

SOFTWARE ARCHITECTURE FOR WEB-ACCESSIBLE HEAT EXCHANGER EXPERIMENT

by

Rubaiyat Amin Khan

B.Sc. in Civil Engineering
Bangladesh University of Engineering and Technology, 2000

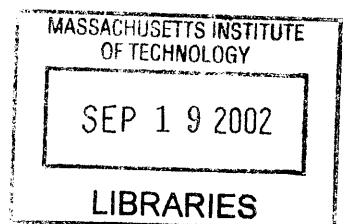
SUBMITTED TO THE
DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2002



© 2002 Massachusetts Institute of Technology. All rights reserved.

BARKER

Signature of Author:

Department of Civil and Environmental Engineering
August 16, 2002

Certified by:

Clark K. Colton

Professor of Chemical Engineering
Thesis Supervisor

Certified by:

Kevin S. Amaralunga

Associate Professor of Civil and Environmental Engineering
Thesis Reader

Accepted by:

Oral Buyukozturk

Chairman, Departmental Committee on Graduate Studies

SOFTWARE ARCHITECTURE FOR WEB-ACCESSIBLE HEAT EXCHANGER EXPERIMENT

by
Rubaiyat Amin Khan

Submitted to the Department of Civil and Environmental Engineering
On August 16, 2002 in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Civil and Environmental Engineering

ABSTRACT

Web-accessible laboratory experiments are gaining popularity due to their advantages over traditional laboratory experiments. With advancement in internet technology, more and more laboratories are becoming web-accessible. The needs for these virtual labs fit perfectly into the modern methods of assembly, delivery and access to educational technology resources adopted by educational institutions around the world.

But still there is an absence of proper standards as to how to properly design and deal with the software infrastructure issues that make it possible for the labs to be accessible through web. It is always necessary to design a software system that is robust, platform independent and easily modifiable to accommodate changing requirements. It is also necessary for the system to be easily replicable for newer labs. The software architecture for the MIT I-Lab Heat Exchanger experiment is designed keeping in mind all these needs and it has been modified at different times to make room for changing requirements both in part of students performing the experiment (through student assessment of the experiment) and the instructors of the courses that the system has been deployed to.

The design and implementation of the software architecture for the heat exchanger experiment is discussed in this thesis. The key component of the system is a Laboratory Heat Exchanger, which is employed to study principles of heat transfer. The software system is divided into four functional components: local server control of the laboratory equipment, remote client control of the equipment, client collaboration and user registration, authentication and experiment scheduling. The system has been successfully used in three MIT courses and one course at University of Texas at Austin over the span of two semesters and is scheduled to be used in two more courses in addition to these.

With evolution of newer and better technology, the system will be able to accommodate itself to suitable changes that conform to the requirements of the system and always thrive to provide a more robust solution to the problems at hand.

Thesis Supervisor: Clark K. Colton
Title: Professor of Chemical Engineering

ACKNOWLEDGEMENTS

I would like to avail this opportunity to express my sincere gratitude to Professor Clark K. Colton for granting me this rare opportunity to work with him in this research project. It is his untiring effort, invaluable suggestions and constant guidance throughout the research period that enabled me to present this thesis.

I would like to thank Professor Kevin S. Amaratunga for his assistance and valuable suggestions throughout the period of research and the preparation of this thesis. I am particularly grateful to him for all the time he has spared me out of his busy schedule.

I am solely indebted to my parents Firoza Rahman and Abdur Rahman Khan and my entire family for their patience, support, encouragement and efforts that helped me to be who I am today.

I am thankful to my wife Samia Ilham for her constant support, patience and inspiration that got me through the long and hard days at MIT.

Lastly, I would like to pay my utmost respect to God Almighty, for granting me this wonderful opportunity to work and to live.

Table of Contents

Chapter 1. INTRODUCTION.....	8
1.1 Why Web-accessible Experiment?.....	8
1.2 I-Lab Heat Exchanger Project - Objectives.....	10
1.3 Problem Statement and Challenges.....	11
1.4 Outline of the Thesis	13
Chapter 2. EQUIPMENT DESCRIPTION	15
2.1 Introduction	15
2.2 Heat Exchanger and Types.....	15
2.3 Armfield Heat Exchanger Equipment.....	19
2.4 Channel Allocations	24
2.5 Heat Exchanger Driver Software and Library Functions.....	28
Chapter 3. RESEARCH PATH AND THE SOFTWARE ARCHITECTURE.....	31
3.1 Introduction	31
3.2 Research path	31
3.3 Software Solutions	33
3.4 Chosen Solution & Reasons.....	34
3.5 Software Architecture	35
Chapter 4. EQUIPMENT CONTROL AND MONITORING	38
4.1 Introduction	38
4.2 Server Control and Monitoring using LabView.....	38
4.3 Publishing Real-time data and getting Real-time input with DataSocket Server ...	50
4.4 Remote Control and Monitoring using Java Applet.....	55
Chapter 5. USER MANAGEMENT AND COLLABORATION.....	60
5.1 Introduction	60
5.2 Data Model and the Database.....	60
5.3 User Management System.....	63
5.4 Java Collaboration System.....	75
Chapter 6. STUDENT ASSESSMENT	81
6.1 Introduction	81
6.2 Objectives of the Assessment.....	81
6.3 Student responses	82
6.4 Comparative evaluation.....	94
Chapter 7. CONCLUSION AND FUTURE IMPROVEMENTS.....	98
7.1 Summary of Accomplishments	98
7.2 Future Work	100
APPENDIX	102
Operating the system.....	102
REFERENCES.....	110

List of Figures

Figure 2-1: (a)Double-pipe heat exchanger manufactured by Armfield Ltd., (b)Schematic diagram of doublepipe heat exchanger showing Cocurrent fluid flow, (c)Schematic diagram of doublepipe heat exchanger showing Countercurrent fluid flow	17
Figure 2-2: (a)Shell and tube manufactured by Armfield Ltd., (b)Schematic diagram of shell and tube hear exchanger showing countercurrent fluid flow.....	18
Figure 2-3: (a)Plate heat exchanger manufactured by Armfield Ltd, (b)Schematic diagram of Plate heat exchanger showing countercurrent flow	18
Figure 2-4: HT 30XC Computer Controlled Heat Exchanger Service Unit (a)Front View, (b)Plan View	21
Figure 2-5: Simplified Block Schematic Diagram of the HT30XC Service Unit.....	23
Figure 3-1: Research path in I-Lab Heat Exchanger project.....	32
Figure 3-2: Chosen Software Architecture.....	36
Figure 4-1: <i>Call Library Function</i> Icon	40
Figure 4-2: Setup dialog box of the <i>Call Library Function</i> Icon	41
Figure 4-3: <i>Call Library Function</i> icon with Input parameters	41
Figure 4-4: Output from the function call displayed in Indicator ‘Value’	42
Figure 4-5: The Controls panel of the Front panel of the LabView control and monitoring software	43
Figure 4-6: The Graph panel of the Front panel of the LabView control and monitoring software	46
Figure 4-7: The PID parameters panel of the Front panel of the LabView control and monitoring software	48
Figure 4-8 The Data Logging panel of the Front panel of the LabView control and monitoring software	49
Figure 4-9: Publishing real-time data and getting real-time input from clients using DataSocket server.....	50
Figure 4-10: DataSocket server manager	51
Figure 4-11: The VI used to publish data to the DataSocket server	52
Figure 4-12: The VI used to get the data from the DataSocket server.....	52
Figure 4-13: The VI used to convert data from Variant to other types.....	53
Figure 4-14: Connecting to the DataSocket Server.....	53
Figure 4-15: Updating (publishing) data to DataSocket server.....	54
Figure 4-16: Version 1.1 of the Client-side Java Applet (November 2001)	55
Figure 4-17: Version 2.22 of the Client-side Java Applet (February 2002)	56
Figure 4-18: Version 3.2 of the Client-side Java Applet (April 2002)	58
Figure 4-19: Version 4.1 of the Client-side Java Applet (June 2002).....	59
Figure 5-1: Database E-R diagram for the database	61
Figure 5-2: Sample code for database access.....	64
Figure 5-3: User authentication login page.....	65
Figure 5-4: Blank user registration page	66
Figure 5-5: User registration page filled with sample data and showing data validation	66
Figure 5-6: Stored procedure in the database to add new users	67
Figure 5-7: Registration confirmation page	67

Figure 5-8: User authentication login page (for already registered users)	68
Figure 5-9: The default schedule management page.....	69
Figure 5-10: Signing up for an available timeslot.....	70
Figure 5-11: Stored procedure to get available time slots.....	70
Figure 5-12: Stored procedure for signing up a user for a timeslot	71
Figure 5-13: Signup confirmation page	71
Figure 5-14: Times that the user has signed up for	72
Figure 5-15: The page showing the schedule of use for all the time slots	72
Figure 5-16: Rescheduling by removing user from a time slot.....	73
Figure 5-17: Stored procedure to remove user from a time slot	73
Figure 5-18: Confirmation of removal of user signup	74
Figure 5-19: Collaboration server-side Application.....	76
Figure 5-20: Database access code.....	77
Figure 5-21: Client Applet: Equipment control and Collaboration.....	79
Figure 6-1: 10.302 Assessment (November 2001): Extended Homework	94
Figure 6-2: 10.26 Assessment (February 2001): Writing Technical Report.....	95
Figure 6-3: 10.450 Assessment (April 2001): Small Project	96
Figure 6-4: UT 354 Assessment (April 2001): Homework Problem	96

List of Tables

Table 2-1: Analog Input Signals from Heat Exchanger to Computer.....	25
Table 2-2: Analog Output Signals from Computer to HT30XC Service Unit.....	26
Table 2-3: Digital Input Signals from HT30XC Service unit to Computer.....	26
Table 2-4: Digital Output Signals from Computer to HT30XC Service Unit	27
Table 5-1: List of Stored Procedures, their purposes and use	62
Table 6-1: Student response : 10.302 : Extended Homework.....	83
Table 6-2: Student response : 10.26 : Writing Technical Report.....	87
Table 6-3: Student response : 10.450 : Small Project	90
Table 6-4: Student response : UT354 : Homework.....	92

Chapter 1. INTRODUCTION

1.1 Why Web-accessible Experiment?

Laboratory experiences can substantially enhance educational effectiveness. Students can compare measured characteristics with theoretical predictions and reflect on discrepancies, limitations, and design criteria. In addition, a hands-on interaction with a physical system allows curiosity-driven exploration and becomes a powerful motivation for students. As a result of all this, students learn better. But many subjects in science and engineering education do not include a laboratory experience. Because traditional laboratory experiments have some disadvantages [1]

- Require more laboratory space and more staffing
- Are more costly
- Require training
- Have safety issues involved
- Have time constraints and scheduling issues involved.

On the other hand, web-accessible experiments have the following advantages

- Minimum staffing requirements
- Single setup for multiple users
- Laboratory equipment can be accessed from anywhere at anytime
- Minimum training
- No safety concerns involved
- With collaboration capability, multiple students can perform the experiment.
- Minimum supervision involved

With technological advancements, software platforms are being developed to enable traditional labs to be web-accessible. The needs for these virtual labs fit perfectly into the modern methods of assembly, delivery and access to educational technology resources adopted by educational institutions around the world. While there are no physical boundaries involved, the labs can be accessed from virtually anywhere in the world, making these labs more efficient. A single laboratory setup can serve educational and research institutions around the globe [2].

Let us consider a scenario: A group of students at MIT are gathering necessary data from an experiment with a laboratory equipment situated at MIT, for their problem set in their graduate laboratory course. After they are done, another group of students from the University of Texas start the experiment with the same equipment. At 2am at night, another group of students from University of Tokyo are taking the necessary data for another experiment by logging into the same equipment at MIT when it is 3pm their local time. Now, in the morning, while the problem set is due in two hours, the first group of students is running the experiment again because they are in need of more data for their problem set. They don't need to rush back to the lab for this. All they have to do is to log into the website of the experiment, and if the equipment is available (no one else is doing an experiment), perform the experiment from their homes.

This might have been seemed somewhat odd a few years ago, but today, this is not only possible, but is becoming an integral part of the educational system.

1.2 I-Lab Heat Exchanger Project - Objectives

The main objective of the I-Lab Heat Exchanger project was to design and implement a real-time, robust and scaleable software system around a laboratory heat exchanger for use in courses and to provide students with

- hands-on experience with heat transfer experiment for them to compare measured characteristics with theoretical predictions and reflect on discrepancies, limitations, and design criteria.
- to provide students with access to the lab equipment anytime and all the time from anywhere even from the comfort of their homes.

The work in Chemical Engineering Heat Exchanger project began with the goal of developing computer simulations that would follow the principles of heat and mass transfer and give the undergraduate students a feel of real laboratory experiment. Later on, simulations were felt to be of secondary importance when a software platform was developed for accessing a custom-made small-scale laboratory heat exchanger. And with positive student response, more effort was made on making this software platform more suitable for course problem, rather than developing simulations.

The initial goals of the Heat Exchanger project were

- Development of a remotely controlled heat transfer experiment for the online use in lectures in 10.302 “Transport Processes”, a core engineering science subject in the Chemical Engineering Curriculum.
- Permit the students to achieve the following without even coming to the lab
 - perform the experiment over the web with real equipment (for homework problems),
 - analyze the data,
 - determine if the observed behavior is consistent with theory and
 - use the data to predict how the heat exchanger would operate at other conditions and with modified designs,

- o explore this as a vehicle for teaching data analysis and report writing in 10.26 “Chemical Engineering Project Laboratory.”

Later on, with successful completion of these goals, we expanded our horizon and scope. We broadened our objectives to develop a powerful educational platform that can be used to study both steady state and transient behavior and that can be used in a variety of courses. The system has been implemented in four courses so far with improvements at each cycle of the development to accommodate new features based upon assessment and evaluation by the students.

The courses are

- 10.302 “Transport Processes”: November 2001
- 10.260 “Chemical Engineering Projects Laboratory”: February 2002
- 10.450 “Process Dynamics, Operations, and Control”: April 2002
- ChE 354 “Transport Processes”: April 2002 (University of Texas at Austin)

1.3 Problem Statement and Challenges

In order to setup an infrastructure, we were faced with challenges in four distinct functional areas with problems associated with each of these. These functional areas are:

- Software for local control and monitoring of the equipment from the server machine,
- Software for remote control and monitoring of the equipment from the client machine,
- User registration, authentication and experiment scheduling,
- Collaboration among multiple users.

Also obtaining a computer-controlled heat exchanger was a challenge. The problems associated with these areas are described below.

Computer controlled Heat Exchanger

The heat exchangers used in the industry are large in size and it is not feasible to use those in an educational setting. So, we worked with Armfield Ltd of UK, a leading manufacturer of small-scale educational laboratory equipment, to develop a small-scale educational heat exchanger equipment that is instrumented with electrically activated valves and switches so as to be controllable from a computer. The instrumentation involves seven variables, four of which are independently controllable. Armfield had to modify their standard laboratory heat exchanger extensively to allow all the functions to be implemented under computer control, with no manual intervention other than setting the equipment up and switching it on. The details of the equipment are described in Chapter 2.

Local Control and Monitoring of the equipment (Server software)

For supervisory control of the equipment, we needed to develop a software to control the equipment. This software should have the following capabilities

- to retrieve data from the equipment,
- control the equipment by inputting the necessary parameter values to the equipment
- publish the retrieved data to the internet,
- allow control of the equipment from any client machine over the internet.

In other words, the software should allow local (server) control as well as remote (client) control of the equipment. This local version of the control and monitoring software is required because the scopes of the experiments allow the students to control only certain specific parameters and for different experiments, these parameters are different. So, this local control and monitoring software should allow the remote user to control only those specific parameters to be controlled and hide the other unnecessary details from them.

Web-access to the equipment (Client software)

We needed to have a web application that can act as a web front end to our system, through which users in remote location can control the equipment and perform experiments.

Scheduling and Security

The nature of the equipment allows only one person to control and perform experiment at one time. Since each course consists of multiple students, there was a need for a scheduling and user authentication system.

Collaboration

Sometimes there are teams of students performing the experiment. We needed a means of allowing multiple students to log on and do the experiment. While the students can collaborate among each other, the system should allow only one person to change the parameters and the others can view the results. The system should also allow students to pass the control of the equipment among each other.

The way these requirements are met and the solutions are architected, is described in details in Chapter 3.

1.4 Outline of the Thesis

The thesis starts in Chapter 1 with a discussion of the necessity of web-accessible experiments, an introduction to the Heat Exchanger experiment and its objectives and the problem statement and challenges.

Chapter 2 discusses the details of the hardware aspects of the Heat Exchanger experiment. The chapter starts with a discussion of the common types of heat exchangers available, and then continues with the discussion on the Armfield computer-controlled Heat Exchanger equipment, its key components and the software driver to access the parameter variables to control and monitor the equipment from a computer.

Chapter 3 discusses the research path taken to develop the experimentation system, the software challenges faced, solutions chosen and the overall architecture of the system.

Chapter 4 describes in details the software components developed for the local control and monitoring of the Heat Exchanger equipment, publishing real-time data to the internet and getting real-time input from client machines over the internet, and the client interface for remote control and monitoring of the equipment.

Chapter 5 discusses the various components of the user management and collaboration support system. The user management system consists of user authentication and experiment scheduling. The collaboration system includes user authentication, user interaction by text chat and the equipment control transfer among users.

Chapter 6 gives an overview of the student assessment and summarizes the assessment results.

Chapter 7 concludes the thesis with a discussion on future work

Chapter 2. EQUIPMENT DESCRIPTION

2.1 Introduction

This chapter describes the details of the hardware aspects of the Heat Exchanger experiment. We begin our discussion with the definition and types of Heat Exchangers available. Then we introduce the Armfield Heat Exchanger equipment used for the Heat Exchanger experiment. We continue our discussion with the key components of the Heat Exchanger Service Unit HT30XC and the description of the USB channel allocation of the HT30XC Service Unit. At the end, we conclude with a description of the Heat Exchanger driver software and the software library functions, which are key to building a software interface to control the laboratory equipment.

2.2 Heat Exchanger and Types

According to Frank P. Incropera and David P. DeWitt [7],

*“The process of heat exchange between two fluids that are at different temperatures and separated by a solid wall occurs in many engineering applications. The device used to implement this exchange is termed a **heat exchanger**, and specific applications may be found in space heating and air-conditioning, power production, waste heat recovery, and chemical processing.”*

Heat exchangers are typically classified according to *flow arrangements* and *type of construction*.

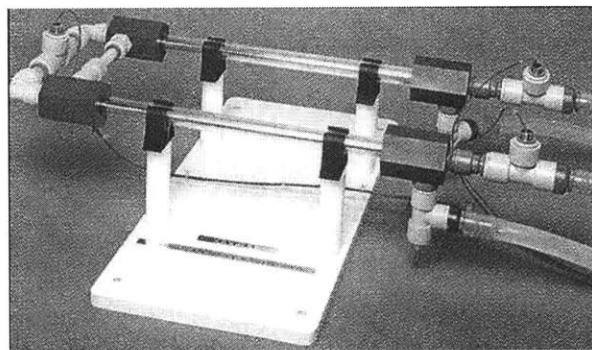
There are three types of *flow arrangements* available

- **parallel-flow or cocurrent:** hot and cold fluid flow in the same direction.
- **counterflow or countercurrent:** hot and cold fluid flow in opposite directions.
- **cross-flow:** hot and cold fluid flow in perpendicular directions to each other.

Several types of heat exchangers are available according to *type of construction*. Some common types are[4]

- Concentric tube or double-pipe
- Shell-and-tube
- Plate

Concentric tube or double-pipe: this is the simplest type of heat exchanger, which has a long, small-diameter tube placed concentrically within a larger tube (Figure 2-1). One fluid passes through the inner tube, and the other fluid passes through the outer tube. This type of heat exchanger is capable of handling high pressures and wide temperature differences, but it provides rather poor thermal performance because of a small heat-transfer area.



(a)

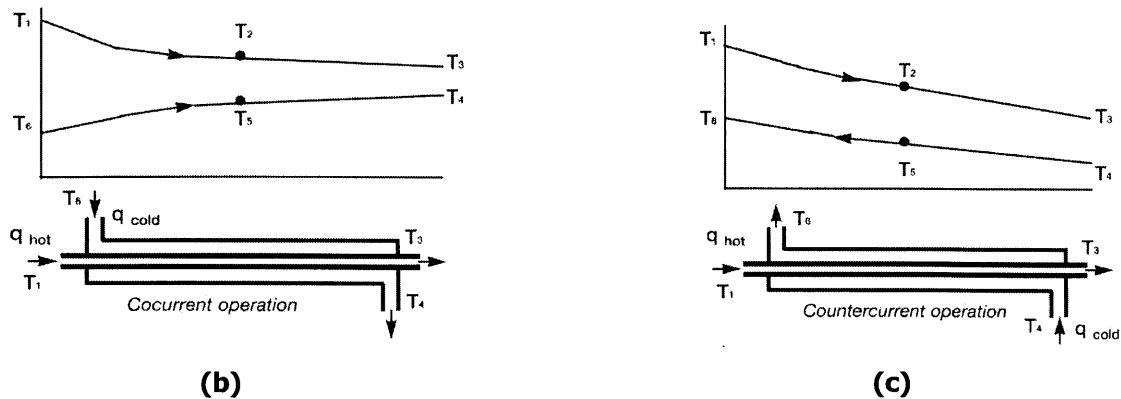
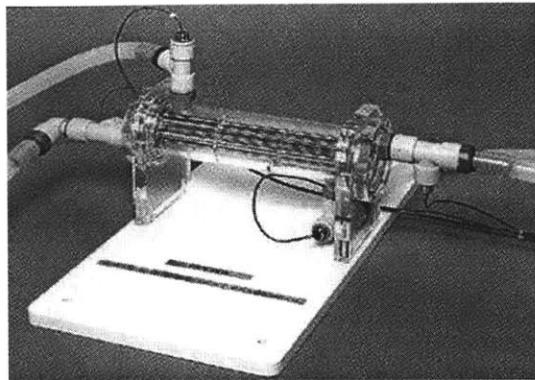
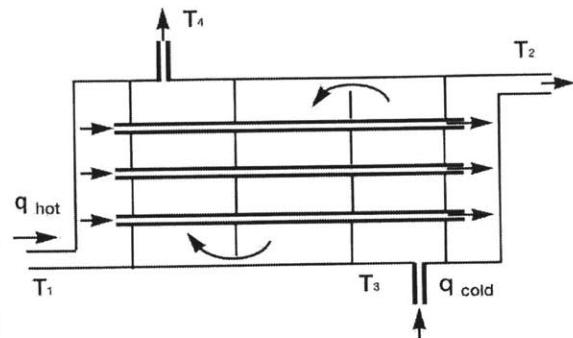


Figure 2-1: (a)Double-pipe heat exchanger manufactured by Armfield Ltd., (b)Schematic diagram of doublepipe heat exchanger showing Cocurrent fluid flow, (c)Schematic diagram of doublepipe heat exchanger showing Countercurrent fluid flow

Shell-and-tube: this is another common configuration. Shell-and-tube heat exchangers (Figure 2-2) consist of a bundle of parallel tubes that provide the heat transfer surface separating two fluid streams. The tube-side fluid passes axially through the inside of the tubes; the shell side fluid passes over the outside of the tubes. Specific forms differ according to the number of shell-and-tube passes, and the simplest form, which involves single tube and shell passes, is shown in Figure 2-2 (b). Baffles external and perpendicular to the tubes are usually installed to direct the flow across the tubes, to provide tube support and also to increase the convection coefficient of the shell-side fluid by inducing turbulence and a cross-flow velocity component. The thermal performance of such an exchanger usually surpasses a tubular type but is less than a plate type. Pressure capability of shell-and-tube exchangers is generally higher than a plate type but lower than a tubular type.



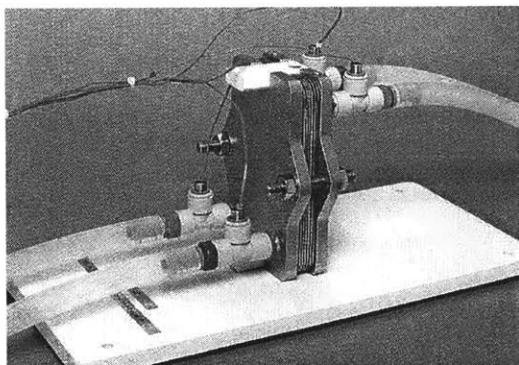
(a)



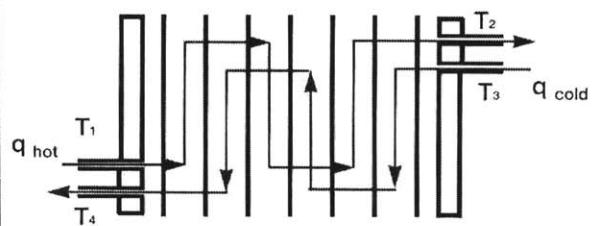
(b)

Figure 2-2: (a)Shell and tube manufactured by Armfield Ltd., (b)Schematic diagram of shell and tube hear exchanger showing countercurrent fluid flow

Plate heat exchanger: Plate heat exchangers (Figure 2-3) consist of a stack of parallel thin plates that lie between heavy end plates. Each fluid stream passes alternately between adjoining plates in the stack, exchanging heat through the plates. The plates are corrugated for strength and to enhance heat transfer by directing the flow and increasing turbulence. These exchangers have high heat-transfer coefficients and area, the pressure drop is also typically low, and they often provide very high effectiveness. However, they have relatively low pressure capability.



