-Wing Pilot Schedule Modifications</title>
<link rel="stylesheet" type="text/css" href="/intensiveair/include/style.css">
</head>

<body topmargin=0 leftmargin=0<%If blnDone Then response.write " onLoad=""endIt()"" End
If%>>
<form method="post" name="frmForm1" action="<%=Request.ServerVariables("SCRIPT_NAME")%>">
<input type="hidden" name="blnPass" value="true">
<input type="hidden" name="id" value="<%=iID%>">
<table border=0 cellpadding=0 cellspacing=0>
    <tr>
        <td class="subhead">Schedule Date:</td>
        <td class="data"><%=sSchedDate%></td>
    </tr>
    <tr>
        <td class="subhead">Schedule Time:</td>
        <td class="data"><%=sSchedTime%></td>
    </tr>
    <tr>
        <td class="subhead">Scheduled Pilot:</td>
        <td class="data"><%=sOldPilot%></td>
    </tr>
    <tr>
        <td class="subhead<%=sErrNewPilot%>">New Pilot:</td>
        <td class="data"><select name="newpilot">
<%
'Display all other users to choose from
sql = "SELECT * FROM Users WHERE iUserID <> " & sOldPilotID & " ORDER BY sLName,
sFName"
Set oPilot = server.CreateObject("ADODB.RecordSet")
oPilot.Open sql, oConn

If oPilot.EOF Then
    Response.Write "<option value="">No Pilots Found</option>" & VbCrLf
Else
    Response.Write "<option value=""></option>" & VbCrLf
End If

'Loop thru recordset
While NOT oPilot.EOF
    sPilotID = trim(oPilot("iUserID"))

```



```

        sPilotName = Trim(oPilot("sLName")) & ", " & Trim(oPilot("sFName"))
        Response.Write "<option value="" & sPilotID & """"
        If Trim(sPilotID) = Trim(sNewPilot) Then
            Response.Write " selected"
        End If
        Response.Write ">" & sPilotName & "</option>" & VbCrLf
        oPilot.MoveNext
    Wend

    Set oPilot = nothing
%>
        </select></td>
    </tr>
    <tr>
        <td class="subhead">Notes</td>
        <td class="data"><textarea name="notes" cols=30
rows=4><%=sNotes%></textarea></td>
    </tr>
    <tr>
        <td></td>
        <td><input type="submit" name="btnSubmit" value="SAVE"></td>
    </tr>
</table>
</form>
</body>
</html>

```

copyDate.asp

(The JavaScript code embedded in this section will automatically close the popup window if the copy date function is successful.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->
<%
    'Grab the schedule id
    sDate = Trim(request("dt"))
    'Compute the needed dates
    sCopyDate = DateAdd("d",-91,sDate)
    sCheckDate = Month(sCopyDate) & "/" & Day(sCopyDate) & "/" & Year(sCopyDate)

    'If a real date has been submitted to page keep going
    If sDate <> "" AND IsDate(sDate) Then
        'Grab info from previous schedule
        sql = "SELECT * FROM Schedule " & _
            "WHERE dShift >= '" & sCheckDate & " 00:00:01' " & _
            "AND dShift <= '" & sCheckDate & " 23:59:59'"
        Set oCopy = server.CreateObject("ADODB.RecordSet")
        oCopy.Open sql, oConn

        'If no info set a flag
        If oCopy.EOF Then
            blnEnd = true
        End If

        'Loop thru the recordset
        While NOT oCopy.EOF
            'Grab the variables
            iUserID = Trim(oCopy("iUserID"))
            dShift = Trim(oCopy("dShift"))
            dShift = DateAdd("d",91,dShift)
            'Insert these pilots into db with new schedule date