(a)



(b)

Figure 2-3: (a)Plate heat exchanger manufactured by Armfield Ltd, (b)Schematic diagram of Plate heat exchanger showing countercurrent flow

2.3 Armfield Heat Exchanger Equipment

2.3.1 Introduction

Armfield Ltd of UK builds a range of small scale heat exchangers which represent the common types of heat exchanger found in industry and demonstrate different techniques for indirect transfer of heat from one fluid stream to another [3]. We worked with Armfield Ltd and acquired three common types of heat exchangers: Double-pipe or Tubular type (HT31), Plate type (HT32) and Shell-and-Tube (HT33) and one Service Unit (HT30XC).

The Service Unit (HT30XC) is a bench-top apparatus on which the heat exchangers may be individually mounted. The Service Unit provides the necessary services and measurement facilities for investigation and comparison of the different heat exchanger working principles and operating characteristics. The ability to change the type of exchanger quickly, without the use of tools, and the fast response of the system to changes in water flow, temperature etc. allow the relevant experiments to be carried out in a relatively short period of time.

The HT30XC Heat Exchanger Service Unit provides

- streams of
 - hot water (heating fluid)
 - cold water (process fluid)
- at variable flow rates

to the heat exchanger under evaluation. The HT30XC is connected to the computer through an USB (Universal Serial Bus) port and comes equipped with a software driver (described later in the chapter) for windows operating system. It is designed to operate from a Windows computer and all the parameters are computer controlled, with no manual intervention other than setting the equipment up and switching it on.

2.3.2 Computer Requirements

The computer that controls the Heat Exchanger equipment should run Microsoft Windows 98 (or above) and have a USB interface. For best results, it should have a minimum processor speed of 500 MHz, have at least 64 MB of RAM and 20 Mbytes of free hard disk space.

The Computer is able to control

- The flow rates in both fluid streams.
- Reversing the flow in one stream in order to demonstrate both cocurrent and counter-current flow conditions.
- Direct control of the heater modulation, thus allowing temperature control of the hot fluid to be achieved.

The controls are explained in details in the next section.

The computer is able to display

- Temperature readings from up to ten thermocouples fitted to the heat exchangers.
- Fluid flow rates from both the hot and cold water streams.
- Status information from the HT30XC Service Unit.

2.3.3 Main components of the HT30XC Service Unit

The Service Unit is equipped with standard mounting arrangement and service connections for any one of the interchangeable heat exchangers. Following are some of the key components of the Service Unit (Figure 2-4).

1. Hot Water Vessel: This vessel (1) is situated on top of the Service Unit. The water in this vessel is electrically heated. The heating element incorporates an over-temperature thermostat that prevents the water being heated beyond a nominal 85 °C. Also in the hot water vessel is a conductivity level sensor (2) that prevents either the heater or pump being turned on unless the vessel is sufficiently full of water.

2. Heater: The electric supply to the heater (3) is modulated by a solid state relay (SSR), which is located inside the plinth base. The modulation signals to the SSR are provided directly by the computer, via the USB interface. This modulation signal is transmitted through a digital channel (Channel 2: described in details in the next section), which means that the only values that can be passed are 0(heater off) and 1(heater on). This requires the need of implementing a PID algorithm in the controlling software in order to keep the hot fluid inlet temperature steady.

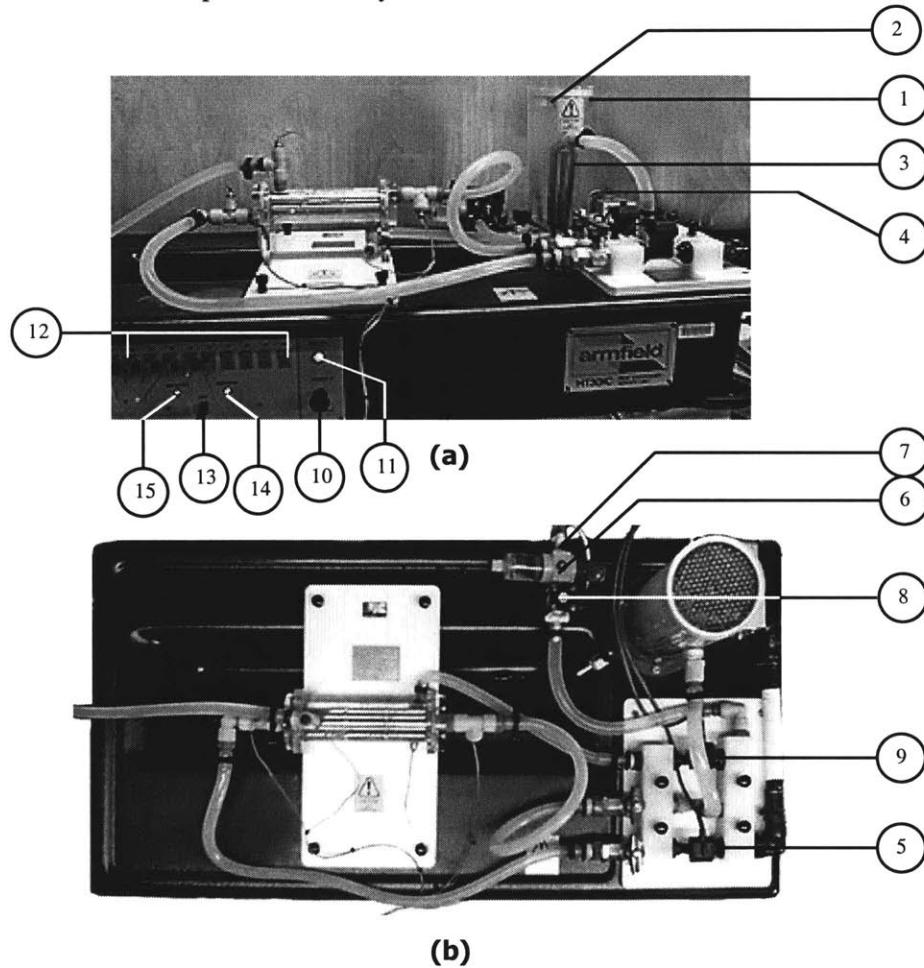


Figure 2-4: HT 30XC Computer Controlled Heat Exchanger Service Unit
(a)Front View, (b)Plan View

3. Pump and Flow meter: Mounted by the side of the hot water vessel is the gear pump (4), driven by an electric motor, which is used to circulate the water through the heat exchanger and back to the vessel. The rotational speed of the motor/gear pump, and

hence the hot water flow rate can be controlled from the computer. The hot water flow rate is measured by the in-line flow meter (5) and displayed on the computer. The hot water system includes a strainer on each side of the flow meter, contained within the acrylic mounting blocks. These strainers protect the delicate paddle wheel mechanism of the flow meter from any particles, dirt, hair etc. that may find their way into the water.

The direction of rotation of the pump can be changed by using the computer to control a changeover relay mounted in the plinth base, thus achieving a cocurrent or a counter-current flow. This relay reverses the polarity of the electrical voltage applied to the motor.

4. Cold water pressure regulator: The cold water flow (the process flow) for the heat exchanger is derived from the local mains supply. A pressure regulator (6) complete with integral filter/strainer isolates the HT30XC from the minor variations in the pressure of this supply. The cold water supply is connected to the inlet (7) of the pressure regulator. The flow rate through the heat exchanger is then controlled using an electronically driven proportioning solenoid valve (8). Again this valve is controlled from the computer. A second in-line flow meter (9) measures the cold flow rate.

5. Flexible tubes: Flexible tubes are used to connect the circulator to each heat exchanger and quick release fittings allow rapid connection. Red collars identify the hot water connections and a blue collar identifies the cold water connection.

6.Others: A panel on the front of the Service unit contains the ‘Standby/Enable’ switch (10) with ‘Control’ indicator (11), the ‘Emergency Stop’ switch with ‘Process’ indicator and input connectors (12) for up to 10 standard ‘k’ type thermocouples, labeled T1 to T10. The thermocouples are supplied with the individual heat exchangers and appropriately connected and marked. Also mounted on the panel are the connector (13) for the USB interface for connection to the computer, and two USB status indicators. A red ‘power’ LED (14) lights when the unit is connected to the PC and a green ‘active’

LED (15) lights when the unit has been recognized by the PC. The USB interface is located behind the front panel.

Figure 2-5 shows a simplified Schematic Block Diagram of the HT30XC Service Unit. This diagram takes into consideration the HT33 type Shell-and Tube Heat Exchanger with 4 thermocouple sensors.

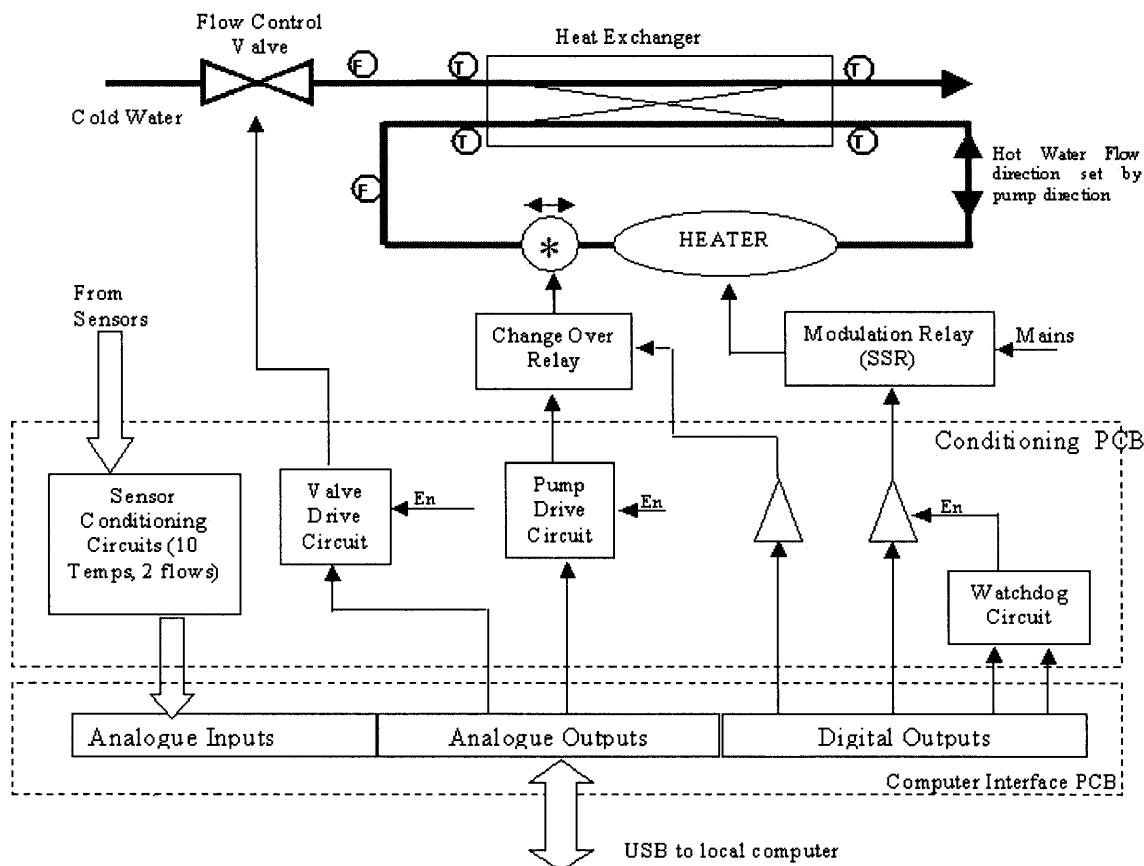


Figure 2-5: Simplified Block Schematic Diagram of the HT30XC Service Unit

Cold water flows from the mains through a flow-control valve, which is controlled by the Valve Drive Circuit. This Valve drive circuit is controlled by the computer through Analog input Channel. Information from the thermocouples (T) and the flow meters (F) are sent to the Sensor Conditioning Circuits, which, eventually are sent to the computer through USB as Analog Inputs. The motor receives two types of signals from the computer. One is flow direction signal through the Digital Output channel, which sets the Change Over Relay and we achieve either a cocurrent or a counter-current hot fluid flow.

The other is the pump speed signal, which is received from the computer through the Analog Outputs channel to the Pump Drive Circuit, which eventually sets the speed of the motor/pump. The heater on/off is controlled by the Modulation Relay, which receives its input from the Computer through the Digital Output channel. The Watchdog Circuit ensures the connection between the computer software interface and the HT30XC Service Unit. On the event of a connection failure, the watchdog circuit shuts down the entire Service Unit. The Analog and Digital Input and Output channels and their functions are described in details in the following section.

2.4 Channel Allocations

The interface between the Armfield heat exchanger HT30XC Service Unit and the computer is a Universal Serial Bus (USB) interface, meeting the standard Microsoft protocols.

The interface is capable of passing data on 26 channels, as described below:

- **Analog Inputs (from HT30XC to computer):** 8 differential channels or 16 single ended channels, each with -5V to 5V signals digitized into a 12-bit number. The interface will pass a value between -2047 and 2047 to the computer.
- **Analog Outputs (from computer to HT30XC):** 2 channels, each with -5V to 5V signals, taken from a 12-bit number.
Computer must pass a value between -2047 and 2047 to the unit.
- **Digital Inputs (from HT30XC to computer):** 8 channels each receiving a 0 or 1 by the computer.
- **Digital Outputs (from computer to HT30XC):** 8 channels each passing a 0 or 1 from the computer.

The channel allocations for the HT30XC are tabulated below. It is assumed that the type of Heat Exchanger used is the HT33 Shell-and Tube type.

Table 2-1: Analog Input Signals from Heat Exchanger to Computer

Channel	Code	Use (HT33)	Scaling
0	T1	Hot Water In (counter-current)	0V = 0°C, 5V = 200 ° C
1	T2	Hot Water Out (counter-current)	0V = 0°C, 5V = 200 ° C
2	T3	Cold Water In	0V = 0°C, 5V = 200 ° C
3	T4	Cold Water Out	0V = 0°C, 5V = 200 ° C
4	T5		0V = 0°C, 5V = 200 ° C
5	T6		0V = 0°C, 5V = 200 ° C
6	T7		0V = 0°C, 5V = 200 ° C
7	T8		0V = 0°C, 5V = 200 ° C
8	T9		0V = 0°C, 5V = 200 ° C
9	T10		0V = 0°C, 5V = 200 ° C
10	F1	Hot Water Flow	0V = 0 L/min, 5V= 5 L/min
11	F2	Cold water Flow	0V = 0 L/min, 5V= 5 L/min
12		Not used	
13		Not used	
14		Not used	
15		Not used	

Table 2-2: Analog Output Signals from Computer to HT30XC Service Unit

Channel	Code	Description	Scaling
0	P1	Hot Water Pump Speed	0V = stopped, 5V = full speed
1	V1	Cold Water Valve Setting	0V = Closed, 5V = Fully open

Table 2-3: Digital Input Signals from HT30XC Service unit to Computer

Channel	Title	Description	Scaling
0	24V Monitor	Indicates that the power on relay has switched and the 24V power supply used to drive the pump and valve is functional	0 = no 24V 1 = 24V OK
1	Power on Relay	If power on has been requested, but this signal remains low, the emergency stop or the enable switches may be OFF	0 = no power 1 = power on relay coil
2	SSR Monitor	If SSR output is high, but this signal is low, there is a fault in the hot water circulation system, e.g. no water or heater over temperature	0 = no signal at SSR 1 = Heater on requested
3		Not Used	
4		Not Used	
5		Not Used	
6		Not Used	
7		Not Used	

Table 2-4: Digital Output Signals from Computer to HT30XC Service Unit

Channel	Title	Description	Scaling
0	Power on request	Allows the power on relay to be energised, subject to the presence of an appropriate watchdog pulse	0 = Power Off 1 = Power On
1	Watchdog Pulse	Pulsed signal to keep the watchdog circuit energised, enabling the Heat Exchanger bench power to be turned on	Pulsed signal, min rate 1 pulse every 5 seconds
2	SSR Drive	Time modulated signal controlling the hot water heater Solid State Relay (SSR)	0 = heater off 1 = heater on
3	Pump Direction	Controls the change-over relay which reverses the hot water pump direction	0 = counter-current 1 = co-current
4	Stirrer On	Not used on HT33	0 = off 1 = on
5		Not Used	
6		Not Used	
7		Not Used	

2.5 Heat Exchanger Driver Software and Library Functions

2.5.1 Software Driver Files

The Heat Exchanger equipment is provided with the driver software to allow the USB interface to be controlled from computer. This driver software contains a Dynamic Link Library file (DLL), which can be accessed from the control and monitoring software on the computer to exchange parameter values with the HT30XC Service Unit. Dynamic Link Libraries can be defined as:

“A library of executable function or data that can be used by a Windows Application. Typically, a DLL provides one or more particular functions and a program accesses the functions by creating either a static or dynamic link to the DLL.”

In this case, the DLL file provides a means to access the input and output parameters of the HT30XC heat exchanger Service Unit. The input and output data is accessed by a standard function call to the DLL.

The software driver consists of the following files and the files are installed at the following locations in the hard disk drive:

```
%windir%\INF\OTHERS\ARMFIELD_LTDTHERMUSB.INF  
%windir%\INF\OTHERS\ARMUSB.INF  
%windir%\SYSTEM32\DRIVERS\THERMUSB.SYS  
%windir%\SYSTEM32\DRIVERS\ARMUSB.SYS  
%windir%\SYSTEM\ARMIFD.DLL
```

where %windir% is the windows installation directory. In case of Windows 2000,

%windir% = C:\WINNT

The first two files tells the computer how to recognize the HT30XC IFD5 when it is plugged in to the PC, where the IFD5 is the Data acquisition card of the HT30XC and is installed within the HT30XC base unit. The next two files are the drivers for the IFD5s

USB interface and the last file is a library file, used to pass data between the user program and the IFD5 driver.

2.5.2 IFD5 Driver Function Calls

The driver for the Armfield IFD5 data logger is accessed using function calls to the dynamic link library file ARMIFD.DLL. There are four basic calls that can be made to the library file, based on the four types of data I/O described in the previous section. These functions must be supplied with a full set of variables of the correct format. All of the variables passed are 32-bit integer types.

The analog channels use values between -2047 and 2047, relating to -5V to +5V on the apparatus. The digital channels use either a 0 or a 1 for OFF or ON respectively.

Read Analog (Access Analog Input Channel)

This call returns a value from one of the analog channels. The syntax for the call is:

```
procedure ReadAnalog(var channel: integer; var value: integer);  
      stdcall;
```

The **stdcall** directive indicates that the call is handled in a way which is recognizable by most programming languages. The channel number should be selected as follows:

- Channels 0-7 for differential channels
- Channels 0-15 for single ended channels
- Channels 16-31 for multiplexed channels (where appropriate)

The value returned will be an integer ± 2047 corresponding to $\pm 5V$. A value of 9999 indicates an error.

Write Analogs (Access Analog Output Channel)

This call sends values to the two analog output channels. The syntax is:

```
procedure WriteAnalogs (var A01: integer; var A02: integer);  
    stdcall;
```

The values sent should be between ± 2047 corresponding to $\pm 5V$.

Read Digitals (Access Digital Input Channel)

This call returns the values from one of the eight digital channels. The syntax for the call is:

```
procedure ReadDigital (var channel: integer; var value: integer);  
    stdcall;
```

The return value will be 0 if the channel is off or 1 if the channel is on. A value of -1 indicates an error.

Write Digitals (Access Digital Output Channel)

This call writes values from the eight digital output channels. The syntax for the call is:

```
procedure WriteDigitals(var DO1: integer; var DO2: integer;  
    var DO3 : integer; var DO4 : integer;  
    var DO5 : integer; var DO6 : integer;  
    var DO7 : integer; var DO8 : integer);  
    stdcall;
```

Chapter 3. RESEARCH PATH AND THE SOFTWARE ARCHITECTURE

3.1 Introduction

This chapter describes the research path chosen to design and deploy the Heat Exchanger Experiment and the Software architecture of the entire system. Some discussion is also made on alternate solutions that were considered and the issues involved in choosing the solution that was deployed.

3.2 Research path

As discussed in Section 1.3, we were faced with challenges in four distinct functional areas while designing the system:

- Software for local control and monitoring of the equipment from the server machine,
- Software for remote control and monitoring of the equipment from the client machine,
- User registration, authentication and experiment scheduling,
- Collaboration among multiple users.

Also there was an issue with obtaining the appropriate hardware from the manufacturer.

There was time constraint involved from the beginning of the research work. The actual work on the design and development of the system started on February 2001 and the first

deployment of the system in a course was scheduled on October 2001. So, there was only seven months in which time the Heat Exchanger equipment needed to be manufactured and delivered, the server control software was to be developed and tested with first test data and then with the actual heat exchanger equipment, the client-side software interface for remote control of the equipment was to be developed, the communication between the server and client softwares had to be tested. So, the research path was chosen keeping in mind all these issues.

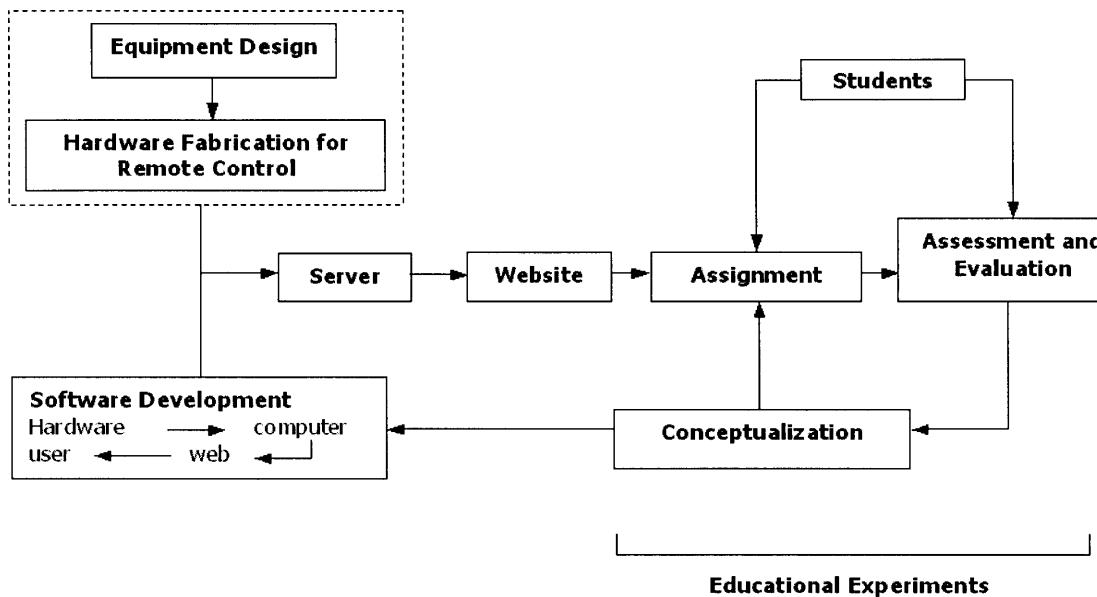


Figure 3-1: Research path in I-Lab Heat Exchanger project

The path we chose to go about solving the problem is shown in figure 3-1. We started designing the server and client components at the same time that we ordered the equipment. We built a prototype of the client/server system to test with test data. On successful completion of the prototyping, we started building the website and user authentication and scheduling system, which was not available on the first release and deployment. The collaboration system was on our low priority list, so we started developing it after all the other components were ready to go. Then after each use of the system in a course, we modified the system according to student feedback from the assessment. Then we implemented another iteration of the development process with a newer version of the system as the end product at the end of the iteration.

3.3 Software Solutions

For each of the software challenges discussed in the previous section and in section 1.3, we were faced with multiple solutions with constraints in each of them. Considering different factors, we chose to move forward with a specific set of solutions. These are discussed in this section and the next section.

Local Control and Monitoring of the equipment (Server software)

For the local control and monitoring of the heat exchanger equipment, we had to make use of the driver software that came with the equipment. For accessing the different parameters, we had to access particular channels of the IFD5 Data Acquisition interface of the HT30XC Heat Exchanger Service Unit, by making appropriate driver function call (described in Sections 2.4 and 2.5).

The next phase was choosing a suitable programming language to write the software and provide all the functionalities we needed. While comparing Java, C++, Visual Basic and LabView [a 4GL (Fourth Generation Programming Language) developed by National Instruments: allows developing programs graphically using block diagrams], we found that using LabView on the server side cuts down on the development time considerably.

LabView was designed to work as instrumentation software and the functionalities and the ‘look and feel’ matched our requirements.

Also in terms of web-enabling capability, it is easier to publish the data on the web by making use of the publish-subscribe model of DataSocket Server [a National Instruments product that allows publishing and subscribing to data by providing specific URI]. This is described in details in Section 4.3

Web-access to the equipment (Client software)

National Instruments provide two sets of APIs to access the data published by DataSocket server. One set is for Java and the other set for ActiveX. National Instruments also provides a set of ActiveX controls to develop web applications containing virtual

instrumentation with knobs, dials, charts etc. But the ActiveX has its limitations. Only Internet Explorer browser supports ActiveX. Netscape doesn't support ActiveX and the only browser available in Athena was Netscape Navigator.

Scheduling, Registration and User Authentication system

While designing the scheduling, registration, and user authentication system, we could make use of any of the common scripting languages available nowadays like ASP (Active Server Pages), JSP (Java Server Pages), CFM (Cold Fusion Markup Language), CGI (Common Gateway Interface).

On the database side, we could make use of MSAccess, MS SQL server, Oracle.

Collaboration

The collaboration really depended on the choice of client side software platform. The client side of the collaboration component could be embedded in the client-side interface of control and monitoring of the equipment, or it could be separate. The Server-side component could be developed in any standard programming language platform that provides socket connection, database access and multithreading capability.

3.4 Chosen Solution & Reasons

The main reason of choosing LabView as the server side equipment control software was the time constraints we had. We cut down on the development time considerably using LabView. So, we could focus more on developing other components of the system.

Also we were able to publish the data easily on the web in a format that the client software can access.

We chose Java on the client side because it is browser independent. The client interface should be able to run on any browser in any operating system.

We chose ASP.NET to develop the client scheduling, registration and authentication system. ASP.NET was chosen for the ease of development, database integration and ease of configuration of the hosting server in our case, IIS 5.0 (Internet Information Services). Initially, MS Access was used for developing the system database, but later on we migrated to MS SQL server 2000, which gave us more control over how data was accessed by making use of Stored Procedures.

The collaboration system was developed in Java and was integrated in the client side Java applet. The collaboration system handles the user authentication system through a server side Java application that again accesses user data in the SQL server database through JDBC.

3.5 Software Architecture

Figure 3-2 shows the Software Architecture of the Heat Exchanger experiment. It has three distinct components to it.

- Client side software components
- Server side software components
- Heat Exchanger Equipment

The server-side software components include the following

- The LabView software for local control and monitoring of the Heat Exchanger equipment. The Heat Exchanger is connected to the computer through USB port and the software accesses the Heat Exchanger equipment through the heat exchanger driver software (described in Section 2.4 and 2.5).
- The DataSocket server for publishing real-time data and to get real-time input from the client-side Java interface (described in details in Section 4.3).

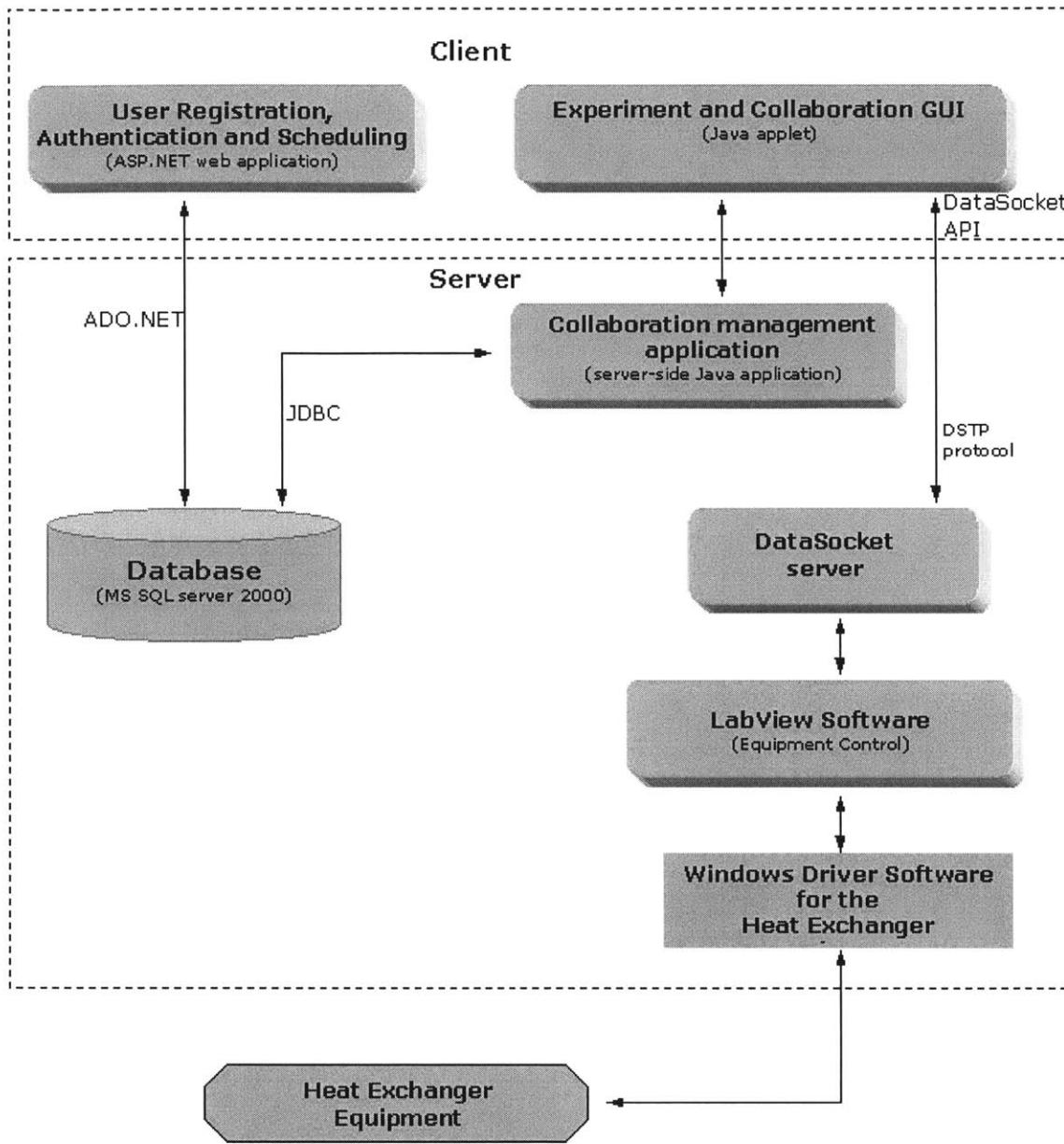


Figure 3-2: Chosen Software Architecture

- The Collaboration management application that handles the user authentication and manages the collaboration among multiple users (described in details in Section 5.4.2). This application authenticates users by connecting to the SQL server database through JDBC.
- The Database (MS SQL server 2000) stores information about users and also the schedules (described in details in Section 5.2).

The client-side software components include

- A User Authentication, Registration and Scheduling system to manage the users that perform the experiment (described in details in Section 5.3). This system connects to the database using ADO.NET. The user data is added to the database at Registration time. The user data is associated with particular signup times when the user signs up for that timeslot.
- The Experiment and Collaboration GUI (described in details in Sections 4.4 and 5.4.3). This Java Applet connects to the Collaboration management application through Sockets and authenticates the users and allows collaboration among multiple users. This Java Applet connects to the DataSocket server through DataSocket API by providing valid URIs for the data points. The Java applet can either listen to these URIs for update on data values or publish parameter values to some URIs for the LabView application to receive and pass them to the equipment. This way, the users can control and monitor the Heat Exchanger equipment from remote locations using only the web browser.

Chapter 4. EQUIPMENT CONTROL AND MONITORING

4.1 Introduction

This chapter discusses the software components developed for the local control and monitoring of the Heat Exchanger equipment, publishing the real-time data and getting real-time user input over the web with DataSocket server and the software interface for remote control and monitoring of the equipment. The next chapter discusses the user management system and the collaboration system.

4.2 Server Control and Monitoring using LabView

LabView is a graphical programming language, where block diagram or VIs (Virtual Instruments) are used to develop Graphical User Interfaces to monitor and control instruments. While this programming language has many advantages like ease of use, very short development time, a very large library of Graphical Instrument control tools like knobs, dials, charts, LabView is not as robust as other object oriented languages like Java and C++ in terms of multi-threading, inheritance, defined class hierarchy etc.

The reason for developing a server-side equipment control software was that in many instances external users are given limited control and this is because the scope of the homework problems limits the usage of the equipment. So, it wasn't necessary to burden the students with the unnecessary load of all the controls. So, it was decided that the

server-side software would have all the capability and the client side Java applet would have limited capability and these limited capabilities would be controlled from the server-side software. This is explained in details later in the chapter.

LabView was used in this project in order to cut down on the development time since the tight time-schedule was a primary concern. Also, the rich libraries of Graphical tools are very user-friendly to develop and use.

While developing the control and monitoring software, some of the key challenges faced were:

- accessing the driver files to pass parameter values from and to the Heat Exchanger equipment
- switching of control between the user and the administrator.
- develop an efficient heater control system using PID algorithm
- data logging system, triggered from the Java applet

4.2.1 Accessing equipment Driver files

The Driver files and the Functions that are to be called in order to access the Heat Exchanger equipment are described in Section 2.5.2. This section shows in step by step how to access these functions. This section assumes that the user has knowledge of the basics of LabView.

Step 1

We load LabView and choose *new VI* from the initial menu screen. This gives us a blank document in which to work. Alternatively, we could add the function access method to an existing VI. We then select *View Diagram* from the window menu. We make sure that the tools palette and functions palette are visible.

Step 2

From the functions palette, we select *Advanced* and then *Call Library Function* (Figure 4-1). We Place the icon on the diagram.

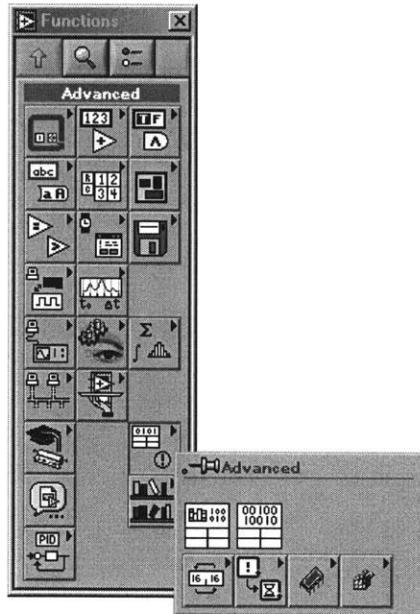


Figure 4-1: Call Library Function Icon

Step 3

We then double click on the icon with the selector tool to access the setup page (Figure 4-2). We click on the *browse* button at the top of the page, and locate the file C:\WINDOWS\SYSTEM\ARMIFD.DLL. We click on the *Function Name* box and type 'ReadAnalog'. We check that the *Calling Conventions* is set to 'stdcall(WINAPI)' and that the function is set to 'Run in UI Thread'.

We click on the *Add a Parameter After* button. The parameter area will change, allowing various choices. We name the parameter 'Channel', set the *Type* to 'numeric', set the *Data Type* to 'Signed 32-bit Integer' and set *Pass* to 'Pointer to Value'. These settings apply to all of the parameters used by the library file. We repeat this step so that there are two parameters, naming the second one 'Value'. When this is done, we click *OK* to close the page.

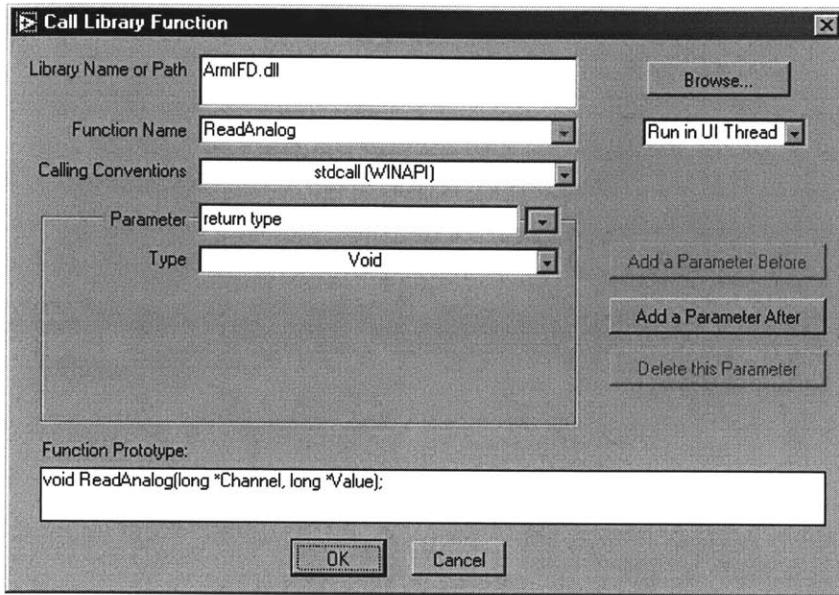


Figure 4-2: Setup dialog box of the *Call Library Function* Icon

Step 4

The icon on the diagram has four cells on it (Figure 4-3), representing the two parameters before and after being passed to the library file. The column on the left contains the input parameters, while the column on the right contains the outputs.

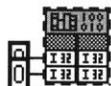


Figure 4-3: *Call Library Function* icon with Input parameters

We popup (click with the right-hand mouse button) the top left-hand cell and choose *create constant*. We then type ‘0’ in the box that appears. We repeat this step for the second cell in the left-hand column. This means that the channel number and output will be set to zero before passing to the library file.

We popup on the bottom cell in the right-hand column and choose *Create Indicator*. A blue box will appear with the caption *Value* (Figure 4-4). We click on *View Panel* in the

window menu to see the front panel. There should be an indicator box present, titled 'value'.

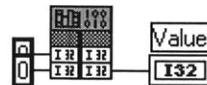


Figure 4-4: Output from the function call displayed in Indicator 'Value'

Step 5

We save the VI using the file menu, then click on the *run continuously* button. The value in the indicator box will update continuously, showing the value being read by the IFD5 on channel 0. This procedure can be repeated in order to create indicators for analog channels described in Table 2-1.

The same procedure is used to access the Digital Input Channels described in Table 2-3. Writing to the Analog and Digital channels are also similar. The function call parameters are described in Section 2.5.2 and the Output channels are described in Table 2-2 and Table 2-4.

4.2.2 Controls and Indicators on the Front Panel

The front panel (Figure 4-5) of the control and monitoring software has 5 panels for different purposes. These panels are Controls, Graphs, I/O, PID parameters and Data Logging.

4.2.2.1 Controls panel

The Controls panel (Figure 4-5) has all the controls and indicators to pass values to the equipment and from the equipment through USB. The POWER switch on the top turns the equipment on or off. This is a digital control accessing the equipment through Digital

Output function call. The Power Indicator lights up when the equipment is turned on. This is the output from the Digital Input function call.

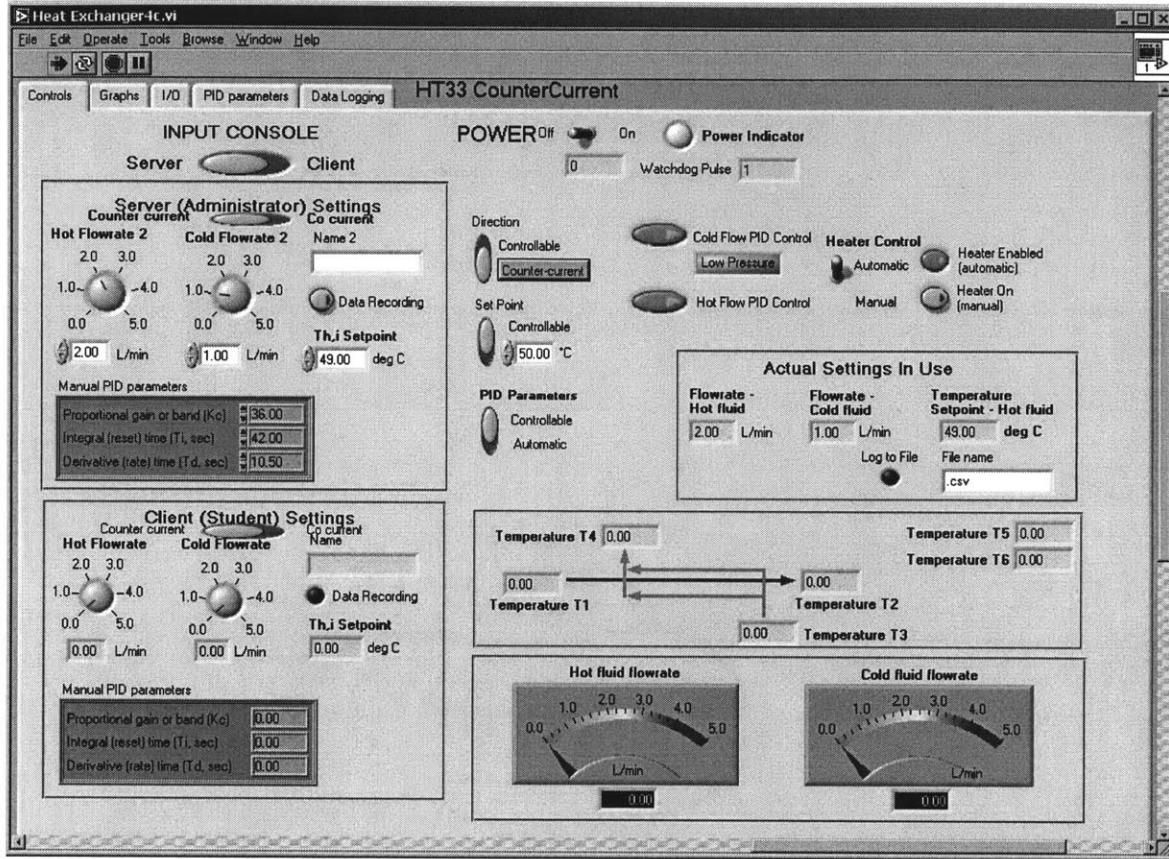


Figure 4-5: The Controls panel of the Front panel of the LabView control and monitoring software

On the left side of the panel is the INPUT CONSOLE. At the top is a switch that switches the control of the equipment between the Administrator (Server) and the user (Client). When in client mode, the Server Administrator doesn't have any control over the variable parameter values passed to the equipment. The software is in listen mode and listens to particular DataSocket addresses (described in the next section) for values to these parameters and passes the values to the equipment as they become available. The client applet passes these values to the DataSocket addresses. But while in Server mode, the client machines cannot change the parameter values. The values are changed from the local machine. The parameters to be controlled are

Primary controls

These are the parameter values that are sent directly to the equipment

- The flow direction (Counter current or Cocurrent)
- Hot fluid flow rate: the flow rate of the hot fluid stream is controlled by varying the speed of the pump and this is done by accessing one of the channels through Analog Output function call of the dll
- Cold fluid flow rate: the flow rate of the cold fluid stream is controlled by controlling the valve (described in section 2.3.3) through the Analog Output function call and accessing the appropriate channel
- Heater Control: The heater in the hot water reservoir can only be turned on or off. This heater is used to control the temperature of the hot fluid stream. Although this is a direct control parameter, it is controlled from within the PID (proportional-integral-derivative) algorithm.

Secondary controls

These are the control parameters that are processed within the software and depending on the output, the software either sends any value to the equipment, or performs some other operation.

- Temperature Setpoint: the hot water inlet temperature of the equipment is controlled by controlling the heater. The heater can only be turned on or off. So, we devised a PID algorithm that determines how much of the time the heater is turned on or off. It takes a certain time interval and then depending on the PID values, the setpoint value and the current thermocouple reading at the hot water inlet (received from the equipment), decides what percent of this time interval, the heater is turned on and what percent it is turned off and thus reaches the set point temperature and keeps it steady.
- PID (Proportional-Integral-Derivative) parameters: These are the parameters used in the PID algorithm to compute how much of the time the heater is turned on or off. Depending on the output of the algorithm, the heater is either turned on or off.

- Data Recording: This consists of a text box to input the filename and a data recording button. The software creates a file with the same name as specified at a defined location (described later in this section). When the button is pressed, data recording starts and when the button is pushed again, data recording stops.

The actual parameter values that are passed to the equipment are shown on the right side of the panel. The administrator can also give limited control capability to the students by turning off some of the controls. This is shown in the middle of the panel. The direction of flow, set point and PID parameters can be turned to `Controllable` and `Automatic` depending on the scope of the experiment. When these controls are turned off, the corresponding controls on the client applet are grayed out.

Shown at the bottom right part of the panel are the actual temperature and flow values. The temperature values are received from thermocouples mounted at different flow locations. For the Shell and Tube HT33 Heat Exchanger, the thermocouple locations are:

- Hot fluid inlet
- Hot fluid outlet
- Cold fluid inlet
- Cold fluid outlet

These are the four standard thermocouple and temperature readings received from the equipment. For the Tubular HT31 Heat Exchanger, two additional thermocouples are provided at the middle of each stream. There is a provision of a maximum of 10 thermocouples.

The flow rate values are received from the two flow meters mounted at the two flow streams with a maximum of 5 L/min.

The temperature and flow rate readings are accessed by making Analog Input function calls.

4.2.2.2 Graphs Panel

The graphs panel holds three strip charts showing the Hot inlet temperature, Cold flow rate and the Hot flow rate.

The temperature chart shows the following parameters

- temperature set point set by the user
- the instantaneous value of the temperature output from the thermocouple at the hot water input stream
- the mean value of the temperature over some period of time
- The PID output value showing the percent of time the heater is on or off.

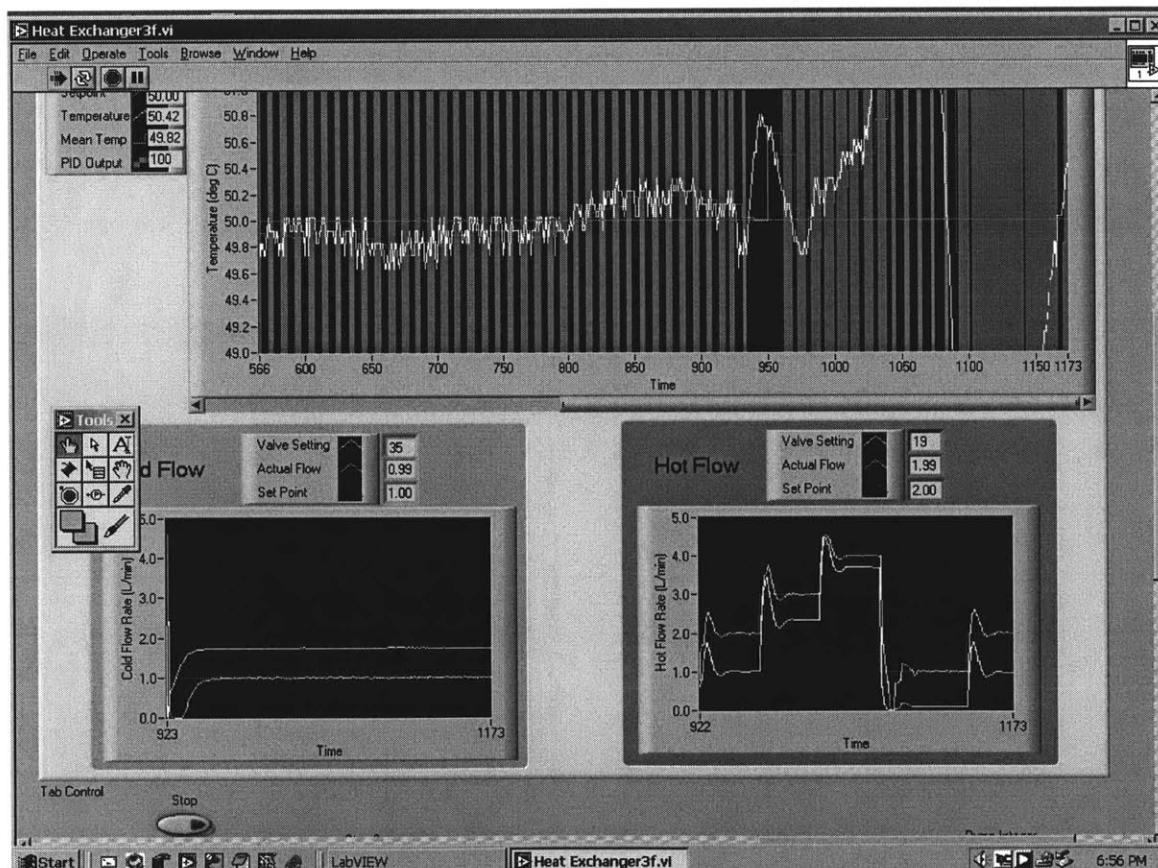


Figure 4-6: The Graph panel of the Front panel of the LabView control and monitoring software

The flow rates are also controlled by PID algorithm. The Cold flow rate chart shows the actual valve setting, the actual flow received from the output from the flow meter and the set point set by the user for the flow rate. The Hot flow rate chart shows the same parameters, but for the hot fluid stream.