```

```

        sql = "INSERT INTO Schedule (iUserID, dShift) VALUES (" & iUserID &
-
        ", ' " & dShift & "' ) "
        oConn.execute sql
        oCopy.Movenext
        blnDone = true
    Wend

    Set oCopy = nothing
End If
%>

<html>
<head>
<script language="javascript">
function endIt()
{
    sHref = window.opener.document.location.href;
    window.opener.document.location.href=sHref;
    this.window.close();
}
</script>
<link rel="stylesheet" type="text/css" href="/intensiveair/include/style.css">
</head>

<body topmargin=0 leftmargin=0<%If blnDone Then response.write " onLoad=""endIt()"" End
If%>>
<form method="post" name="frmForm1" action="<%=Request.ServerVariables("SCRIPT_NAME")%>">
<input type="hidden" name="blnPass" value="true">
<input type="hidden" name="dt" value="<%=sDate%>">
<table border=0 cellpadding=0 cellspacing=0>
<tr>
<td class="subhead">No Pilots scheduled on <%=sCheckDate%></td>
</tr>
<tr>
<td class="subhead"><a href="javascript:window.close()">Close
Window</a></td>
</tr>
</table>
</form>
</body>
</html>

```

login.asp

(The JavaScript code embedded in this section sets the focus to the appropriate login field.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/functions.asp" -->

<%
    'Grab the variables
    sUser = Trim(Request.Form("sUser"))
    sPass = Trim(Request.Form("sPass"))
    sURL = Trim(request("surl"))
    blnPass = Trim(Request.Form("btnSubmit"))

    'Set the page to flow to
    If sURL = "" Then
        sURL = "default.asp"
    End If

    'If the form was submitted continue
    If blnPass <> "" Then

```

```

'Error Checking
If sUser = "" OR sPass = "" Then
    blnError = true
End If

If NOT blnError Then
    'Check login data
    If Login(sUser, sPass) Then
        'Go to the requested page if info is correct
        Response.Redirect sURL
    Else
        blnError = true
    End If
End If
If blnError Then
    'Display this error at the top of the page if info is incorrect
    sError = "<tr><td class=""error"">The username and/or password did
not " & _
                                "match any in our system. Please try
again.</td></tr>"
    End If
End If
%>

<!-- #include file="include/header.asp" -->

<input type="hidden" name="surl" value="<%=sURL%>">
<div class="login">
    <table border=0 cellpadding=3 cellspacing=0 align="center"><%=sError%>
        <tr>
            <td valign="top" align="right" class="login">Username: </td>
            <td valign="top" align="left"><input type="text" name="sUser"
value="<%=sUser%>"
                                size="20" maxlength="20"></td>
        </tr>
        <tr>
            <td valign="top" align="right" class="login">Password: </td>
            <td valign="top" align="left"><input type="password" name="sPass"
                                value="" size="20" maxlength="20"></td>
        </tr>
        <tr>
            <td valign="top" align="center" colspan=2><input type="submit"
name="btnSubmit"
                                value="SUBMIT"></td>
        </tr>
    </table>
</div>
</form>

<script language="javascript">
if (document.frmForm1.sUser.value=='')
{
    document.frmForm1.sUser.focus();
} else {
    document.frmForm1.sPass.focus();
}
</script>

<!-- #include file="include/footer.asp" -->

```

monthly.asp

(The embedded JavaScript in this section will hide or show the "Notes" section depending on the user's prior selection.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->

<%
    'Initialize arrays for date display
    ArrMonth = Array("", "January", "February", "March", "April", "May", "June", "July", "
        "August", "September", "October", "November", "December")

    arrShift = Array("", "shift1", "shift2", "shift3")

    Dim arrDays(43)
    Dim arrCols(43)

    'Grab variables
    sDate = trim(Request.Form("sDate"))

    'Set default date if one is not specified
    If sDate = "" Then
        sDate = Month(Date) & "/" & Day(Date) & Year(Date)
    End If

    'Calculate beginning and ending dates
    sBeginDay = DatePart("w", sDate)
    sLoadDate = DateAdd("d", 1 - sBeginDay, sDate)

    'Loop thru every square of the calendar
    For intX = 1 to 42
        'Decide if the day displayed is the current month or another month
        If (intX < 8 AND Day(sLoadDate) > intX) OR (intX > 20 AND Day(sLoadDate) <
8) Then
            sTempClass = "offMonth"
        Else
            sTempClass = "onMonth"
        End If

        'To add stuff to the day area on the map enter it here
        sTemp = "<td valign=""top"" class="" & sTempClass & "">" &
Day(sLoadDate) & "<br>"
        sql = "SELECT u.*, s.dShift, s.iScheduleID " & _
            "FROM Schedule s " & _
            "INNER JOIN Users u ON u.iUserID=s.iUserID " & _
            "WHERE dShift >= ' " & sLoadDate & " 00:00:01' " & _
            "AND dShift <= ' " & sLoadDate & " 23:59:59' " & _
            "ORDER BY dShift ASC"
        Set oRS = Server.CreateObject("ADODB.RecordSet")
        oRS.open sql, oConn

        sCompareOutTime = ""

        'Loop thru the recordset
        While NOT oRS.EOF
            iScheduleID = Trim(oRS("iScheduleID"))
            sThisDate = Trim(oRS("dShift"))
            sHour = Hour(sThisDate)
            sMinute = Minute(sThisDate)
            If Len(sMinute) < 2 Then
                sMinute = "0" & sMinute
            End If
            sAMPM = "AM"
        End While
    End For
%>

```

```

If sHour > 11 Then
    If sHour > 12 Then
        sHour = sHour - 12
    End If
    sAMPM = "PM"
End If
sThisTime = sHour & ":" & sMinute & " " & sAMPM

'Find if any Modifications
sql = "SELECT * " & _
        "FROM ScheduleModifications sm " & _
        "INNER JOIN Users u ON sm.iNewPilot=u.iUserID " & _
        "WHERE iScheduleID=" & iScheduleID & _
        " ORDER BY dChanged DESC"
Set oMod = server.CreateObject("ADODB.RecordSet")
oMod.Open sql, oConn

If oMod.eof Then
    sName = Trim(oRS("sFName")) & " " & Trim(oRS("sLName"))
    sUser = Trim(oRS("iUserID"))
    sNotes = ""
Else
    sName = Trim(oMod("sFName")) & " " & Trim(oMod("sLName"))
    sUser = Trim(oMod("iNewPilot"))
    sNotes = Trim(oMod("sNotes"))
End If

Set oMod = nothing

'Find the correct shift number
sCompTime = Hour(sThisDate) & ":" & Minute(sThisDate)
If DateDiff("n",sCompTime,Session("intensivair_shift3")) <= 0 Then
    iShift = 3
ElseIf DateDiff("n",sCompTime,Session("intensivair_shift2")) <= 0
Then
    iShift = 2
Else
    iShift = 1
End If

If Trim(sThisTime) <> Trim(sCompareOutTime) Then
    sOutTime = sThisTime & ":"
    sCompareOutTime = sThisTime
Else
    sOutTime = "<font color=""white"">" & sThisTime &
":</font>"
End If

'Create clickable text to display on schedule
sTemp = sTemp & "<table border=0 cellpadding=0
cellspacing=0><tr><td nowrap><a " & _
        "style=""cursor:hand"" onClick=""modMe(' " &
iScheduleID & _
        "')" title=""modify this schedule""><span class=""
& _
        arrShift(iShift) & "">" & sOutTime &
"&nbsp;</span></a></td><td nowrap><a " & _
        "style=""cursor:hand"" onClick=""modMe(' " &
iScheduleID & _
        "')" title=""modify this schedule""><span class=""
& _
        arrShift(iShift) & "">" & sName & _
"</span></a><div class=""data hide""
id=""tdNotes""><b>Notes:</b>" & _
        sNotes & "</div></td></tr></table>"
oRS.movenext
If NOT oRS.EOF Then
    sTemp = sTemp & "<br>"
End If
Wend