4.2.2.3 I/O Panel

The I/O Panel shows the actual integer values received from the Analog and Digital Input channels and passed to the Analog and Digital Output channels. These values are then transformed in to more meaningful values. For example, the analog channels use values between -2047 and 2047, relating to -5V to +5V. For temperatures, 0V=0 °C and 5V=200 °C. We receive values between -2047 and 2047 from the Analog Input Channels; we have to transform these values to represent a temperature value in °C. The raw value is shown in the I/O Panel and the transformed value is shown in the Control panel. This panel provides a means for checking whether the USB interface is working properly.

4.2.2.4 PID parameters Panel

This panel (Figure 4-7) displays all the PID input parameters for the threes PID controls used for

- Controlling the hot fluid inlet temperature
- Controlling the hot fluid flow rate
- Controlling the cold fluid flow rate

The default values for these PID controls are the optimum values at which the system performs well. The PID parameters can be tuned to get better results. For this, more tests will have to be performed.

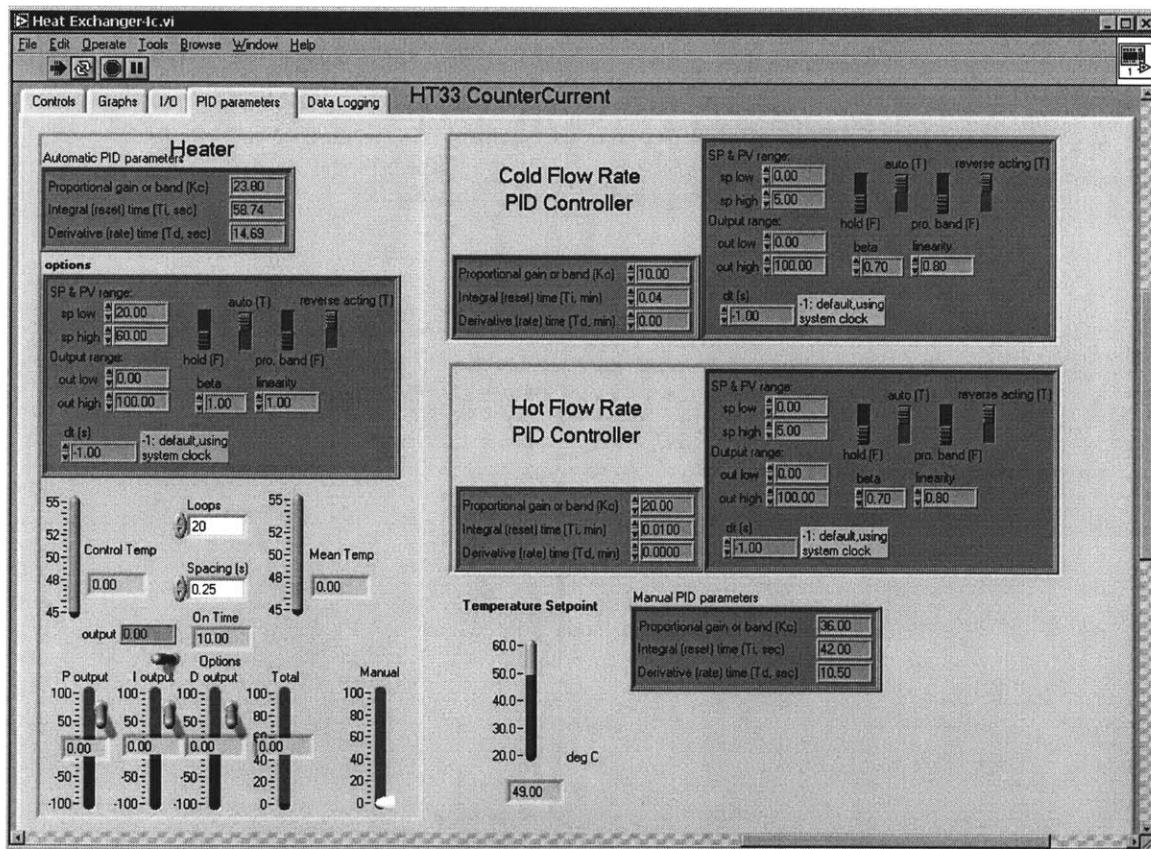


Figure 4-7: The PID parameters panel of the Front panel of the LabView control and monitoring software

4.2.2.5 Data Logging Panel

In this panel the Administrator can modify the path of the data file to be created. Also the number of decimal places that is going to be recorded in the data log file can be set. We can also set the time interval at which data points will be recorded. At the bottom of the panel, we can see what data are being recorded and their current values. We can modify this to add any number of data points to be recorded to the file.

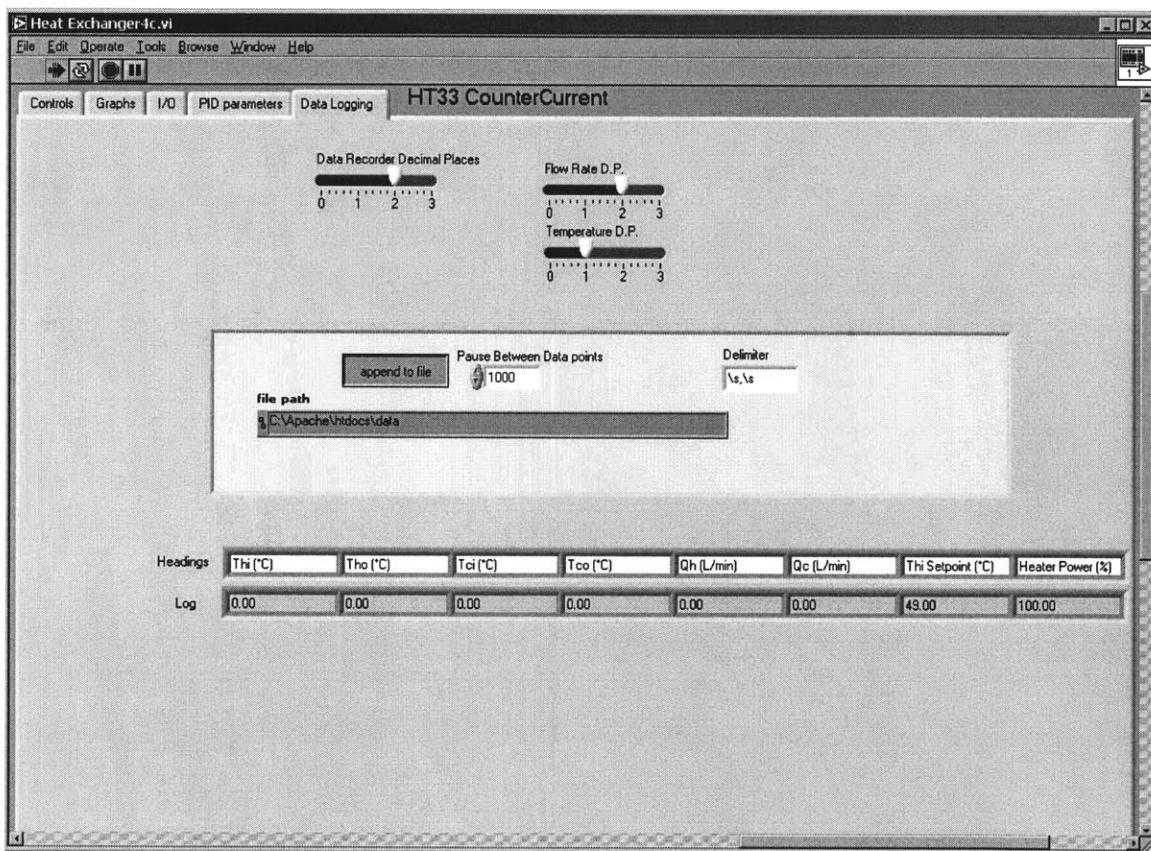


Figure 4-8 The Data Logging panel of the Front panel of the LabView control and monitoring software

4.3 Publishing Real-time data and getting Real-time input with DataSocket Server

4.3.1 DataSocket Server

DataSocket server is a software product developed by National Instruments that enables real-time sharing of data among a variety of client softwares developed in different programming languages [11].

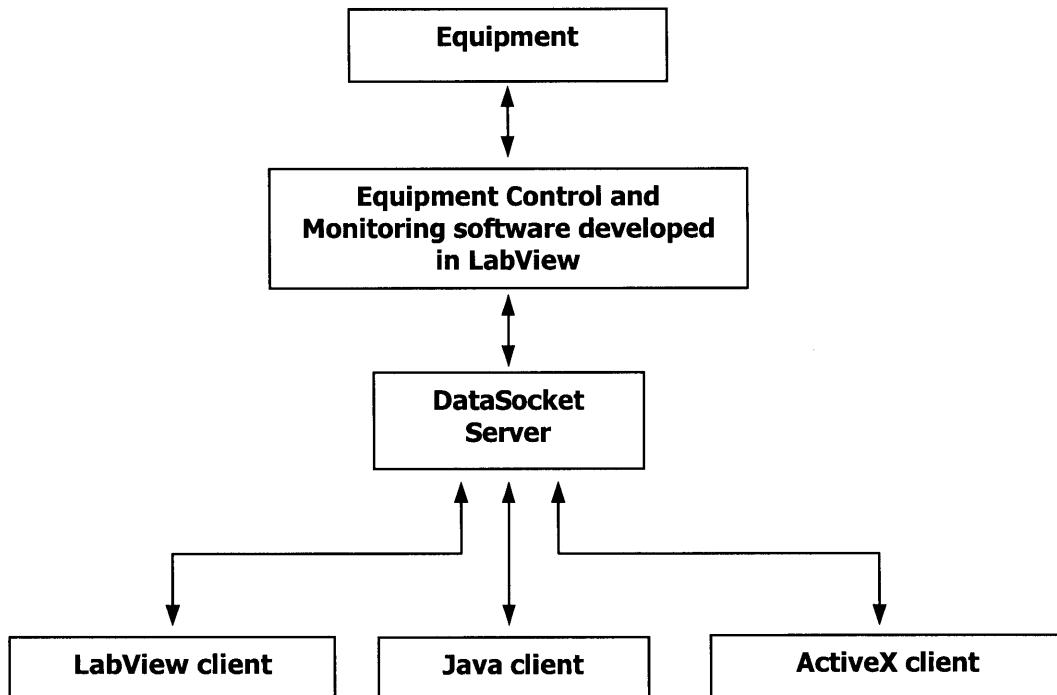


Figure 4-9: Publishing real-time data and getting real-time input from clients using DataSocket server

The DataSocket server uses a publish-subscribe model to share the data. The server publishes real-time data to the internet at a specific address. It uses a dstp protocol (Data Sockets Transfer Protocol) to publish the data. Each data point binds to a specific URI (with a prefix `dstp://`). The publisher (LabView software) sends data to this URI. The clients subscribe to the server using these specific data addresses and listen for updates. Upon update of the data value, the values at the data points also update and the clients get

an updated value of the data points. On the other hand, while the clients control the equipment, the client software publishes the parameter control values to the DataSocket server and the Lab view software listens for updates. The server and the API take care of making multiple connections and transferring the data.

Figure 4-9 shows the publish-subscribe model of the DataSocket server. The server side equipment control and monitoring software passes and gets the parameter values from the equipment and publishes the data at specific URIs using the DataSocket server (explained in details in the next section). This software also listens to some predefined URI for control parameter values. The clients can be a LabView application, a Java applet or a Visual basic application. These clients subscribe to the data using the DataSocket API. Currently, data socket API implementations exist for Java and ActiveX. The client, while controlling the equipment, publishes the control parameter values to the predefined URI through the data socket server. Thus the server-software gets the control parameter values from the client and passes these to the equipment at real-time.

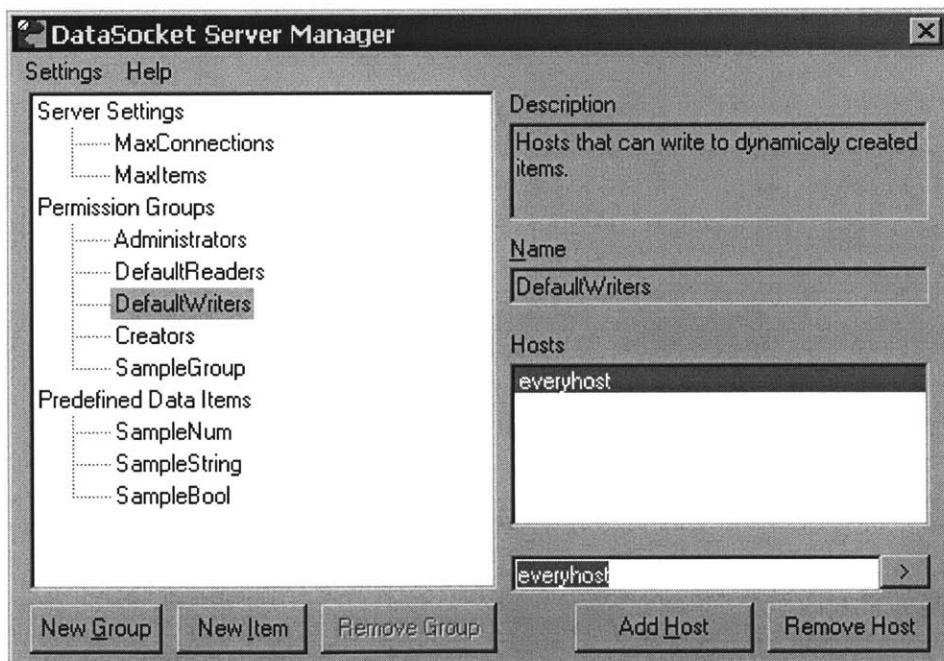


Figure 4-10: DataSocket server manager

The DataSocket server manager (Figure 4-10) provides a way of configuring client access parameters and data parameters. The DefaultReaders and DefaultWriters should be set to everyhost in order to publish the data to all clients and get input from all clients.

4.3.2 Publishing Real-time data and getting real-time input from LabView

Publishing data to the internet using the DataSocket server from LabView is made very simple in the latest version of LabView. Two functions, one to read data from the DataSocket server and the other to write data to DataSocket server, are provided.

The DataSocket Write VI is shown in figure 4-11. The data of any type needs to be wired to the data parameter in the VI and the URL parameter needs to be specified with a String constant. When this is done, data is being written in that specific URL.

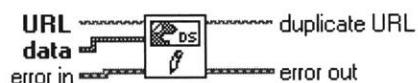


Figure 4-11: The VI used to publish data to the DataSocket server

The DataSocket Read VI is shown in Figure 4-12. The URL parameter needs to be specified with a String constant. This is the address from where the data is read. The data is read as variant by default. The type can also be specified with the type parameter. The VI can wait for updated value when the parameter wait for updated value is set to True. The ms timeout parameter specifies how long to wait for a value update.

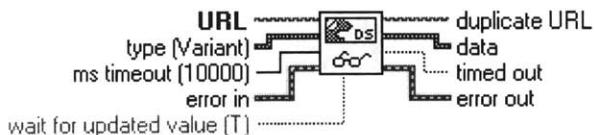


Figure 4-12: The VI used to get the data from the DataSocket server

The VI that converts the variant data type to any other type is shown in figure 4-13. The variant parameter needs to be wired to the data that we want to convert. The type parameter is the data type that we want to convert the variant data into. The data parameter is the resulting data type.



Figure 4-13: The VI used to convert data from Variant to other types

If the DataSocket server is running, it takes care of the communication between the subscriber and the publisher.

4.3.2 Subscribing to Real-time data and publishing real-time input from Java Applet

The Java Applet uses DataSocket Java Bean [10], developed by National Instruments, to connect to specific data URIs through DataSocket Server.

Figure 4-14 shows a sample code to connect to the DataSocket server from a Java Applet. The `DataSocket` class takes care of connecting to the DataSocket server in the same machine or in a remote machine. A Listener object, `DSOnStatusChangedListener` is attached to the `Data Socket` object in order to call the `dsInHflow_DSOnStatusChanged` method whenever the data value in the specific DataSocket `dsInHflow` changes. The specific tasks to be performed when the data value of the DataSocket changes, are implemented in the `dsInHflow_DSOnStatusChanged` method.

```
private DataSocket dsInHflow;
dsInHflow = new DataSocket();
dsInHflow.addDSOnStatusChangedListener(
    new DSOnStatusChangedListener() {
        public void DSOnStatusChanged(DSOnStatusChangedEventArgs e) {
            dsInHflow_DSOnStatusChanged(e);
        }
    }
);
```

Figure 4-14: Connecting to the DataSocket Server

Figure 4-15 shows a sample code that updates a data value in a DataSocket connection and published in the DataSocket Server.

Whenever the `updateHFlow` method is called from within the Applet, it captures the data from the corresponding `TextBox`, `inHFlow` and converts it into a `double` value. Then it publishes the data value to the DataSocket server by updating the corresponding data value in the `DataSocket` object `dsInHFlow` by calling its `getData()` method and `SetValue()` method.

```
public void updateHFlow()
{
    String hText = inHFlow.getText();
    try{
        double hValue = Double.parseDouble(hText);
        dsInHFlow.getData().SetValue(hValue);
    }catch(NumberFormatException ex){ }
}
```

Figure 4-15: Updating (publishing) data to DataSocket server

4.4 Remote Control and Monitoring using Java Applet

The client-side Java Applet was developed for remote Control and Monitoring of the Heat Exchanger equipment.

The first version, **Version 1.1** of the Java applet, shown in figure 4-16 was launched in November 2001, when the system was used in the course 10.302 “Transport Processes”. This version of the java applet had the minimum capability to monitor and control the heat exchanger. It had three panels

- Data Socket Connection Panel,
- Input Panel and
- Output Panel

The Server-side LabView software needs to be in the Client mode in order for the input panel of the java applet to work.

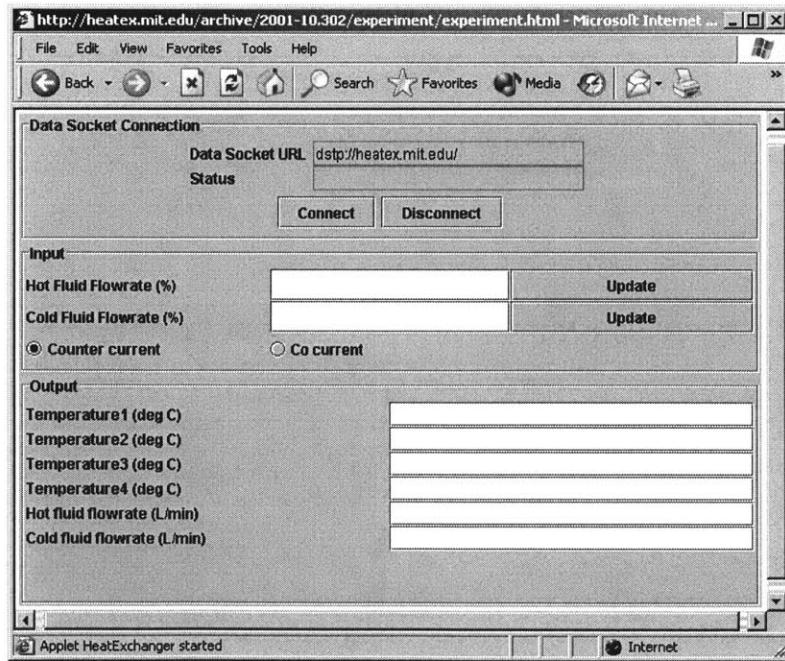


Figure 4-16: Version 1.1 of the Client-side Java Applet (November 2001)

After the applet loads on the client browser, the user presses the Connect button. The applet then creates the DataSocket connections associated with the input and output

parameters. Upon successful connection to the DataSocket server through the given URIs, the Status textbox shows that the applet is connected to the server and the current values of the temperature and flow rate data are shown in the Output Panel. If the users want to vary any of the flow rate values, they need to input the new value in the appropriate textbox and press the Update button. The Java Applet then publishes this value to the appropriate URI and the LabView software gets the value and inputs it to the equipment in real-time. The change can be seen in the appropriate flow rate output textbox in the Output panel. The communication of the Applet with the DataSocket server is made through the DataSocket Java Bean API, which is explained in details in the earlier section. When the users are done with the experiment, they have to press Disconnect button and close the browser window to completely log out of the system

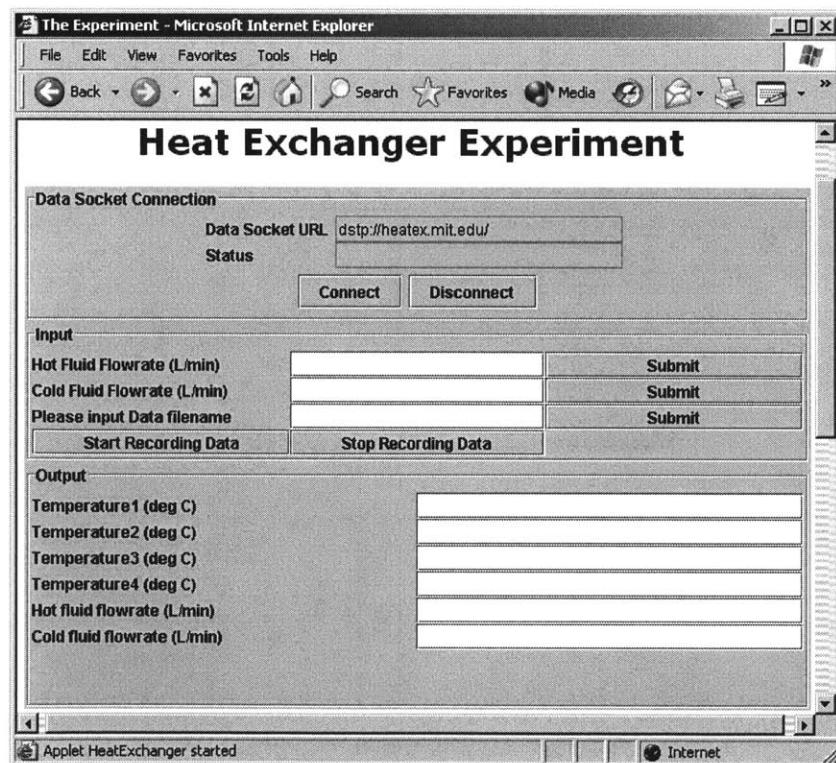


Figure 4-17: Version 2.22 of the Client-side Java Applet (February 2002)

The next version of the applet, **Version 2.22**, as shown in figure 4-17, was launched in February 2002, when the system was used in the course 10.26 “Chemical Engineering Projects Laboratory”. This version of the applet had the same capabilities as Version 1.1

with one added feature, which was Automatic Data Recording. This system had improved control of the hot water inlet temperature, which was done by implementing a PID algorithm on the Server-side LabView Software. Also, the Data Recording capability was added to the Server-side software. But the input to start recording and stop recording the data was sent from the Java Applet while in the system was in client mode. While doing the experiment, when the steady-state condition is reached and the client wants to start logging the data, he/she inputs the name of the file that is to be created to store the data and presses the Submit Button. The LabView software creates a file at a particular path with this filename. Then the user presses the Start Recording button to start recording data and then if he/she is done taking the data, presses Stop Recording button to stop recording the data. The user then retrieve the file from a certain location on the website. The detailed description of how to obtain the data is explained in the website <http://heatex.mit.edu>.

The next version of the Applet, **Version 3.2**, as shown in figure 4-18, was released in April 2002 for use in two courses, 10.450 “Process Dynamics, Operations and Control” and UT354 “Transport Processes”. This version of the applet has some more added features. A data recording indicator was added to the Data Recording panel. Whenever the system is recording data this indicator turns Green. And whenever the system stops recording data, it turns to gray again.

A graph showing real-time variation of the hot-water inlet temperature is added to the applet. This graph stores all the values in a vector and updates the chart in every 5-second interval.

The Parameter Controls panel was updated with some more controllable parameters. A new feature was added on the Server-side LabView software that allows the Administrator of the system to limit the number of controllable parameters from the server-side software. If any of these parameters are set to Not Controllable, then the buttons and text boxes corresponding to that parameter on the client applet becomes

grayed out. With the implementation of the PID algorithm, three new controllable parameters are added to this newer version of the applet.