```

```

        sTemp = sTemp & "</td>"
        arrDays(intX) = Day(sLoadDate)
        arrCols(intX) = sTemp
        sLoadDate = DateAdd("d",1,sLoadDate)
    Next

%>

<!-- #include virtual="/intensiveair/include/header.asp" -->

<script language="javascript">
function modMe(iID)
{
    newWin =
window.open('modifyschedule.asp?id='+iID,'modWin','left=10,top=10,width=500,height=200,resizeable,scrollbars');
}
function toggleNotes()
{
    var oNotes = document.all['tdNotes'];
    for (var i=0;i<document.all['tdNotes'].length;i++)
    {
        if (oNotes[i].className == 'show')
        {
            oNotes[i].className = 'hide';
        } else {
            oNotes[i].className='show';
        }
    }
}
</script>

<table border=0 cellpadding=3 cellspacing=0 class="printLine">
    <tr>
        <td width="100%" align="right"><a style="cursor:hand;color:blue;text-decoration:underline"
            onClick="toggleNotes()" title="Toggle Notes On/Off">Toggle Notes
On/Off</a>
            &nbsp;&nbsp;&nbsp;<a style="cursor:hand;color:blue;text-decoration:underline"
            onClick="window.print()" title="Print This Page">Print</a><p></td>
    </tr>
</table>

<select name="sDate" onChange="document.frmForm1.submit()">
    <option value=""></option>
<%
    sThisDate = Month(Date) & "/1/" & Year(Date)

    'Create the drop down listing the next years dates
    For intX = -1 to 12
        sTempDate = DateAdd("m",intX,sThisDate)
        Response.Write "<option value="" & Month(sTempDate) & "/1/" &
Year(sTempDate) & """"
        If DateValue(sDate) = DateValue(sTempDate) Then
            Response.Write " selected"
        End If
        Response.Write ">" & arrMonth(Month(sTempDate)) & " " & Year(sTempDate) &
"</option>"
    Next
%>
</select>
<table border=1 cellpadding=3 cellspacing=0>
    <tr>
        <td>Sunday</td>
        <td>Monday</td>
        <td>Tuesday</td>
        <td>Wednesday</td>
        <td>Thursday</td>
        <td>Friday</td>
    </tr>
</table>

```

```

                <td>Saturday</td>
            </tr>
<%
    'Write it all out
    iCnt = 0
    iRowCnt = 0
    For intX = 1 to 42
        iCnt = iCnt + 1
        If iCnt = 1 Then
            Response.Write "<tr>"
            iRowCnt = iRowCnt + 1
        End If
        Response.Write arrCols(intX)
        If iCnt = 7 Then
            Response.Write "</tr>"
            iCnt = 0
            If arrDays(intX) < 7 AND iRowCnt > 1 Then
                intX = 100
            End If
        End If
    Next
%>
</table>
<!-- #include virtual="/intensiveair/include/footer.asp" -->

```

admin.asp

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->
<%
    'If they don't have rights deny them
    If Session("intensiveair_loggedin") <> "2" Then
        Response.Redirect "denied.asp"
    End If

    'Grab the variables
    sUserName = Trim(Request.Form("username"))
    sPass1 = Trim(Request.Form("pass1"))
    sPass2 = Trim(Request.Form("pass2"))
    sFName = Trim(Request.Form("fname"))
    sLName = Trim(Request.Form("lname"))
    sSecLevel = Trim(Request.Form("secLevel"))
    sPilotID = Trim(Request.Form("sPilotID"))
    sEmail = Trim(Request.Form("sEmail"))

    'If the form was submitted run this code
    If Trim(Request.Form("blnPass")) <> "" Then
        'Error Checking
        If sUserName = "" Then
            blnError = true
            sErrUserName = " error"
        End If
        If sPass1 = "" OR sPass2 = "" OR sPass1 <> sPass2 Then
            sErrPass = " error"
            blnError = true
        End If
        If sFName = "" Then
            sErrFName = " error"
            blnError = true
        End If
        If sLName = "" Then
            sErrLName = " error"
        End If
    End If

```

```

        blnError = true
    End If
    If sSecLevel = "" Then
        sErrSecLevel = " error"
        blnError = true
    End If
    '
    ' If sEmail = "" OR (sEmail <> "" AND NOT isEmail(sEmail)) Then
    '     sErrEmail = " error"
    '     blnError = true
    '
    End If

    'If no errors then continue
    If NOT blnError Then
        'If this is a new pilot insert into the db tables
        If sPilotID = "" Then
            sql = "INSERT INTO Users
(sUserName,sPass,sFName,sLName,iSecurityLevel,sEmail) " & _
                "VALUES ('" & formatSQL(sUserName) & "','" &
formatSQL(sPass1) & _
                "','" & formatSQL(sFName) & "','" &
formatSQL(sLName) & "','" & _
                sSecLevel & "','" & formatSQL(sEmail) & "'"")"
            oConn.execute sql

            'Reload the page
            Response.Redirect "admin.asp?added=1"
            'If this is an existing pilot update the db with the new info
            Else
                sql = "UPDATE Users SET sUserName='" & formatSQL(sUserName)
& "','" & sPass='" & _
                formatSQL(sPass1) & "','" & sFName='" &
formatSQL(sFName) & _
                "','" & sLName='" & formatSQL(sLName) & "','" &
iSecurityLevel='" & _
                sSecLevel & "','" & sEmail='" & formatSQL(sEmail)
& _
                "'" WHERE iUserID=" & sPilotID
            oConn.execute sql

            'Reload the page
            Response.Redirect "admin.asp?added=2"
        End If
    End If

    'If the form was submitted when selecting a pilot grab that pilot info
    ElseIf sPilotID <> "" Then
        sql = "SELECT * FROM Users WHERE iUserID=" & sPilotID
        Set oPilot = server.CreateObject("ADODB.RecordSet")
        oPilot.Open sql, oConn

        'If a record came back grab the data
        If NOT oPilot.EOF Then
            sUserName = Trim(oPilot("sUserName"))
            sPass3 = Trim(oPilot("sPass"))
            sPass4 = Trim(oPilot("sPass"))
            sFName = Trim(oPilot("sFName"))
            sLName = Trim(oPilot("sLName"))
            sSecLevel = Trim(oPilot("iSecurityLevel"))
            sEmail = Trim(oPilot("sEmail"))
        End If

        Set oPilot = nothing
    End If

%>
<!-- #include file="include/header.asp" -->

<input type="hidden" name="blnPass" value="true">
<table border=0 cellpadding=3 cellspacing=0>
<%
    If Trim(Request.QueryString("added")) = "1" Then

```



```

%>
    <tr>
        <td class="head" colspan="3"><span class="error">New User
Added</span></td>
    </tr>
<%
    ElseIf Trim(Request.QueryString("added")) = "2" Then
%>
    <tr>
        <td class="head" colspan="3"><span class="error">User Updated</span></td>
    </tr>
<%
    End If
%>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrUserName%>">Choose User to Edit:</td>
        <td class="data"><select name="sPilotID"
onChange="frmForm1.blnPass.value='';frmForm1.submit()">
            <option value="">New User</option>
<%
    'Grab pilot info from the database
    sql = "SELECT * FROM Users ORDER BY sLName, sFName"
    Set oPilots = Server.CreateObject("ADODB.RecordSet")
    oPilots.Open sql, oConn

    'Loop thru the recordset
    While NOT oPilots.EOF
        sName = Trim(oPilots("sLName")) & ", " & Trim(oPilots("sFName"))
        sID = Trim(oPilots("iUserID"))
        Response.Write "<option value="" & sID & """"
        If sID = sPilotID Then
            Response.Write " selected"
        End If
        Response.Write ">" & sName & "</option>" & VbCrLf
        oPilots.MoveNext
    Wend

    Set oPilots = nothing
%>
        </select></td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrUserName%>">Username:</td>
        <td class="data"><input type="text" name="username" value="<%=sUserName%>"
            size="20" maxlength="50"></td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrPass%>">Password:</td>
        <td class="data"><input type="password" name="pass1" value="<%=sPass3%>"
            size="20" maxlength="50"></td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrPass%>">Password Again:</td>
        <td class="data"><input type="password" name="pass2" value="<%=sPass4%>"
            size="20" maxlength="50"></td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrFName%>">First Name:</td>
        <td class="data"><input type="text" name="fname" value="<%=sFName%>"
            size="20" maxlength="50"></td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;&nbsp;</td>
        <td class="subHead<%=sErrLName%>">Last Name:</td>
        <td class="data"><input type="text" name="lname" value="<%=sLName%>"
            size="20" maxlength="50"></td>

```


default.asp

(The embedded JavaScript in this section will hide or show the "Notes" section depending on the user's prior selection.)