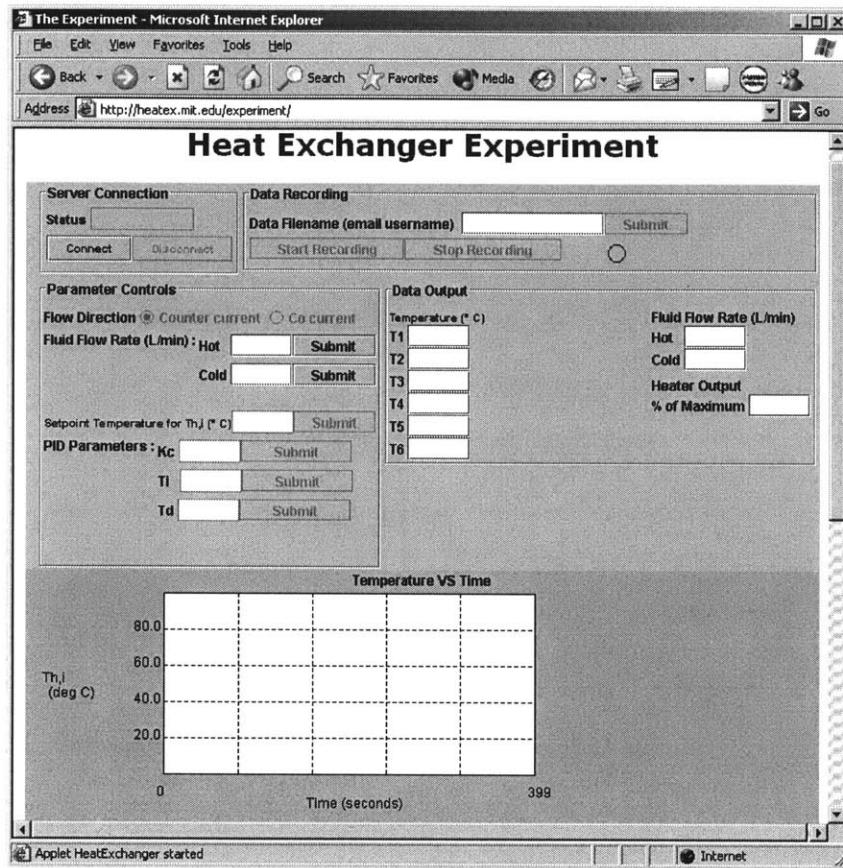


Figure 4-18: Version 3.2 of the Client-side Java Applet (April 2002)

The latest version of the applet, Version 4.1, as shown in figure 4-19, has client Authentication and Collaboration capability. Only the registered students will be able to perform experiment with this applet. The students need to register and sign up for a particular time-slot. This is described in details in the next chapter.

The collaboration component of this applet is described in details in Section 5.4.3.

After the applet loads, the user needs to input a valid set of LoginID and Password to gain access to the system. After authentication is complete, data outputs from the DataSocket server are shown in the Data Output Panel. The user sees a list of logged on users in the appropriate Text box in the Collaboration Panel. Also the name of the user who is in

control of the equipment is shown in the appropriate place in this panel. The user can share thoughts with other users by typing in a message, which is broadcast to other logged on users. The user can also request for control of the equipment by pressing the Request For Control button. If his/her request is granted, the user will be able to control the equipment and record the data.

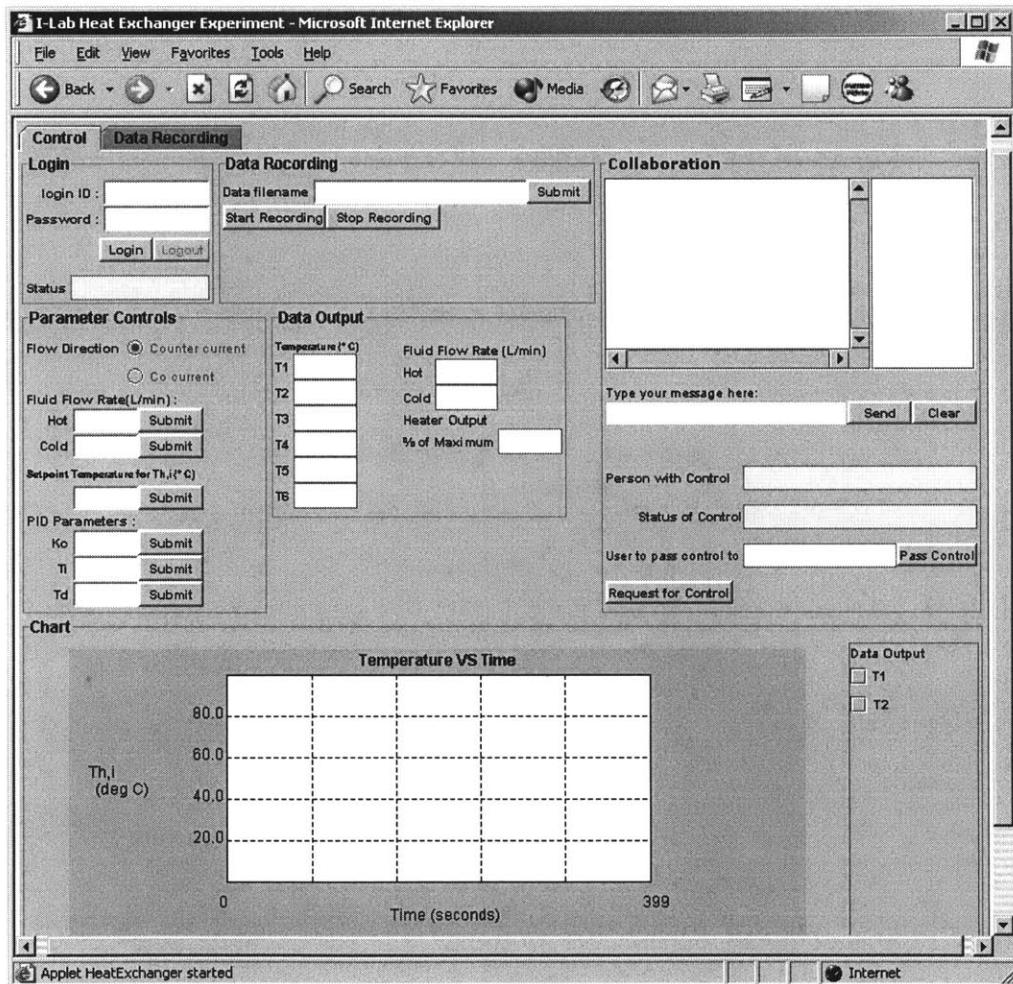


Figure 4-19: Version 4.1 of the Client-side Java Applet (June 2002)

There is an option for adding a table of output values from the equipment, which is going to be added in the next release.

Chapter 5. USER MANAGEMENT AND COLLABORATION

5.1 Introduction

This chapter discusses the various components of the user management and collaboration support for the system. User authentication, scheduling and collaboration are very important in courses involving high volume of students. We begin our discussion with the data model and the database used for this purpose. We then continue with the user registration, authentication and scheduling system. We finish our discussion with the details of the collaboration application.

5.2 Data Model and the Database

Microsoft SQL Server 2000 was used as the database. The Database server resided on the same machine as the Web Server without hampering the performance of either of them. Preferably, they should reside on separate machines.

5.2.1 The data model

The database `classlist` has two tables and 10 stored procedures to keep track of the user data and the experiment scheduling data.

The `Students` table stores all the information about the users of the system. This information can be input either from

- the user registration page (`registration.aspx`) or
- manually from the SQL server Enterprise Manager

`StudentID` is the primary key of this table and it is automatically generated **SQL int** type. The `Name`, `EmailAddress`, `Phone`, `LoginID` and `Password` all are **SQL varchar** type.

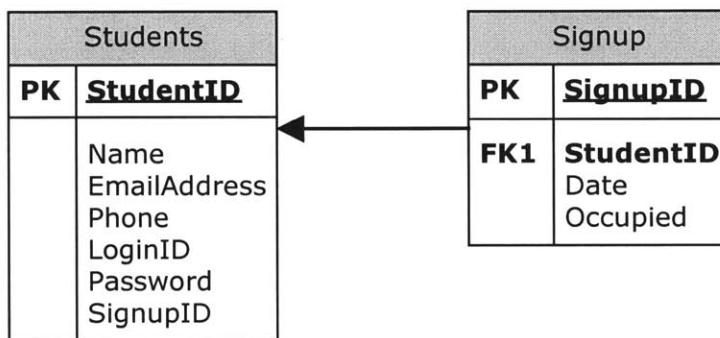


Figure 5-1: Database E-R diagram for the database

The `Signup` table stores scheduling data of a particular experiment. `SignupID` is the primary key of this table and `StudentID` is the foreign key, which relates to the `StudentID` of `Students` table. The `Date` field is of **SQL varchar** type and contains information about schedule times and dates. This data is input manually from the Enterprise Manager, but later can be modified to be entered from the website. The `Occupied` field is of type **SQL bit** and contains either 1 or 0 depending upon whether the particular scheduling time is occupied or not.

5.2.2 Stored Procedures

In the User Management system, stored procedures are used in order to get a better performance in data access and manipulation. 10 stored procedures were developed using Microsoft Visual Studio.NET Enterprise Architect and they were tested and debugged to

ensure reliability and better performance before actually deploying. The stored procedures and their functions are tabulated below.

Table 5-1: List of Stored Procedures, their purposes and use

Stored Procedure	Description and Purpose	Used by
addStudents	Adds new user information to the Students database	DBAccess.cs Register.aspx
addSignup	Signs up a user for the particular signup time	DBAccess.cs Signup.aspx.cs
checkLogin	Authenticates a user in the Login page	DBAccess.cs Login.aspx.cs
CheckSignupIDStudentID	Checks whether the particular user is signed up for that particular signup time	DBAccess.cs Remove.aspx.cs
deleteStudentSchedule	Removes the user from the particular signup time	DBAccess.cs Remove.aspx.cs
getEmptySchedule	Gets the unoccupied signup times	DBAccess.cs
getSchedule	Gets the signup times and the corresponding names and email addresses	Schedule.aspx.cs
getScheduleIDStudentSchedule	Gets signupID and signup times for a particular StudentID	Remove.aspx.cs
getStudentName	Gets the student name of a particular StudentID	DBAccess.cs
getStudentSchedule	Gets signup information of a particular StudentID	MySchedule.aspx.cs

These will be explained in details later in the chapter with appropriate ASPX pages that use these stored procedures.

5.3 User Management System

The user management system is developed in ASP.NET using C# as the language platform. The system is hosted by IIS 5.0 (Internet Information Services) web server and resides on the same machine as the website.

The System is divided into two parts

- Database access
- .aspx pages (forms and tables)
 - User Registration, Authentication and Session Management
 - Scheduling

5.3.1 Database access using ADO.NET

The database access for the user registration and scheduling system is provided using ADO.NET and is separated from the presentation layer (.aspx pages).

All the data access and manipulation capabilities are provided in the `DBAccess.cs` file. The methods for data access and manipulation are made `static` so that this class doesn't need to get instantiated. We get a better performance by getting rid of the numerous objects created in such a way. Another important feature is that in every method, the connection to the database is created, the data operations are carried out and then the connection is terminated. This adds an overhead of creating and destroying the connection at every function call of the `DBAccess.cs` class, but on the other hand, we get rid of the numerous live connection created by numerous users at the same time. Thus reducing the load on the SQL server.

In Figure 5-2 a sample database access using ADO.NET is shown. The key components are `SqlConnection`, `SqlCommand` and `SqlDataReader`. At the beginning, a connection is established with the SQL server by specifying the connection parameters as `ConnectionString` attribute of `myConn` object of type `SqlConnection`. Then a `SqlCommand` type object of name `checkLoginCMD` is instantiated and its `CommandType` is set to `StoredProcedure`. This means that in this connection, we are going to execute a Stored Procedure on the database and get the result back from the database. We then add parameter values to the `checkLoginCMD` object that are to be passed to the corresponding stored procedure.

```

public static int login(string ID, string Password)
{
    SqlConnection myConn = new SqlConnection();
    myConn.ConnectionString = "data source=local;initial catalog=classlist;persist
    security info=False;user id=" +
    ";password=;workstation id=;packet size=4096";

    SqlCommand checkLoginCMD = new SqlCommand("checkLogin", myConn);
    checkLoginCMD.CommandType = CommandType.StoredProcedure;
    checkLoginCMD.Parameters.Add("@LoginID", SqlDbType.VarChar, 50).Value = new
    SqlString(ID);
    checkLoginCMD.Parameters.Add("@Password", SqlDbType.VarChar, 50).Value = new
    SqlString(Password);
    myConn.Open();

    SqlDataReader myReader = checkLoginCMD.ExecuteReader();

    int studentID = 0;
    while (myReader.Read())
    {
        studentID = myReader.GetInt32(0);
    }
    myReader.Close();
    myConn.Close();
    return studentID;
}

```

Figure 5-2: Sample code for database access

Then we open the connection `myConn` and execute the command `checkLoginCMD`. The result is put into `myReader`, which is an instance of `SqlDataReader`. By browsing through `myReader` (`myReader.Read()`), we get all the values from the execution of the stored procedure in the database.

At the end of the process, we need to close the `SqlDataReader` `myReader` and also the connection to the database, `myConn`.

5.3.2 User Registration, Authentication and Session management

The user Registration and Authentication system is developed using ASP.NET. The presentation HTML forms pages have an extension .aspx and the code files associated to these forms pages have an extension .aspx.cs. The user registration and authentication system uses the classlist database discussed earlier and the database access is made through the DBAccess.cs file discussed earlier.

User Registration:

Figure 5-3 shows the initial Login page for the user authentication system. New students will follow the Register link and they will be redirected to the user registration page. The details of the user authentication and session management are described after the user registration overview.

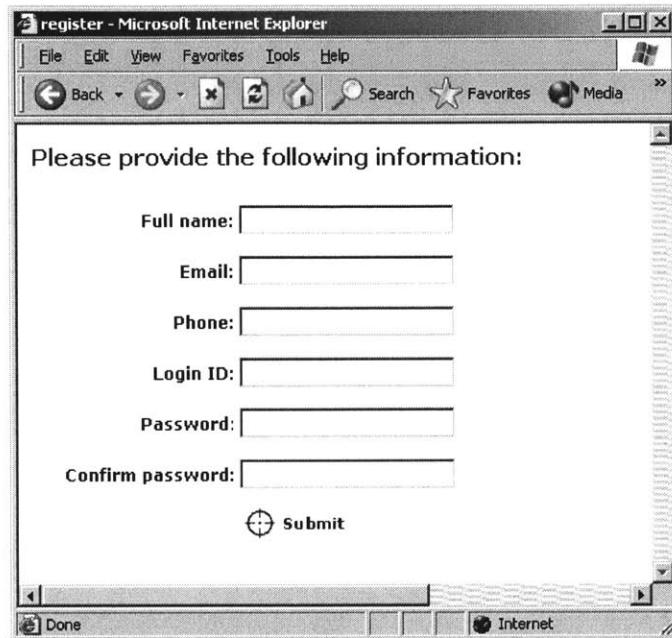


Figure 5-3: User authentication login page.

Figure 5.4 shows a blank user registration page that the users are greeted with after clicking the Register hyperlink. All the fields are required in this form page. In order to

check the validity of the form, web form validation controls were used. Two types of validations controls were used

- **RequiredFieldValidator:** In order to make sure that no field is empty.
- **CompareValidator:** In order to make sure that the password fields match.

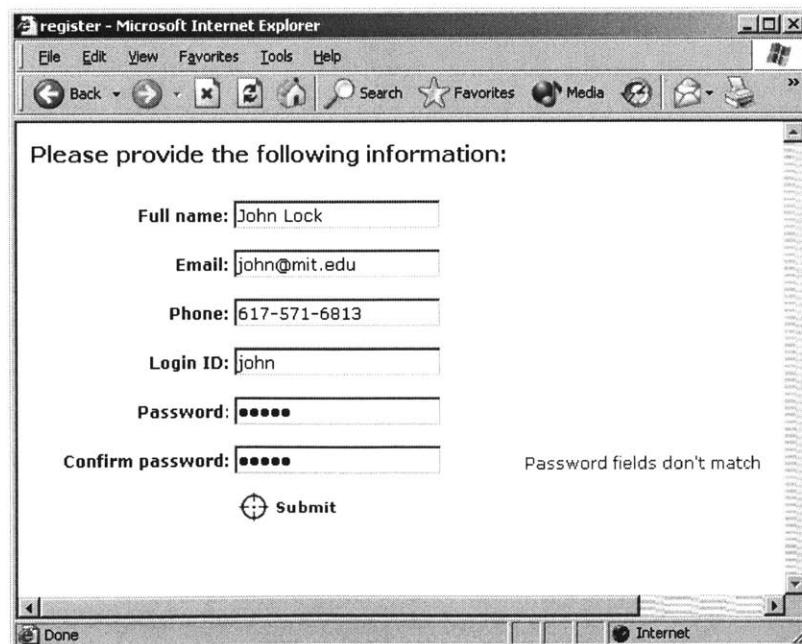


A screenshot of a Microsoft Internet Explorer window titled "register - Microsoft Internet Explorer". The page displays a registration form with the following fields:

- Full name: [empty input field]
- Email: [empty input field]
- Phone: [empty input field]
- Login ID: [empty input field]
- Password: [empty input field]
- Confirm password: [empty input field]

A "Submit" button is located below the fields. The status bar at the bottom shows "Done" and "Internet".

Figure 5-4: Blank user registration page



A screenshot of a Microsoft Internet Explorer window titled "register - Microsoft Internet Explorer". The page displays a registration form with the following data:

- Full name: John Lock
- Email: john@mit.edu
- Phone: 617-571-6813
- Login ID: john
- Password: *****
- Confirm password: *****

A message "Password fields don't match" is displayed next to the "Confirm password" field. A "Submit" button is located below the fields. The status bar at the bottom shows "Done" and "Internet".

Figure 5-5: User registration page filled with sample data and showing data validation

After checking the validity of the user inputs, the system registers the user in the database by calling the static register() method of the DBAccess class. Figure 5-6 shows the construction of the stored procedure to add new user to the table Students. The register() method transforms the form input values to corresponding SqlDbType and passes the values to dbo.addStudents. The stored procedure takes the parameter values and executes a SQL query to insert the values into the database.

```
ALTER PROCEDURE dbo.addStudents
(
    @Name varchar(50),
    @Email varchar(50),
    @Phone varchar(50),
    @LoginID varchar(50),
    @Password varchar(50)
)
AS
/* SET NOCOUNT ON */
INSERT INTO Students(Name, EmailAddress, Phone, LoginID,
Password)
VALUES (@Name, @Email, @Phone, @LoginID, @Password)

RETURN
```

Figure 5-6: Stored procedure in the database to add new users

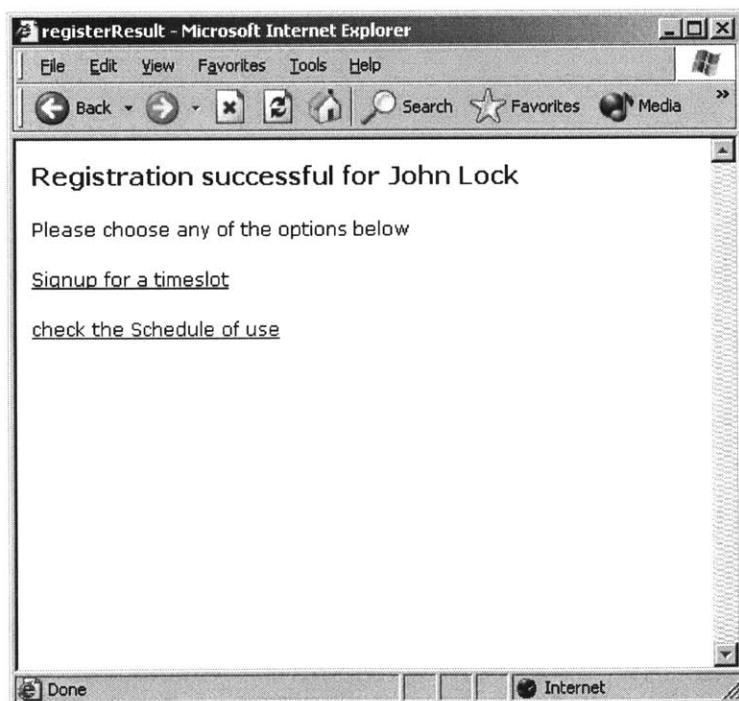


Figure 5-7: Registration confirmation page.

On successful completion of user registration, a confirmation page is shown with the option for checking the schedule of use for all the time slots and also signing up for a time slot.

User Authentication:

Already registered students can manage their scheduling like signing up for an available timeslot, reschedule their experiment, view their signup times and perform experiments, all without any intervention by the system administrator. The user Authentication Login page takes user inputs and authenticates the user as a valid user of the system. This page also use form validation control to check that neither of the fields are left empty.

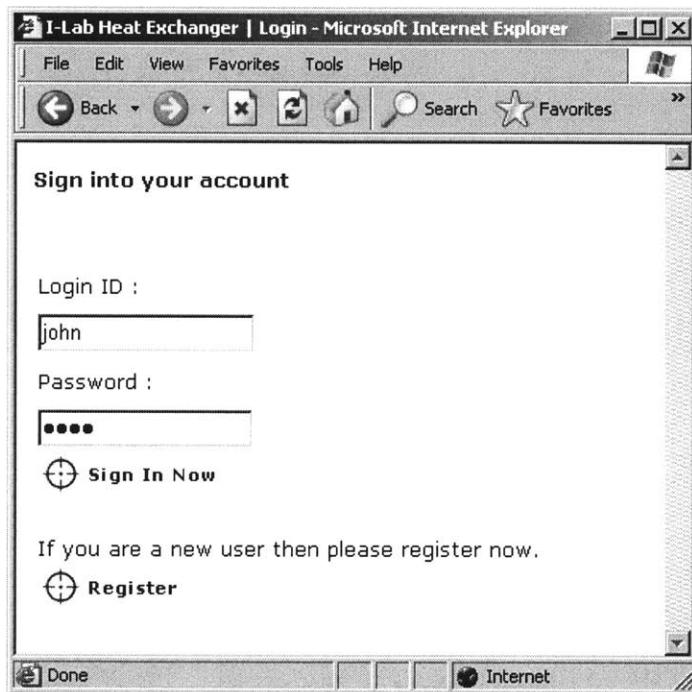


Figure 5-8: User authentication login page (for already registered users)

After the form is validated, the program calls the `login()` method to authenticate the user inputs. If authenticated, the user's user ID is returned.

Session Management:

Once the user is authenticated and a valid user ID corresponding to the logged in user is returned, the system sets a cookie to the user's computer and sets the cookie value to this user ID. The system keeps track of the user session by tracking this cookie.

For this session management to work, the browser on the user's computer should be set to accept cookies.

In Internet Explorer, this can be achieved by accessing *Tools>Internet Options..* then choosing the *Privacy* tab and set the *Settings* slider to *Accept All Cookies*.

In Netscape Navigator/Communicator, this can be achieved by accessing *Edit>Preferences...then in the Advanced Category, set the Cookies value to Accept all cookies.*

Now that the cookie is set, every time the user makes a http request to this website, the system identifies the user and sends corresponding information. Figure 5-9 shows the default page from where the user can manage his/her schedule, which is described in details in the next section. Once the browser is closed, the cookie becomes invalid and the user has to sign in again to manage the schedule or perform the experiment.

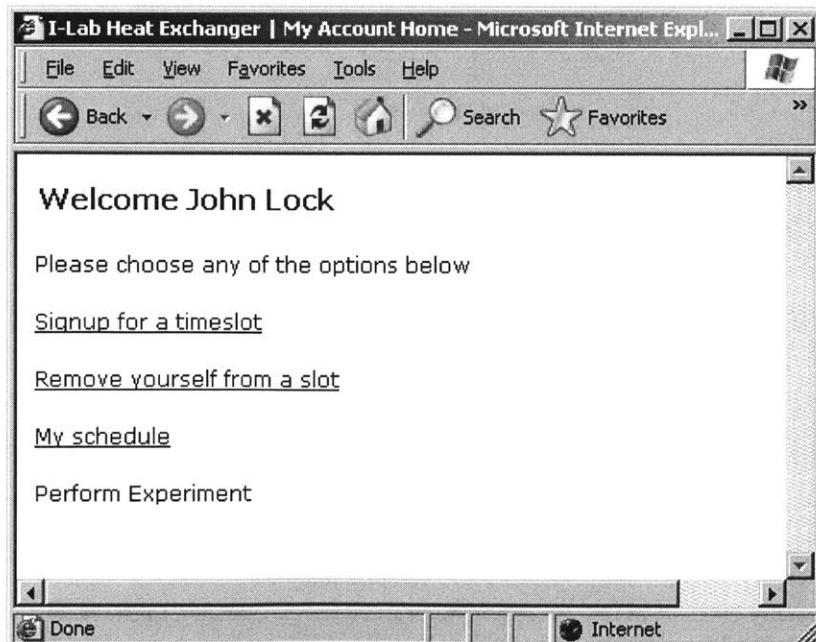


Figure 5-9: The default schedule management page.