```

<!-- #include file="include/dbConnection.asp" -->
<!-- #include file="include/security.asp" -->
<!-- #include file="include/functions.asp" -->

<%
    'Create arrays to show the date info
    ArrMonth = Array("", "January", "February", "March", "April", "May", "June", "July", " _
                "August", "September", "October", "November", "December")

    arrDay =
Array("", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

    arrShift = Array("", "shift1", "shift2", "shift3")

    'Grab the shift info from the configuration table
    sql = "SELECT * FROM Configuration WHERE sItem LIKE 'shift%' ORDER BY sItem"

    Set oConfig = Server.CreateObject("ADODB.RecordSet")
    oConfig.Open sql, oConn

    'Set a session variable for each shift
    While NOT oConfig.EOF
        Session("intensivair_" & Trim(oConfig("sItem"))) = Trim(oConfig("sValue"))
        oConfig.MoveNext
    Wend

    Set oConfig = nothing

    'Create a beginning datetime
    sBeginDate = Month(Date) & "/" & Day(Date) & "/" & Year(Date) & " 00:00:01"
%>

<!-- #include virtual="/intensiveair/include/header.asp" -->

<script language="javascript">
function modMe(iID)
{
    newWin =
window.open('modifyschedule.asp?id='+iID, 'modWin', 'left=10,top=10,width=500,height=200,re
sizable,scrollbars');
}
function toggleNotes()
{
    var oNotes = document.all['tdNotes'];
    for (var i=0;i<document.all['tdNotes'].length;i++)
    {
        if (oNotes[i].className == 'show')
        {
            oNotes[i].className = 'hide';
        } else {
            oNotes[i].className='show';
        }
    }
}
}
</script>

<table border=0 cellpadding=3 cellspacing=0 class="printLine">
<tr>

```


functions.asp

```

<%
'*****
'** Function will return true if login successful or false if not successful **
'*****
Function login(byVal sUserName, byVal sPass)
    'Declare the variables
    Dim sql, oRS

    'Grab info from the db
    sql = "SELECT * FROM Users WHERE sUserName='" & sUserName & "' AND sPass='" &
sPass & "'"
    Set oRS = Server.CreateObject("ADODB.RecordSet")
    oRS.Open sql, oConn

    'Loop thru the recordset
    If NOT oRS.EOF Then
        'Set the session variables used for security
        Session("intensiveair_loggedin") = Trim(oRS("iSecurityLevel"))
        Session("intensiveair_username") = sUserName
        Session("intensiveair_User_ID") = Trim(oRS("iUserID"))
        updateLogin Trim(oRS("iUserID"))
        login = true
    Else
        login = false
    End If

    Set oRS = nothing
End Function

'*****
'** Function updates the dLastOn field in the Users table in the db **
'*****
Function updateLogin(byVal sUserID)
    Dim sql

    sql = "UPDATE Users SET dLastOn='" & now() & "' WHERE iUserID=" & sUserID
    oConn.execute sql
End Function

'*****
'** Function prepares a string for inclusion in SQL statement **
'*****
Function formatSQL(byVal sStr)
    formatSQL = replace(sStr, "'", "'")
End Function

'*****
'** Function sends an email to the requested address **
'*****
Function sendMail(byVal sTo, byVal sFrom, byVal sSubj, byVal sBody)
    Dim Mail

    Set Mail = CreateObject("Persits.MailSender")

    ' enter valid SMTP host
    Mail.Host = "172.20.1.72"

    Mail.From = sFrom
    Mail.AddAddress sTo
    Mail.Subject = sSubj
    Mail.Body = sBody
    Mail.IsHTML = True

    Mail.Send

```

```

        Set Mail = Nothing
End Function

'*****
'** Function returns the users name given the id **
'*****
Function getPilotNameFromID(byVal id)
    Dim sql, oPilot

    sql = "SELECT * FROM Users WHERE iUserID=" & id
    Set oPilot = server.CreateObject("ADODB.RecordSet")
    oPilot.Open sql, oConn