5.3.3 Scheduling

The scheduling system consists of the signup pages, rescheduling and experiment access page.

Figure 5-10 shows the signup page. The dropdown list consists of all the available time slots. This list is generated from the database table Signup where the Occupied field has a value 0 (not occupied). Figure 5-11 shows the stored procedure that is used to do achieve this.

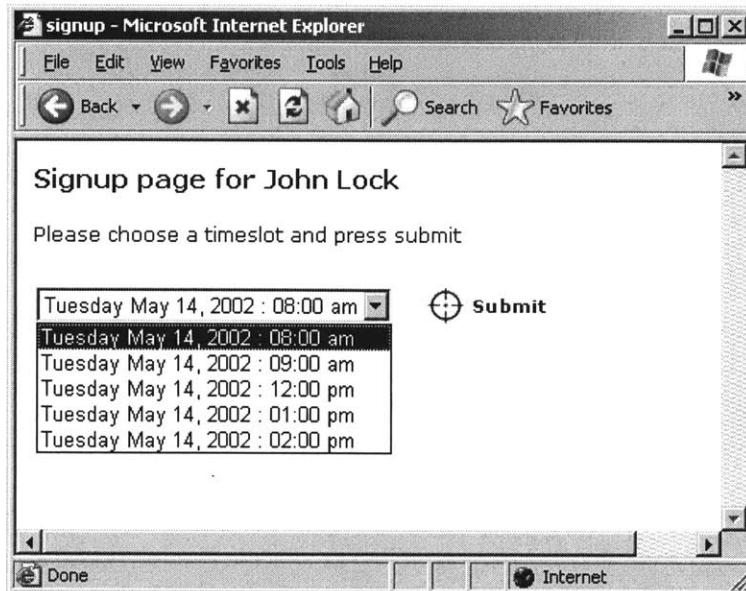


Figure 5-10: Signing up for an available timeslot

```
ALTER PROCEDURE dbo.getEmptySchedule
AS
    /* SET NOCOUNT ON */
    SELECT Date
    FROM Signup
    WHERE (Occupied = 0)
    ORDER BY SignupID

    RETURN
```

Figure 5-11: Stored procedure to get available time slots.

Once the user chooses one of the available time slots and clicks submit, the system associates the user information to that specific time slot and marks that time as

unavailable. So, the next time this time slot is not available for signup. The stored procedure used for this purpose is shown in figure 5-12

```
ALTER PROCEDURE dbo.addSignup
(
    @Date varchar(50),
    @StudentID int
)
AS
/* SET NOCOUNT ON */
UPDATE     Signup
SET         StudentID = @StudentID, Occupied = 1
WHERE       (Date = @Date)
RETURN
```

Figure 5-12: Stored procedure for signing up a user for a timeslot

Upon successful sign up, a confirmation page is shown to the user with some other available choices, which is shown in figure 5-13.

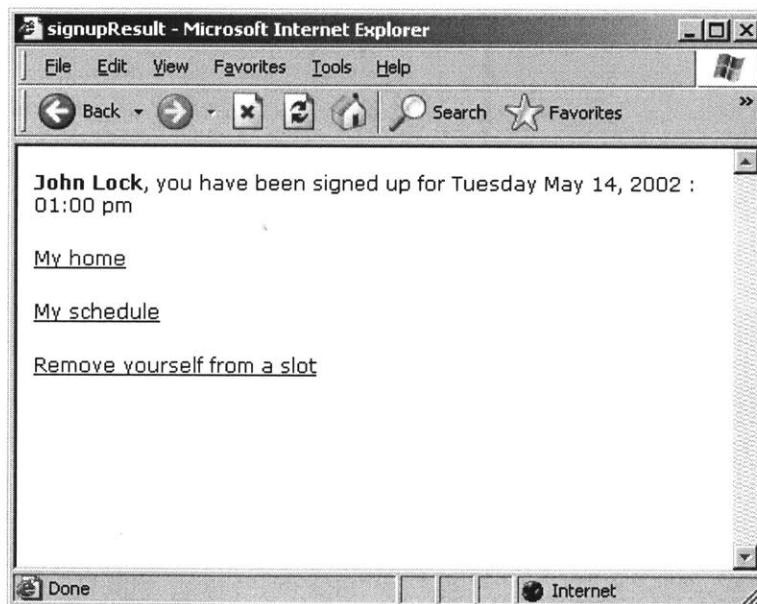


Figure 5-13: Signup confirmation page

Users can check the times that they are signed up for (figure 5-14). In this system, a user can sign up for multiple time slots. Later on, depending on the need of any specific course, this can be modified to restrict the users to only one time slot if necessary.

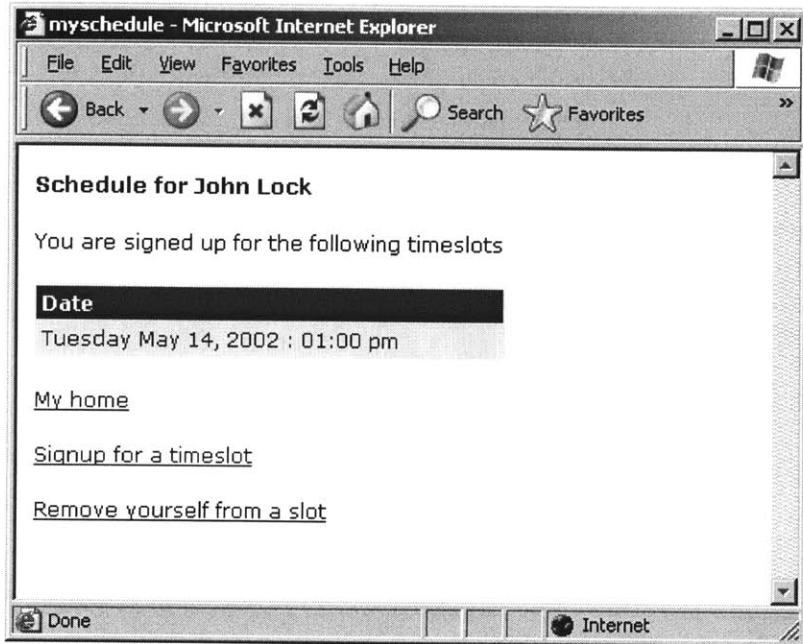


Figure 5-14: Times that the user has signed up for

The users can also check the status of all the timeslots available for the experiment. This is shown in figure 5-15.

Date	Name	EmailAddress
Tuesday May 14, 2002 : 08:00 am		
Tuesday May 14, 2002 : 09:00 am		
Tuesday May 14, 2002 : 10:00 am	Rubaiyat Amin Khan	rakhan@mit.edu
Tuesday May 14, 2002 : 11:00 am	Bayan Shafiq	bayan@yahoo.com
Tuesday May 14, 2002 : 12:00 pm		
Tuesday May 14, 2002 : 01:00 pm	John Lock	john@mit.edu
Tuesday May 14, 2002 : 02:00 pm		

Figure 5-15: The page showing the schedule of use for all the time slots

Users can also reschedule their signup. If they decide to do so, they can remove themselves from a time slot by clicking on the *Remove yourself from a slot*. They will be redirected to `remove.aspx` as shown in figure 5-16 and a list of timeslots will be available for which they have signed up for. They can choose any of these time slots and remove themselves from it. The system checks the validity of the `SignupID` to be removed by checking whether or not the user is actually signed up for that particular time slot. If an invalid `SignupID` is chosen, the system shows an error message and asks the user to enter a valid `SignupID` to remove. When a valid `SignupID` is chosen the system disassociates the user from that particular time slot and mark that slot as available (shown in figure 5-17), so that next time when another user wants to sign up for a time slot, this time slot will be available to him/her.

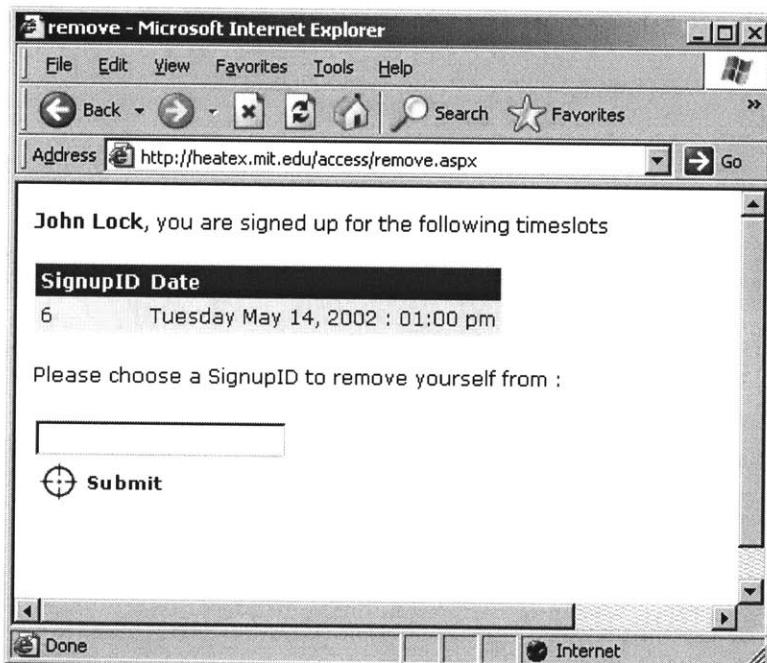


Figure 5-16: Rescheduling by removing user from a time slot

```
ALTER PROCEDURE dbo.deleteStudentSchedule
(
    @SignupID int
)
AS
/* SET NOCOUNT ON */
UPDATE     Signup
SET          StudentID = 1, Occupied = 0
WHERE        (SignupID = @SignupID)
RETURN
```

Figure 5-17: Stored procedure to remove user from a time slot

After successful removal of the user from the time slot, the user is taken to the removal confirmation page, which is shown in figure 5-18. The user is provided with options for signing up for other available time slots, view his/her schedule or go to the scheduling home page

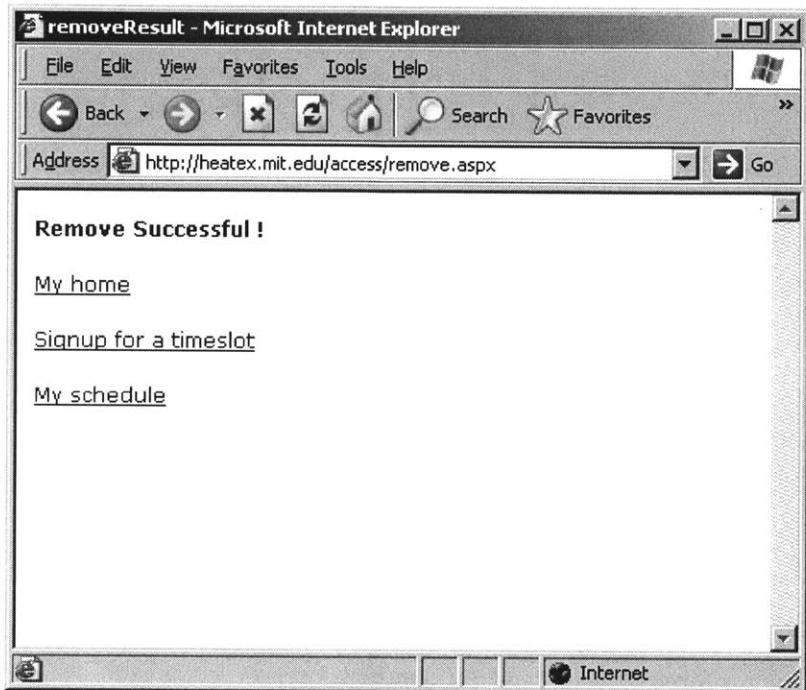


Figure 5-18: Confirmation of removal of user signup

5.4 Java Collaboration System

5.4.1 Introduction

The Collaboration capability is a key component of the system. It allows multiple users to access the Client side experimentation applet and chat among themselves and do the experiment while only one person can change the equipment input parameters. The control of the equipment can also be passed from one user to another.

Due to the nature of the Heat Exchanger equipment, only one experiment can be run at one time. So, in earlier versions of the system, only one user could log into the website and do the experiment. But with the collaboration capability, a team consisting multiple users can log into the website at the same time and perform the experiment.

The Collaboration System employs the client/server concept. The Server-side Collaboration Application manages the clients that are logged in, handles the authentication and database access through JDBC and manages the passing of control of the equipment. The Client-side Collaboration component allows users to control the equipment by checking the status of control, allows the users to chat among themselves and keeps track of the status of the equipment control.

5.4.2 Collaboration Server-side Application

5.4.2.1 Client connection and management

The Server-side Application uses threads to perform multiple tasks. Within one thread, it creates an instance of `java.net.ServerSocket` class and binds it to a specific server port. It then keeps listening to this port for clients without interrupting the main thread. Figure 5-19 shows a screenshot of the server application. When the **Start Server** button is pressed, the `ServerSocket` is created and the server starts listening to this socket.

If a client connects to the server at that specific port, the server reads from the `InputStream` of the Socket and tries to authenticate the user. Client Authentication is described in detail in Section 5.4.2.2.

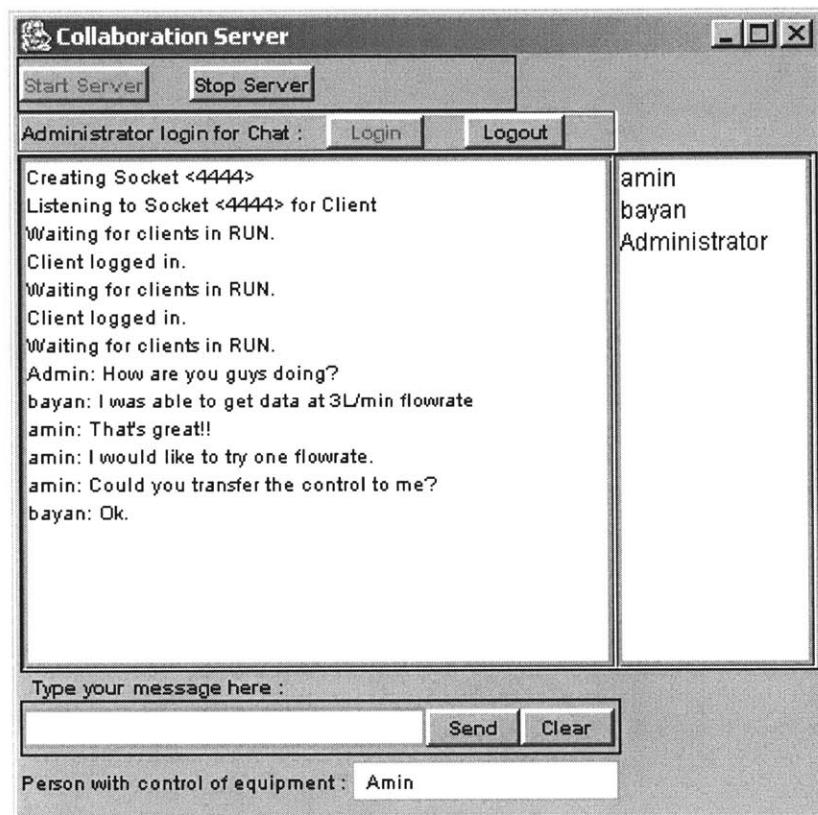


Figure 5-19: Collaboration server-side Application

If the client is not authenticated, a denial message is sent to the client applet stating the reason of denial. If the client is authenticated, a new instance of the `Client` class is created and added to the list of clients, which is kept in a `vector` named `clients`. After the new client is created the server keeps on listening to the same port for new clients to log in.

The `Client` class implements the `java.lang.Runnable` Interface and it listens for any new message from the client applet within its own thread without disrupting either the server functions in the main thread or the functions of its own methods. Message composition, parsing and transfer is described in details in Section 5.4.2.3.

5.4.2.2 Authentication

Client authentication is done by connecting to the SQL server `classlist` database, which is explained in Section 5.2.1. JDBC is used to connect to the SQL server database. The JDBC API [13] provides Java programs access to tabular data sources in a large number of formats. While the core API provides a high-level, platform independent interface, it relies on platform and data format specific implementation/driver to actually access and manipulate the data.

For accessing the database, the Microsoft SQL server 2000 driver for JDBC [8], provided by Microsoft Corporation, was used. All the database access code is provided in class `dbAccess`. Figure 5-20 shows sample database access code using JDBC.

First, the driver needs to be *Registered*. Registering the driver tells the JDBC driver manager which driver to load. `Class.forName()` registers the SQL server 2000 driver.

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");

Connection Con=DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433",
"loginID", "password");

CallableStatement myCS = con.prepareCall("{call checkLogin(?,?)}");

myCS.setString(1, loginID);

myCS.setString(2, password);