    If NOT oPilot.EOF Then
        getPilotNameFromID = Trim(oPilot("sFName")) & " " & Trim(oPilot("sLName"))
    End If

    Set oPilot = nothing
End Function

'*****
'** Function returns the users email address given the id **
'*****
Function getEmailFromID(byVal id)
    Dim sql, oEmail

    sql = "SELECT * FROM Users WHERE iUserID=" & id
    Set oEmail = server.CreateObject("ADODB.RecordSet")
    oEmail.Open sql, oConn

    If NOT oEmail.EOF Then
        getEmailFromID = Trim(oEmail("sEmail"))
    End If

    Set oEmail = nothing
End Function

'*****
'** Function checks if the given email fits the format of a valid email add. **
'*****
Function isEmail(byVal sEmail)
    If Trim(sEmail) <> "" AND NOT IsNull(sEmail) Then
        isEmail = false
        Dim regEx, retVal
        Set regEx = New RegExp

        ' Create regular expression:
        regEx.Pattern = "^[\w-\.] {1,} \@[([\da-zA-Z-] {1,} \. ) {1,} [(\da-zA-Z-] {2,3} )$"

        ' Set pattern:
        regEx.IgnoreCase = true

        ' Set case sensitivity.
        retVal = regEx.Test(sEmail)

        ' Execute the search test.
        If not retVal Then
            exit function
        End If
    End If

    isEmail = true
End Function

%>

```

security.asp

```

<%
    If Session("intensiveair_loggedin") = "" Then
        Response.Redirect "/intensiveair/login.asp?surl=" &
Trim(Request.ServerVariables("SCRIPT_NAME"))
    End If
%>

```

dbConnection.asp

```

<%
    'Create the database connection
    Set oConn = Server.CreateObject("ADODB.Connection")
    oConn.Open "Driver={SQL Server};" & _
        "Server=webtest;" & _
        "Database=FixedWingSchedule;" & _
        "Uid=fixed_user;" & _
        "Pwd=fixe_1117"
%>

```

header.asp

```

<html>
<head>
<title>Intensive Air Fixed-Wing Pilot Schedule</title>
<%
    IF InStr(lcase(Request.ServerVariables("SCRIPT_NAME")), "default.asp") Then
%>
<meta http-equiv="refresh" content="600">
<%
    End If
%>
<link rel="stylesheet" type="text/css" media="screen"
href="/intensiveair/include/style.css">
<link rel="stylesheet" type="text/css" media="print"
href="/intensiveair/include/print_style.css">
</head>

<body topmargin=0 leftmargin=0
background="/intensiveair/images/cloudy_blue.jpg"<%=sBodyCode%>>
<table border=0 cellpadding=0 cellspacing=0 class="main">
    <tr>
        <td><a href="/intensiveair/default.asp" border="0"></a><br></td>
    </tr>

<%
    'If user is logged in let them see the nav bar
    If Session("intensiveair_loggedin") <> "" AND NOT
IsNull(Session("intensiveair_loggedin")) Then
%>
    <tr>
        <td class="nav">&nbsp;<a href="/intensiveair/default.asp"
            title="Weekly View">Weekly</a>&nbsp;<a href="/intensiveair/monthly.asp"
            title="Monthly View">Monthly</a><%
            'If the user has admin privileges give them these links also
            If Session("intensiveair_loggedin") = "2" Then
                %>&nbsp;<a href="/intensiveair/admin.asp" title="Add New User">Add/Edit
                User</a>&nbsp;<a href="/intensiveair/adminSchedule.asp" title="Schedule
                Admin">Schedule Admin</a>&nbsp;<%

```



```
                End If
                %><a href="logout.asp" title="LOGOUT">Log Out</a>&nbsp;</td>
            </tr>
        <%
            End If
        %>
    </table>
    <form method="post" name="frmForm1" action="<%=Request.ServerVariables("SCRIPT_NAME")%>">
```

footer.asp

(This code is used to store "SCRIPT_NAME" on the server side and pass it to the JavaScript code.)

```
</form>
<script language="javascript">
    var sThisPageJS = '<%=Trim(Request.ServerVariables("SCRIPT_NAME"))%>';
</script>
</body>
</html>
```