ResultSet myRS = myCS.executeQuery();
```

Figure 5-20: Database access code

An instance of the `Connection` interface establishes a connection to the SQL server residing on the same machine, by providing the correct login information.

The `CallableStatement` interface is used to execute SQL stored procedures. The JDBC API provides a stored procedure escape syntax that allows stored procedures to be called in a standard ways for all RDBMSs. The name of the stored procedure is `checkLogin`

and `(?,?)` means that it needs two input parameters. The parameters are passed by the `set` methods of the `CallableStatement`. There are a number of `set` methods available for passing parameters of different data types. Here, since the data types are `SQL.VARCHAR` type, we used the method `setString()`.

A `CallableStatement` can return one or more `ResultSet` objects. `myCS.executeQuery()` returns a `ResultSet` and it is stored in `myRS`. We can access the output of the stored procedure execution by browsing through the `ResultSet` object, `myRS`.

5.4.2.3 Message composition and parsing

The messages that are passed between the server and the client have specific predefined format. The messages consist of

- the purpose of the message: the command character,
- information about the sender and
- the actual message.

In order to distinguish among messages, a command character is inserted, which serves as the purpose of the message. Message composition is done using String concatenation.

A typical message will have the following format:

Command Character + ClientID + Body of Message

When a message is received at the server socket, the message is parsed using Java String manipulation methods. First, the message is divided into several substrings. Then, the message is processed according to the Command Character that it contains. For example, if the Command Character is “Login”, then the database access method to check the login is executed. Some other key Command Characters are “Logout”, “Message”, “SeekControl”, “ProvideControl”.

At the bottom of the graphical interface, the userID of the user with equipment control is shown. The Administrator has to press the Login button to take part in the chat. The chat among users and also transfer of equipment control is discussed in the next section.

5.4.3 Collaboration client component

In the collaboration window, there are two distinct parts. One is the chat window and the list of other users logged in, the other is the equipment control status window.

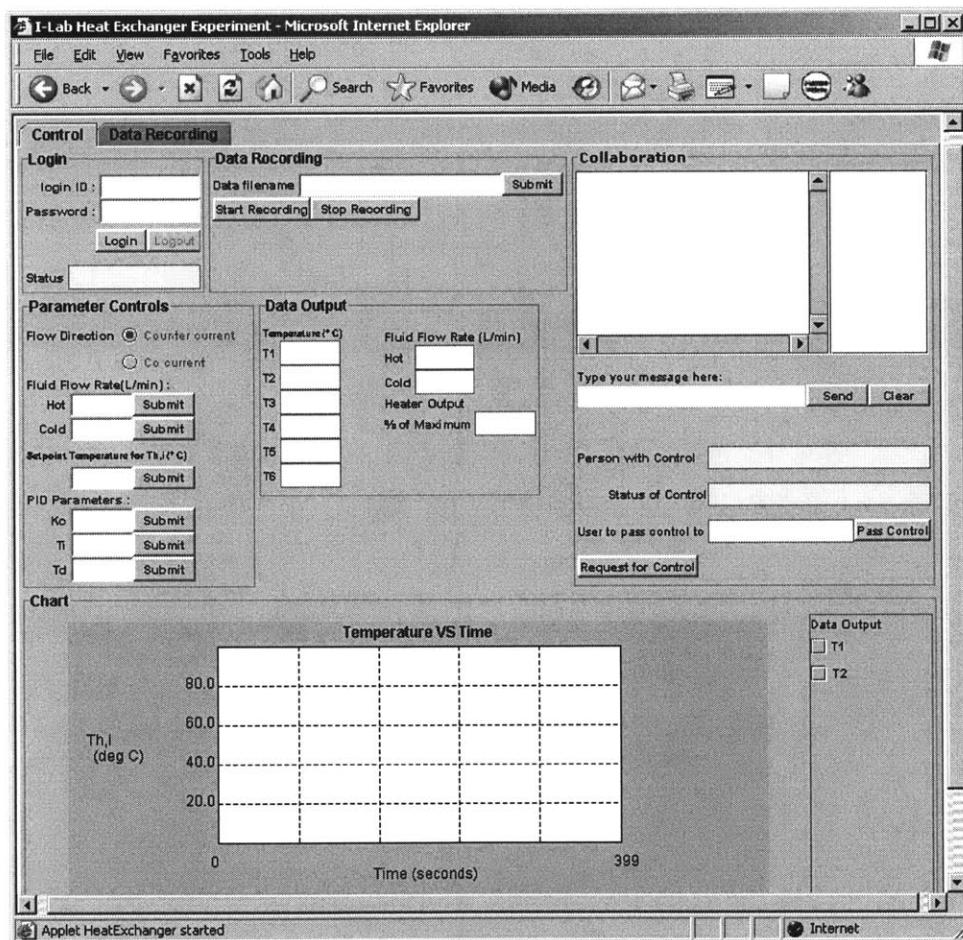


Figure 5-21: Client Applet: Equipment control and Collaboration

Figure 5-21 shows a screen shot of the Equipment control and collaboration applet. While doing the experiment, after the applet has loaded, the user has to input the loginID and

Password that he/she chose during the user registration process. When the “Login” button is pressed, the client applet creates a Socket connection and composes a Login message and passes it to the server through the `outputStream` of the `Socket`. The format of this message is already described in Section 5.4.2.3. If the login is successful, the user is able to see all the Data Output in the respective `TextFields`. The user can also collaborate with other users who are already logged in.

The first user to log in to the system gets the control of the equipment. The other users can request for control and all the requests are shown in the chat window. The user with the control can either approve the transfer of control or deny the request. If the user approves the request, then all the parameter control buttons become disabled and grayed out. But the text boxes show the current parameter values passed by the other user. At the same time the users can continue to chat in the chat window.

The Data Output window and Chart window is not affected by the Control of the user over the equipment. The users who are not controlling the equipment will be able to view the Data outputs and the Charts, but will not be able to record the Data. Only the user with control will be able to record data.

In order to get access to the complete code, please email the Administrator of the site <http://heatex.mit.edu>.

Chapter 6. STUDENT ASSESSMENT

6.1 Introduction

It is critical that we reevaluate and modify design/features of the system at every cycle of development process. Keeping this in mind, a set of assessment questionnaire was designed in order to get student feedback. With changing requirements and feature upgrades at different stages, the questions were modified accordingly and new questions were added, but the main structure of the assessment remained the same. This chapter explains the student responses to this assessment and summarizes the results.

6.2 Objectives of the Assessment

The concept of web-accessible virtual laboratories is still in its infant stage, with a lot of potential. But the framework should be designed and modified based on user needs. If these labs fail to serve its users well, then the educational objectives behind designing these labs are not attained. The key users of the system are the students in the courses in which these labs are being used. Keeping this in mind, the assessment questionnaires were designed to get maximum student input based on their experience in using the system.

In brief, the objectives behind designing the assessment are:

- Get student feedback on
 - Usability of the software
 - Desirable improvements
 - Whether the experiment meets educational objectives
 - Overall experience regarding the web-accessible experiment
- Improve and modify the system according to student needs
- Add desirable features to the system.

6.3 Student responses

It is important to tabulate and summarize the assessment data in a more conceivable format. So, after response data were collected, they were tabulated by respondent group; all responses were anonymous and confidential.

There are questions in four broad categories.

- Usability of the experiment: Responses in this category helped us identify the problems associated with the system that need modification and improvement.
- Improvements the students would like to see: Responses in this category helped us design new features and integrate them to the system.
- Meeting Educational Objectives: Responses in this category helped us judge whether the Heat Exchanger experiment met its Educational goals
- Overall Experience: Responses in this category gave us a broad idea about how the students felt about the Heat Exchanger Experiment.

6.3.1 MIT Course 10.302 Transport Processes: November 2001

A total of 66 students took the course and among them, 52 students filled out the assessment questionnaire. The Heat Exchanger Experiment was an extended homework problem. The responses were segregated between two groups

- Students who had problems controlling the flow rate
- Students who had no problems controlling the flow rate

Table 6-1: Student response : 10.302 : Extended Homework

	7=Strongly Agree 1=Strongly Disagree			
	(Mean \pm SD)		Range	
	Problems controlling flow rate * N=15	All others N=37	Problems controlling flow rate * N=15	All others N=37
Usability when carrying out the experiment on the web				
1. The instructions were clear.	4.5 \pm 1.8	5.7 \pm 0.9	1-7	3-7
2. I had no problems operating the experiment on the web.	4.3 \pm 2.0	5.6 \pm 1.2	1-7	3-7
3. I was able to control flow rates during the experiment.	3.3 \pm 2.1	5.7 \pm 1.1	1-7	3-7
4. I was able to make steady state temperature measurements	3.6 \pm 1.8	4.8 \pm 1.6	1-6	1-7
5. I was able to obtain the data needed for the assignment.	5.1 \pm 1.5	5.9 \pm 1.1	3-7	3-7
6. It actually took me about _____ minutes to carry out the experiment.	38 \pm 10	35 \pm 10	20-60	10-50
Improvements you would like to see on the website				
7. Other means for specifying flow rates (such as _____)	4.9 \pm 1.5	4.7 \pm 2.3	2-7	1-7
8. Graph of measurements versus time	5.5 \pm 2.2	6.0 \pm 1.3	1-7	2-7
9. Table of measurements versus time	6.6 \pm 0.7	6.2 \pm 1.1	5-7	3-7
10. Control over how data is averaged and reported	5.6 \pm 2.0	6.2 \pm 1.2	1-7	1-7
11. Physical properties of water	4.8 \pm 2.0	4.9 \pm 2.0	2-7	1-7
12. Basic equations of heat exchange	4.7 \pm 2.3	4.7 \pm 2.1	1-7	1-7
Meeting Educational Objectives				
13. The remotely controlled experiment provided an experience of measurement of flow rates and temperatures in a real system analysis of real data consideration of measurement errors	4.7 \pm 2.3 5.2 \pm 1.3 4.5 \pm 1.9	5.4 \pm 1.3 6.3 \pm 1.0 5.6 \pm 1.4	1-7 2-7 1-7	2-7 3-7 2-7
14. The assignment provided a vehicle for learning and reviewing thermodynamic limitations on performance, energy balance rate equations for heat exchange calculating overall heat transfer coefficients, correlations	4.4 \pm 1.4 4.6 \pm 1.6 4.7 \pm 1.9	5.6 \pm 1.3 5.7 \pm 1.2 5.6 \pm 1.3	2-6 2-7 1-7	2-7 2-7 2-7
Overall Experience				
15. The I-Lab heat exchanger experiment was fun.	2.9 \pm 1.7	4.5 \pm 1.5	1-7	1-7

This distinction was made based on the student response to Question 3. The students who had problems tend to give lower rating to all the questions. They spent more time

collecting the data (Q6), but it seemed that they were able to collect the necessary data (Q5). This group of students gave an overall lower rating to the Questions on Meeting Educational Objectives since their experience as a whole wasn't good. But a point worth noting that the response on Meeting Educational Objectives is higher than average, which was a good sign. This means even though the students had problems with the experiment, they still think that the Experiment met the Educational Objectives.

This was the first release of the system with no Charts and minimum control over the equipment. Student had different kinds of comments. Most of them seemed to be generous about this new experience and had a positive reaction, others however preferred a more "hands-on" experience. Some of the student comments are stated here:

Usability

- *"It's user friendly and easy to use and the instructions could have been more straight forward, i.e., seemed harder than actually was."*
- *"Very usable; however, time constraints by schedule gave me little time to check over everything."*
- *"The instructions could have been developed by including picture of the control panel and more constant flow rates would have been useful. Maybe a demo would be good."*
- *"Kind of confusing as to how to set new flow rates, never knew whether or not I actually set flow rates."*
- *"It was difficult to get numbers that are steady. Maybe build in an averaging function on the website.'*
- *"Steady state difficult to obtain"*
- *"Onetime the program froze on me. Another time I got really weird flow rates for concurrent. They were very hard to control."*

Improvements

- *"Putting basic equations of heat exchanger would clutter the site-we have textbook for that. It would be nice to have a webcam of the exchanger or something like that. The experiment seemed quite remote as we couldn't see any equipment, or have any feel for what we were controlling"*
- *"It would have been better if we could actually be in the room with the heat exchanger as we ran the experiment."*
- *"More visuals to what is happening on the heat exchanger"*
- *"Automatic data capture"*
- *"A more visual format would be good and make it easier to take reading, e.g.. A graph"*

- “Automate measurement of T and calculation of T_{avg} over 30 seconds”
- “Plots of data would be really nice to get a better idea of the trends while running the exchanger.”
- “More flexible timeslots.”
- “A troubleshooting section on the website would be nice.”

Educational Objectives:

- “I thought it met the objectives fairly well.”
- “Since the running of the experiment worked on an hour signup basis. Why not actually make it “hands on”. I would have much rather seen a heat exchanger in person rather than punching numbers on the computer”
- “Useful tool!!”
- “I would say it met the educational objectives.”
- “Since it is online, the “realness” is reduced to more like a simulation, but dealing with real data sets is useful.”
- “I think it was a valuable experience, but it seemed kind of rushed and not as clean as it could have been, due to time issues(i.e. the expt was explained late), next year should be better though.”
- “I liked using a real system”
- “Excellent job with iLab Heat exchanger! It's got great potential for use in the class. Our Chemical Engineering classes need more hands on experience, application especially in the earlier classes!”

Overall Experience

- “The experiment itself was ok. The ‘strict time slot’ bit was stressful. The writeup was repetitious, time consuming, and dull. Not “fun” at all.”
- “It was very daunting and I think we could have done with a little more discussion of how we were to analyze the data.”
- “Not necessarily fun, but educational.”
- “A better control system is highly necessary. Also, the cocurrent flow didn't work for me.”
- “Overall, I thought this was an interesting assignment, a good departure from textbook problems. Coming up with a good way to record measurements would be helpful. I would have liked to see the experiment carried out in class (to reduce anxiety level). I didn't have any problems with the ilab..got frustrated with the wording of some of the questions on the problemset.”
- “The biggest problem on the accuracy of data was the need to record it by hand. The numbers change too quickly to get a set of numbers for one time before something changes. Having the program tabulate the data would eliminate this problem.”
- “I think, though the exchanger did meet the educational objectives, it was difficult to observe them just by taking a list of temperatures. Other means of illustrating these objectives would be good.”
- “The website was fun-the long problem set questions weren't.”

6.3.2 MIT Course 10.26 Chemical Engineering Projects Laboratory: February 2002

A total of 37 students took the course and among them, 36 students filled out the assessment questionnaire. The main objectives of using the Heat Exchanger experiment in this course were

- Measurement of flow rates and temperatures in a real system
- Analysis of real data
- Consideration of measurement errors, simple statistics
- Fitting data to a straight line
- Preparing a schematic equipment diagram
- Writing a technical report

The responses were again segregated between two groups of students, based on responses to Question 3.

- Students who had problems controlling the flow rate
- Students who had no problems controlling the flow rate

The questions in the ‘Usability’ part and the ‘Desirable Improvements’ part were the same as in 10.302. The questions in the ‘Meeting Educational Objectives’ part were changed reflecting the objectives of 10.26. In the ‘Overall Experience’ part, one question was added to get response on whether the experiment was beneficial from the student perspective.

It is worth noting that the percentage of students who had problems controlling the flow rate decreased considerably from the first use of the system in 10.302. But we see similar patterns here. The students who could not control flow rates gave lower ratings to most of the questions especially to the questions on Overall Experience. This group of students also took more time to collect necessary data for the report. Despite having problems performing the experiment they gave a higher rating to the questions on ‘Educational Objectives’ part. They thought that the experiment was beneficial (Q15), but gave a

lower rating to question 16 stating that the experiment wasn't fun. Their comments reflected however that they enjoyed collecting data, but didn't enjoy analyzing the data and writing the report. Table 6-2 shows the responses to all the questions.

Table 6-2: Student response : 10.26 : Writing Technical Report

	7=Strongly Agree 1=Strongly Disagree			
	(Mean \pm SD)		Range	
	Problems controlling flow rate * N=2	All others N=34	Problems controlling flow rate * N=2	All others N=34
Usability when carrying out the experiment on the web	5.0 \pm 2.8 1.5 \pm 0.7 4.0 \pm 4.2 5.5 \pm 2.1 6.5 \pm 0.7 65 \pm 35	6.1 \pm 1.1 5.5 \pm 1.7 6.6 \pm 0.9 6.2 \pm 1.1 6.7 \pm 0.9 45 \pm 13	3 - 7 1 - 2 1 - 7 4 - 7 6 - 7 40 - 90	3 - 7 2 - 7 3 - 7 3 - 7 2 - 7 25 - 90
Improvements you would like to see on the website	7. Other means for specifying flow rates (such as _____) 8. Graph of measurements versus time 9. Table of measurements versus time 10. Control over how data is averaged and reported 11. Physical properties of water 12. Basic equations of heat exchange	3.5 \pm 0.7 6.5 \pm 0.7 5.5 \pm 2.1 4.5 \pm 0.7 3.5 \pm 0.7 5.5 \pm 2.1	3.3 \pm 2.1 5.9 \pm 1.4 5.4 \pm 1.7 5.4 \pm 1.8 5.6 \pm 1.9 5.4 \pm 1.5	3-4 6-7 4-7 4-5 3-4 4-7
Meeting Educational Objectives	13. The remotely controlled experiment provided an experience of measurement of flow rates and temperatures in a real system analysis of real data fitting data to a straight line preparing a schematic equipment diagram consideration of measurement errors 14. The assignment provided a vehicle for learning how to write a technical report	3.5 \pm 2.1 6.0 \pm 0.0 6.0 \pm 0.0 5.0 \pm 2.8 4.0 \pm 1.4 3.5 \pm 2.1	6.1 \pm 1.4 6.6 \pm 0.7 6.5 \pm 0.9 6.2 \pm 1.2 5.8 \pm 1.2 5.9 \pm 1.2	2-5 6-6 6-6 3-7 3-5 2-5
Overall Experience	15. The I-Lab heat exchanger experiment was a beneficial learning experience. 16. The I-Lab heat exchanger experiment was fun.	4.0 \pm 1.4 2.0 \pm 0.0	5.8 \pm 1.3 4.5 \pm 1.6	3-5 2-2
				2-7 1-7

Some of the student comments are stated here:

Improvements from before:

- "Flow rate control is MUCH better. Data collection is also MUCH easier by saving to file."
- "Many improvements were made - specifying exact flows rather than % & a printout of results made it much more user-friendly."

Usability

- *"I had problems connecting, but once I was connected, everything ran well."*
- *"It was easy to use once I found a computer with the appropriate plug-in. Even when I found a computer with the plug-in, I couldn't input values on my first try."*
- *"Biggest trouble was just logging in"*
- *"Very nice. Would like to see some kind of indicator on whether or not I'm taking data."*
- *"I had some difficulty loading the applet 2.0"*
- *"The instructions were clear. I was one of the first running the exchanger and there were problems connecting at the time, but otherwise very easy to use."*
- *"The flow rates are much closer to the value I assigned, compared to last term."*
- *"The most frustrating thing was getting the lab to actually run. From then on, it really wasn't so bad -- just hard to stop the fluctuation at what you think is steady state."*
- *"The data recording feature made it _much_ easier to do the experiment!"*
- *"The experiment was very user-friendly"*

Improvements

- *"It would be nice to see valve positioning to get a feel for sensitivity of the equipment"*
- *"No real improvements are needed. It'd just be nice if there was a program that could do all the boring work such as interpolating Cp's etc."*
- *"Some sort of indicator telling the experimenter that steady state has been reached"*
- *"All of the improvements I listed from 10.302 were made. Maybe data averaging would be nice."*
- *"An indicator that lights up when it is recording data"*
- *"For physical properties of H₂O, can create a site so that user inputs a temperature and exact values for Cp and Rho can be calculated automatically rather than having to do hand calculations."*
- *"Would it be possible to include the relevant physical props of water with the data that is automatically collected? Typing it in by hand took extra time"*
- *"I would have liked to see that the buttons I pushed had been pushed and not missed, especially the start & stop recording buttons."*
- *"More technical details about Heat Exchanger (exact measurements of spacing between tubes, etc.)"*

Educational Objectives:

- *"The educational objectives were met but there was so much number crunching of data that I felt pressed for time when it finally came to writing the report."*

Overall Experience:

- *"Report was not fun. Data collection was cool."*

- “*The _experiment_ itself was fun - the paper was _not_*”
- “*A lot of work*”
- “*I learned if it doesn't kill you it only makes you stronger*”
- “*I would just like to say that the website is so incredibly improved from last semester that I am in shock. I thought 10.302's I-Lab was the most annoying, confusing, non-functional thing I've encountered (no offense or anything), and this is _very_nice.*”
- “*The added feature of auto-data recording greatly enhances the accuracy of the experiment. Still needs lots of improvements to make it more compatible with various computers.*”
- “*I thought it was a good learning experience, but I was really bored while taking all the measurements - you aren't even in the lab where you can watch the apparatus instead you stare at a gray screen. Maybe there could be a camera the shows the apparatus (even though it doesn't move...).. okay maybe this isn't _that_great of an idea. My point was that it wasn't super exciting to just watch the numbers.*”

6.3.3 MIT Course 10.450 Process Dynamics, Operations, and Control: April 2002

A total of 8 students took the course and all of them filled out the assessment questionnaire.

The main objectives of using the Heat Exchanger experiment in this course were

- Define a control problem on the basis of a piping and instrumentation diagram.
- Use standard correlations to choose proportional-integral-derivative (PID) controller parameters.
- Fine-tune the controller parameters while operating the process equipment.
- Document controller performance by measuring the response to several disturbances.

Table 6-3: Student response : 10.450 : Small Project

	7=Strongly Agree 1=Strongly Disagree	
	(Mean \pm SD)	Range
	All students N=8	All students N=8
Usability when carrying out the experiment on the web		
1. The instructions were clear. 2. I had no problems operating the experiment on the web. 3. I was able to control flow rates during the experiment. 4. I was able to make steady state temperature measurements 5. I was able to obtain the data needed for the assignment. 6. The graphical display of data was useful 7. It actually took me about _____ minutes to carry out the experiment.	5.8 \pm 1.2 3.4 \pm 1.8 5.0 \pm 1.4 4.7 \pm 1.9 6.0 \pm 1.1 6.0 \pm 0.8 143.8 \pm 33.8	4-7 1-6 3-7 3-7 4-7 5-7 90-180
Improvements you would like to see on the website		
8. Other means for specifying flow rates (such as _____) 9. Table of measurements versus time 10. Control over how data is averaged and reported 11. Basic equations of heat exchange 12. Webcam	2.5 \pm 1.6 6.5 \pm 0.9 5.1 \pm 1.1 3.9 \pm 1.7 4.1 \pm 0.9	1-4 5-7 4-7 2-7 3-5
Meeting Educational Objectives		
13. The assignment provided an experience of defining a control problem on the basis of a P&I diagram using standard correlations to choose PID controller parameters fine-tuning controller parameters while operating the process equipment documenting controller performance by measuring response to disturbances 14. The assignment provided a vehicle for learning and reviewing principles of process control covered in class.	5.9 \pm 0.6 6.4 \pm 0.5 4.4 \pm 0.9 5.6 \pm 1.3 6.4 \pm 0.5	5-7 6-7 3-5 4-7 6-7
Overall Experience		
15. The I-Lab heat exchanger experiment was a beneficial learning experience. 16. Operation of the I-Lab heat exchanger experiment on the internet was fun. 17. The entire heat exchanger assignment was fun.	6.1 \pm 1.2 4.8 \pm 0.7 4.1 \pm 0.6	4-7 3-5 3-5

In general, the students gave higher ratings to the questions in ‘Meeting Educational Objectives’. The ‘Heat Exchanger Experiment was Fun’ question in 10.26 was broken into two questions to receive better feedback on the Heat Exchanger Experiment. The experience on ‘operation of Heat Exchanger was fun’ (Q16) received higher rating while experience on ‘the entire assignment’ received a lower rating probably due to its tedious data analysis and report writing part.

Some of the student comments are stated here:

Usability

- “Initially, more time was spent trying to run the experiment but being unable to do so due to equipment/computer problems. TAs were helpful and quick to respond though.”
- “Tough to get steady after others”
- “Flow rate erratic a few times during experiment. More graphs such as heat duty(Q) would have been helpful. Didn’t quite finish experiment, more time was needed.”

Desirable improvements

- “PID parameters included in the table of data captured on Excel.”
- “Interested in empirical response. Display of current controller settings”
- “Data records current tuning parameters(K_c , T_i , T_d). If applet opens in new window, so, instructions could be lift up.”

More Comments

- “How do we know when the previous group has logged off? A 1-hour block is too short.”
- “It was good to see the experiment actually running, good to get experience in general. Would’ve liked the chance to run some data, then had time to talk to instructor before running more. I hour block seemed rather short when trying to reach steady state, needed more time overall. What parameters(K_c , T_i , T_d) is the system running when you first connect? Is it the parameters from the previous group, or some set of “good” parameters predetermined? Some how we managed to lose some of our data in the excel sheet while recording-not sure if we made mistake or system did.”

6.3.4 University of Texas at Austin Course ChE 354 Transport Processes: April 2002

A total of 52 students took the course and all of them filled out the assessment questionnaire.

The main objectives of using the Heat Exchanger experiment in this course were

- Measurement of flow rates and temperatures in a real system
- Analysis of real data
- Fitting data to a straight line
- Preparing a schematic equipment diagram
- Writing a summary report

Table 6-4: Student response : UT354 : Homework

	7=Strongly Agree 1=Strongly Disagree	
	(Mean \pm SD)	Range
	All students	All students
	N=52	N=52
Usability when carrying out the experiment on the web		
1. The instructions were clear.	5.8 \pm 0.9	3-7
2. I had no problems operating the experiment on the web.	6.0 \pm 1.6	1-7
3. I was able to control flow rates during the experiment.	6.6 \pm 0.6	5-7
4. I was able to make steady state temperature measurements.	6.4 \pm 0.8	4-7
5. I was able to obtain the data needed for the assignment.	6.3 \pm 1.4	2-7
6. The graphical display of data was useful	5.4 \pm 1.3	2-7
7. It actually took me about _____ minutes to carry out the experiment.	38.9 \pm 14.6	15-60
Improvements you would like to see on the website		
8. Other means for specifying flow rates (such as _____)	2.8 \pm 1.8	1-7
9. Table of measurements versus time	4.6 \pm 2.3	1-7
10. Control over how data is averaged and reported	4.1 \pm 2.0	1-7
11. Basic equations of heat exchange	4.2 \pm 2.4	1-7
12. Webcam	4.4 \pm 2.5	1-7
Meeting Educational Objectives		
13. The remotely controlled experiment provided an experience of measurement of flow rates and temperature in a real system analysis of real data fitting data to a straight line preparing a schematic equipment diagram	5.3 \pm 1.9 5.3 \pm 1.5 5.3 \pm 1.5 4.6 \pm 1.8	2-7 2-7 2-7 1-7
14. The assignment provided a vehicle for learning how to write a summary report	4.4 \pm 1.9	1-7
Overall Experience		
15. The I-Lab heat exchanger experiment was a beneficial learning experience.	3.9 \pm 1.4	1-7
16. Operation of the I-Lab heat exchanger experiment on the internet was fun.	4.1 \pm 1.8	1-7
17. The entire heat exchanger assignment was fun.	3.3 \pm 1.6	1-6

Some of the student comments are stated here:

Usability

- *"Instructions were very clear and concise"*
- *"There was some erroneous information that was a little confusing but it was very easy to use (with a touch of a mouse). It made the measurement/data section go by very quickly."*
- *"User friendly."*
- *"We didn't wait long enough at first for the temperature to stabilize - got funky data at first"*
- *"Initially, 60 minutes spent at computer, but actual time spent on analysis of data was excessive and not as useful to me at the present as having time to study for finals would have seemed."*

Desirable Improvements

- *"Delete the extraneous information. Have it pertain to the experiment at hand."*
- *"A webcam would be cool so we could see what's really going on. All we seen are numbers which resemble HW problem, not a lab."*
- *"Multiple views of Heat Exchangers."*
- *"Do not allow access to the exchanger via web when doing maintenance & multiple log-ons."*
- *"It would have been better to have the data returned with an average and st. dev to minimize time spent on analysis."*

Additional Comments

- *"The assignment was not as valuable because it was due on the day of the final exam."*
- *"This experiment was redundant for those who are in funlab since they already have done a similar experiment on a heat exchanger. Also during our data collection, maintenance was done on the exchanger which made data collection impossible."*
- *"The experiment has its merits and I understand the motive behind having to perform it while taking a course on heat transfer; the timing of the assignment for us as a class was terribly inconvenient, though. These are the only few days I have to study for finals. The instructions for data analysis were not clear and it was quite tedious to keep going back and correcting"*

6.4 Comparative evaluation

From the previous section, it is evident that the students who had problems performing the experiment gave lower rating to the questions. Figure 6-1 shows this phenomenon in 10.302, but at the same time, the ‘Meeting Educational Objective part’ got higher rating than the other parts of the assessment. The system in 10.302 was a prototype with minimum functionalities. We received a lot of student responses about the system and about potential additional features to the successive versions.

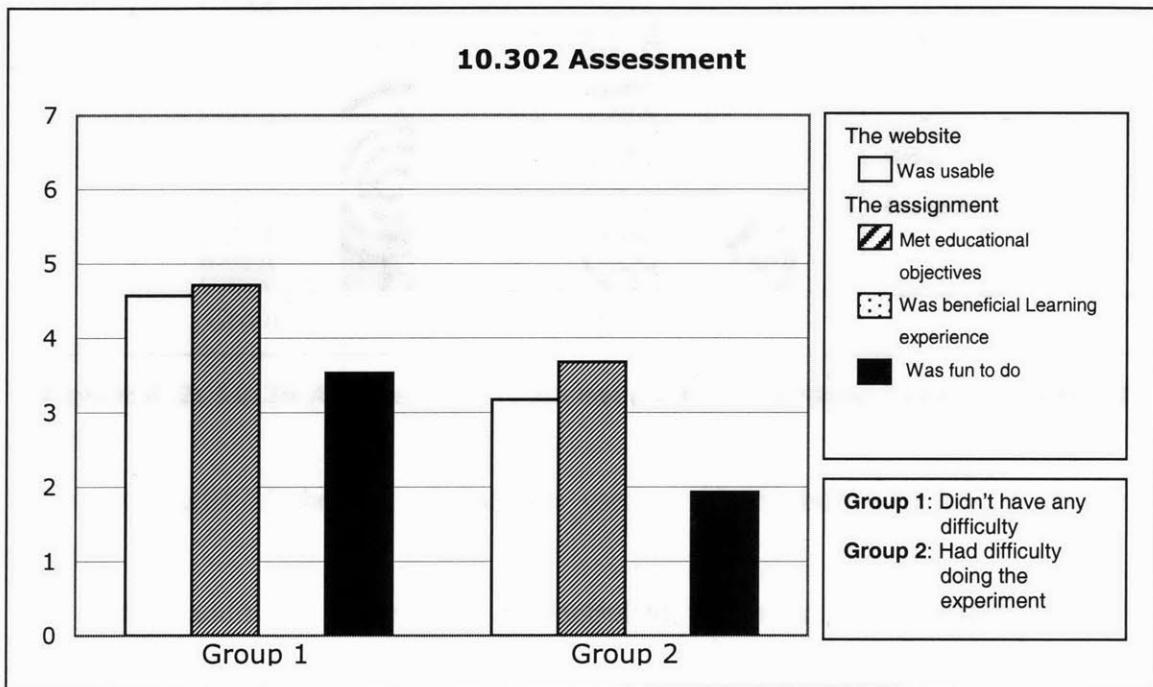


Figure 6-1: 10.302 Assessment (November 2001): Extended Homework

The second release of the system that was used in 10.26, had a better control of the Inlet Hot Fluid Temperature with the deployment of an efficient PID algorithm. As a result, steady state measurements was possible. Another significant feature was the Data Logging capability. In 10.302, the students had to record all the data by hand, where as in 10.26, an automatic data recording capability was added. The students who took both 10.302 and 10.26 expressed that it was a very helpful feature in terms of data collection for the report. As a result, the ratings in 10.26 were higher than that of 10.302 as shown

in Figure 6-2. Another point worth noting that the students thought that the Heat Exchanger Experiment was a '*beneficial learning experience*' although it was not '*fun*.'

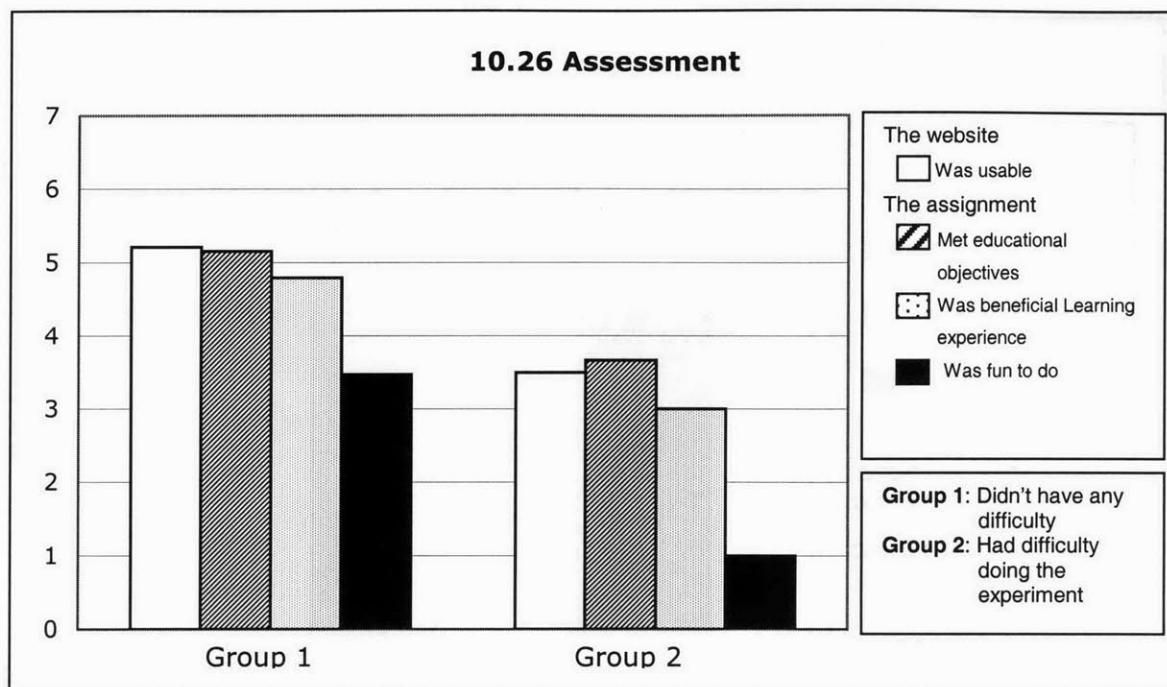


Figure 6-2: 10.26 Assessment (February 2001): Writing Technical Report

The next release of the system that was used in 10.450 had three major additional features.

- Three additional parameter to be controlled, which are the parameters of the PID controller,
- A graph showing the variation of hot fluid inlet temperature with time,
- The capability of giving the students limited access to the parameter control. This is controlled from the server-side LabView software. For example, if the problem set does not ask the students to change the set point temperature of the hot fluid inlet, then this parameter is set to '*Not Controllable*' and the corresponding control on the Java Applet becomes grayed out and the students don't need to think about it.

Figure 6-3 shows an increase in the rating of the questions in different categories and the '*beneficial learning experience*' receiving considerably higher rating.

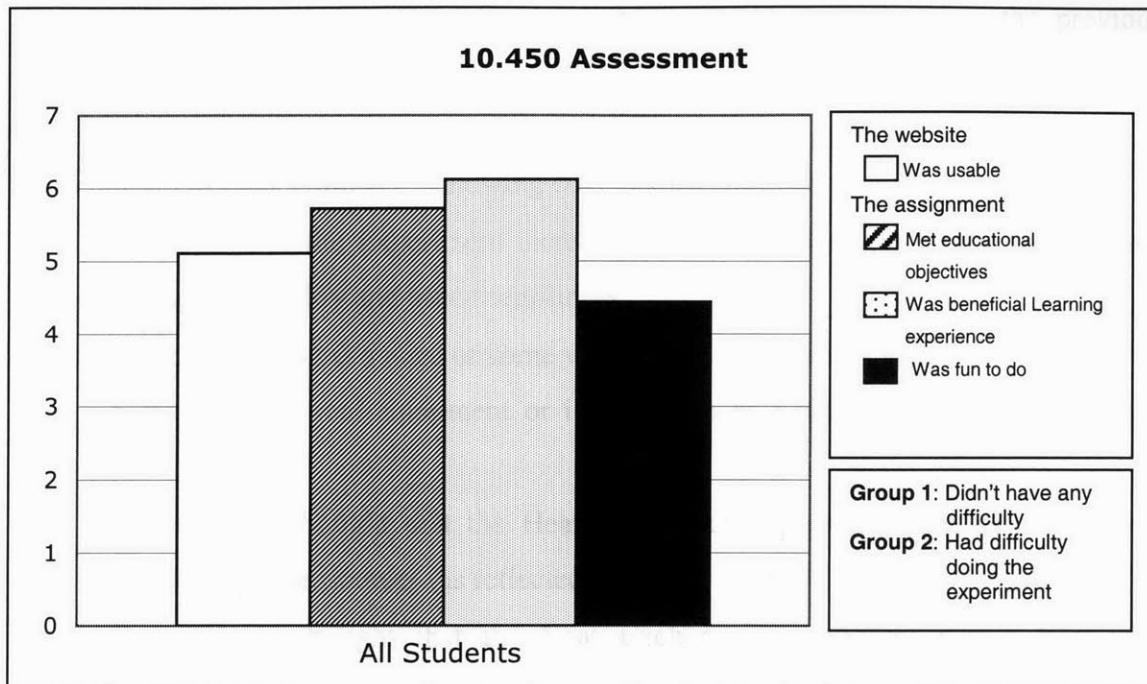


Figure 6-3: 10.450 Assessment (April 2001): Small Project

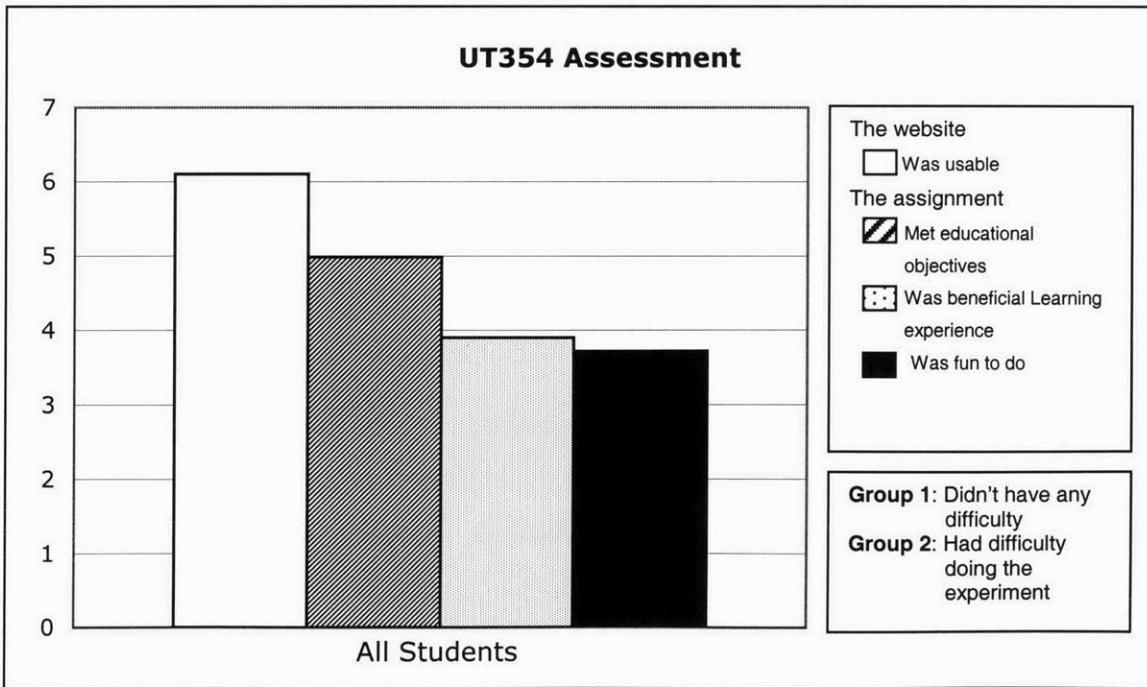


Figure 6-4: UT 354 Assessment (April 2001): Homework Problem

Figure 6-4 shows the assessment summary in UT354. The responses got a lower rating than 10.450 except the ‘*usability*’. This might be because the experiment was conducted just before the final exam and the timing of the experiment was inconvenient on the student part. The student expressed this in their comments as stated in the previous section.

From the assessment summary, following conclusions emerged

- The ratings of the Experiment increased with added features on the website and increased usability and user-friendliness.
- Students expressed a need of some visual on the equipment either in the form of a demo with the real equipment or in the form of a real-time video feed on the website.
- The students thought that the Heat Exchanger experiment met its educational objectives in the courses (as reflected by higher ratings).
- The students thought that the Heat Exchanger experiment was a beneficial learning experience in their part (as reflected by higher ratings), although the entire experiment including the data analysis and report writing was not fun.

Chapter 7. CONCLUSION AND FUTURE IMPROVEMENTS

The Objective of the I-Lab Heat Exchanger project was to design and implement a real-time, robust and scaleable software platform around a laboratory heat exchanger equipment to develop a web-accessible heat transfer experiment for online use in courses.

At each stage of the research effort, a defined set of goals and deadlines were set. Upon successful completion of these tasks, a new set of tasks were assigned. Improvements were made in the software platform with addition of new features at each cycle of the development process.

7.1 Summary of Accomplishments

A summary of accomplishments made to-date was described in the 2002 proposal of the I-Lab project [6]. Some of the key accomplishments are excerpted from the document and stated as follows:

- Worked with manufacturer (Armfield Ltd) to develop heat exchanger equipment that is instrumented with electrically-activated valves and switches so as to be remotely controllable from a computer. The instrumentation involves seven variables, four of which are independently controllable.
- Developed LabView software to monitor and control the equipment locally from the server. This software has the ability to publish real-time data to the

internet and get real-time input from remote users over the internet through DataSocket server at specific URIs for each data points.

- Developed a client-side interface in Java for remote control and monitoring of the equipment over the internet with either Netscape (Athena) or Internet Explorer browser. This Java applet employs the DataSocket API to connect to the DataSocket server at specific URIs in order to get real-time data from the equipment through the server-side LabView interface and also to publish parameter control values to the DataSocket server at real-time to control the equipment.
- Calibrated and tested hardware performance.
- Although a primitive prototype of the system was launched on 11/16/01 and the experiment ran 19 hr per day for 4 days in 45-min segments to accommodate about 65 students in 10.302, this early test uncovered heretofore unknown hardware and software deficiencies leading to problems in operation of flow rate and temperature controls.
- Assessment questionnaire filled out by students provided valuable insight into usability aspects of website and other features in need of improvement. Results demonstrated that meeting educational objectives was scored higher by the 75% of students who operated the experiment without major problems than by the 25% who had problems.
- Improvements in hardware, software, and website performance were made in January '02, leading to stable control of cold water flow and inlet hot temperature and also scheduling capability on the website.
- This new release was used in 10.26 as weblab project, which replaced previous simulated weblab. This new release was launched on 2/10/02 and run 15 hr per day for 8 days in 1-hr segments to accommodate 47 students in 10.26. Results from assessment questionnaire document markedly enhanced experience of most students with poor ratings from a few students who experienced severe problems stemming from incompatibility with some Athena terminals. Results of this test reinforce the finding that I-Lab concept provides high educational benefit when the system (hardware, software,

website) works well but that perceived benefit vanishes when the student's experience is flawed.

- There were two successive uses in 10.450 and UT 354 (University of Texas) in 4/27/02 with added control and added graphical components.
- The collaboration application was developed and tested during May 2002 and will be used in Fall 2002.

7.2 Future Work

The research work in this project is an ongoing effort. There are many additions that can be made.

- Improve usability of I-Lab heat exchanger website
 - Includes capability for improved data acquisition and analysis (smoothing and averaging, variable sampling period, simple statistics), data presentation and recording (more real-time graphs and tables on the web) on the client-side interface, equipment diagrams and descriptions, webcam and microphone (video and audio streaming) on the website.
- Explore other possible client-side interface development platform
 - In the new release of LabView, it is possible to access a LabView VI through the web browser provided the client machine has a browser plug-in (LabView Runtime Engine) installed. Before making any progress in this path, it is very important to consider issues like user authentication and whether this platform will support collaboration. But this is a very promising platform since it will reduce the development effort in terms of GUI development.
- Develop portfolio of heat exchanger experiments on common platform
 - Currently, only the shell and tube heat exchanger is being used in experiments. Development of a base platform employing three other types of heat exchangers (double-pipe, flat plate and stirred vessel), with experiments pertinent to the characteristics of each heat exchanger is one of the future tasks. Assignments will be based upon steady state behavior,

as well as transient behavior for investigating system dynamics and control.

The Heat Exchanger Experiment System is currently operational and can operate independently with minimum supervision. However it is important to assess, document and disseminate the educational know-how generated in the course of this research to the broader academic community in order to develop, deploy and experiment a universal software framework and standard that would allow fast development of new I-lab experiences independent of the particular physical platform

APPENDIX

Operating the system

1. Operating the Heat Exchanger equipment

The procedure of turning on the heat exchanger equipment is described in details on Page 19 of the Instruction Manual for the HT30XC Service Unit [3]. The procedure is described again for convenience.

Step 1: Connect the HT30XC to the electric mains supply and check that the ‘*RCD*’ at the back of the HT30XC is switched on with the lever in the up position. The ‘*Control*’ light should illuminate red, showing the unit is in ‘*standby*’ mode.

Step 2: Release (pull) the ‘*standby/enable*’ button, the ‘*Control*’ light should now turn green showing that the unit is in ‘*standby*’ mode. Note: standby mode means that the unit is capable of being switched fully on, subject to the Emergency Stop button not being pressed and to the unit receiving watchdog pulses from the software.

Step 3: Release (pull) the ‘*Emergency stop*’ button.

Step 4: Check the ‘*power*’ led and the ‘*active*’ led on the front of the HT30XC unit. The ‘*power*’ led should be red and the ‘*active*’ led should be green. The ‘*power*’ led turns to green when the power is turned on in the LabView software.

2. Operating the LabView software

Step 1: Start the DataSocket server by pressing **Start>Programs>National Instruments>DataSocket>DataSocket Server**. The DataSocket server starts and starts to listen for publishing and subscribing clients.

Step 2: Start the LabView software by pressing **Start>Programs>National Instruments>LabView6>LabView**

Step 3: Open the latest version of the LabView software. The latest version of the software is in the folder with highest version number and the latest file has the highest letter associated to it. For example, the latest version of the software is in the folder **VersionRHW5** and in this folder, the latest file is **Heat Exchanger4f.vi**

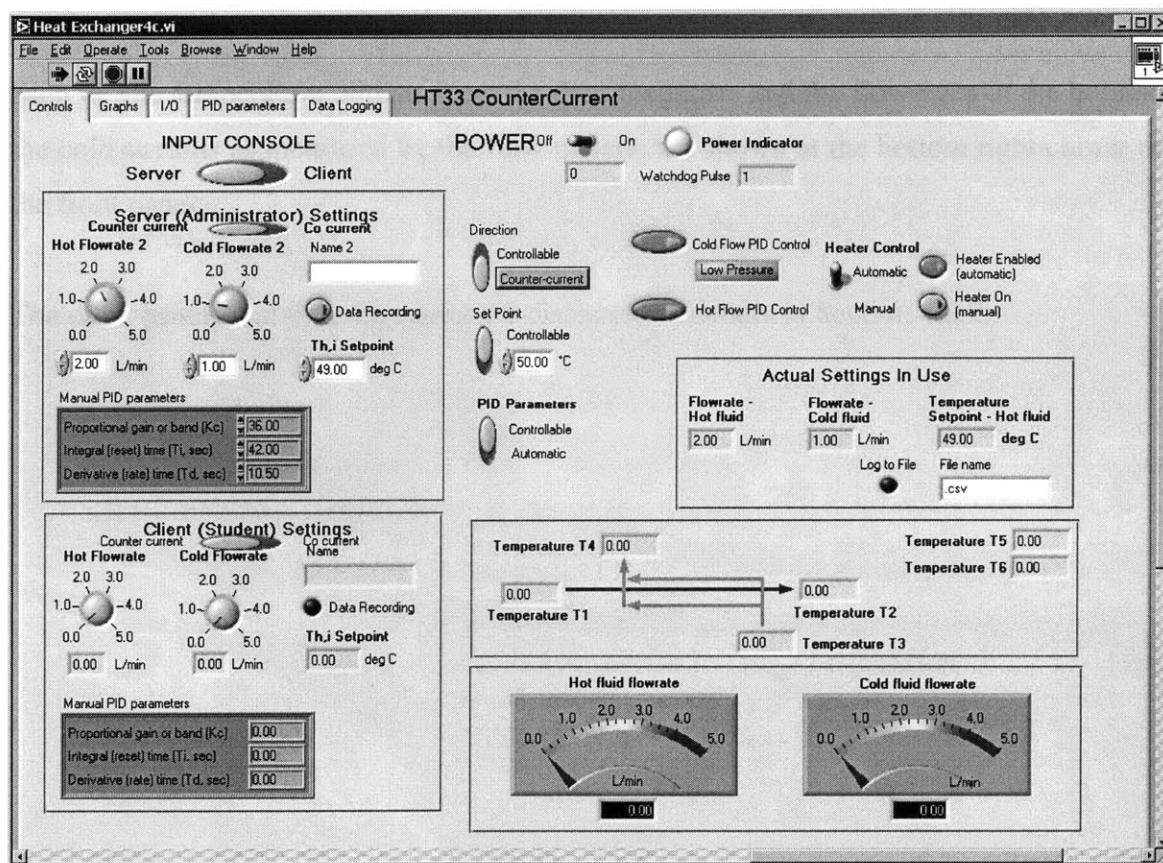


Figure A1: The Controls panel of the Front panel of the LabView Control and monitoring software

Step 3: Press the ‘Run’ button on the toolbar. Turn the ‘POWER’ on. The ‘power’ led at the front of the HT30XC Service Unit should light up.

Step 4: In the '*INPUT CONSOLE*', set the control to '*Client*', to allow the remote users to control the equipment. The data that the remote user inputs is shown in the '*Client Settings*' panel.

Step 5: Set the permission to the parameters that are to be controllable by the user. These buttons are in the middle of the Controls panel. The '*Direction*', '*Set Point*' and the '*PID Parameters*' can be set to controllable. If they are not set to controllable, then the corresponding controls in the client-side Java applet are grayed out. So, the users are not being able to control these parameters from the applet.

Step 6: The temperature outputs from the thermocouples and the flow rates of the hot and the cold streams as measured by the flow meters, are shown at the bottom right corner of the front panel.

The other panels and their functions are discussed in details in Section 4.2.2.

3. The client-side Java Applet

Computer Requirement

Operation of the experiment requires a browser and plug-in that support Java 2 Runtime Environment. The experiment can be operated on a PC with Internet Explorer 5.0 or higher or Netscape Communicator 4.74(MIT version) or on a Mac running MacOS X with Internet Explorer 5.0.

On PC running windows 98/2000/XP, download and install the Java 2 Runtime Environment from the following link. <http://java.sun.com/j2se/1.4/download.html>

On Athena, you need to type the following command before you load Netscape.
athena% add -f java_v1.3.1_02

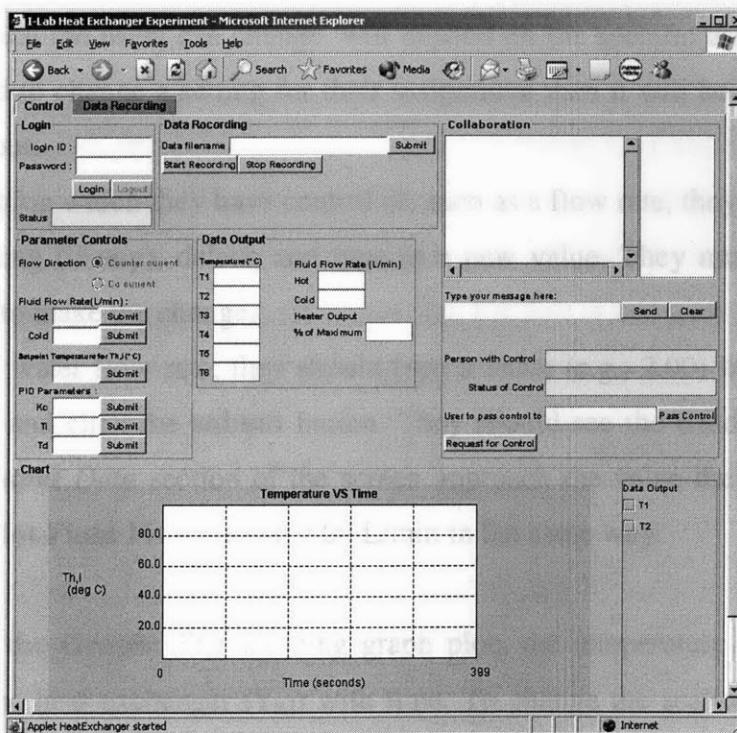


Figure A2: The client-side Java Applet

Step 1: Connecting to the Experiment: The users should go to the website <http://heatex.mit.edu> and sign in to the member login page and then click on the **Perform Experiment** link. The Java Applet should load. If it doesn't, please see the computer requirement section. The users should input their LoginID and Password once again to login to the experiment. Upon successful authentication, a 'success' message is

shown in the ‘*Status*’ window. The temperatures and flow rates should be visible in the ***Data Output*** section of the screen.

The name of the user who is in control of the equipment is shown in the **Person With Control** dialog box. The control of the equipment is assigned automatically to the first person that signs in. In order to get the control of the equipment, the user should press the **Request for Control** button. This sends a message to the person with control of the equipment. He/she can deny the request or accept the request by putting in the name of the person to pass the control to and pressing the ‘Pass Control’ button.

Step2: Setting Controllable Parameters: The same interface is used for several different assignments so it is designed to allow control of many parameters. The parameters, which the users can change, will depend on the specific assignment. If they are not required to change a setting for their assignment then it will be inactive and will appear grayed out.

To change a setting which they have control of, such as a flow rate, they should highlight the current setting (if any), delete, and type in a new value. They must then click the **Submit** button to make the change.

To set the cold water flow rate, they should type a value (e.g., 2.00) in the **Cold Fluid Flow rate** box and click the **submit** button. They should see the Cold Fluid Flow rate value in the ***Output Data*** section of the screen approach the value they have set. They should set the **Hot Fluid Flow rate** to 3.00 L/min in the same way.

Step 3: Using the Graph: The scrolling graph plots the temperature of the hot water flowing in to the heat exchanger ($T_{h,i}$) with time. To change the scales of the axes, the users should click on the graph with the right mouse button. A new options window will appear with values they can change. It is suggested that they start with the following settings.

Time (seconds)	X Min	0.00
	X Max	300.00
Temperature (°C)	Y Min	20.00
	Y Max	80.00

They can change the scale on the Y (temperature) axis whenever they need.

Step 4: Recording Data: Using the **Data Recording** controls, the users can save all the information from the heat exchanger experiment. Values of the temperatures, flow rates, set point temperature and heater power, depending on their assignment, can be automatically recorded every second, downloaded from the website after the experiment, and opened directly into a spreadsheet.

The user should input a name that is going to be used as the data filename and press **Submit**.

When they wish to start recording the data they should click the **Start** button. If they wish to pause the data recording between experiments then they should click the **Stop** button. Restart the recorder when they want to start the next reading. This will insert a break in the data and new column headings so they can easily record the data from several experiments in one file. You can check that the data is being recorded by opening <http://heatex.mit.edu/data/username.csv> (where *username* is the Data Filename they used to record the data) in a separate browser window (**File>New>Window** or shortcut key Ctrl-N).

Step 5: Collaboration: The collaboration part allows users to work in a group and share thoughts and comments with the help of text chat. This also allows users to pass the control of the equipment among them.

Step 5: Disconnecting: When the users have completed the experiment, they should logout of the system by clicking the **Logout** button.

4. User Registration, Authentication and Scheduling

The User Registration, Authentication and Scheduling are described in details in Section 5.3.2. But prior to the use of the system in any course, the schedules are to be input in the database manually. The steps involved in doing so are:

Step 1: Open **Start>Programs>Microsoft SQL Server>Enterprise Manager.**

Step 2: Double Click on **Microsoft SQL Servers>SQL Server Group>(local)>DataBases.** The database **classlist** shows up. Double click on it. Then click on the **Tables.** Right click on the table named **Signup** and choose **Open Table>Return all rows.** Now manually input all the schedule times in the **Date** field. Put 1 in all the **StudentID** fields. Put 0 in all the **Occupied** fields. The **SignupID** field is automatically generated by the database.

5. Collaboration

The Collaboration Java Application needs to be run before the students can log in to the experiment through the Java Applet. This Java Application handles the user authentication and all the collaboration among the users through Socket connection. Open and run the Java application in the following manner.

Step 1: Open the command prompt by clicking **Start>Run...and then type cmd and press enter.** This will open up a command prompt.

Step 2: add path to the current JDK directory by typing the following at the prompt

C:\>path C:\JDK1.4\bin

The JDK path could be different from C:\JDK1.4. Please input the path of the installation of the JDK in your system instead of C:\JDK1.4

Step 3: Change your directory to the directory of the Java Collaboration application and at the prompt type

Javaw ChatServer

This should launch the Application.

Step 4: Press the **Start Server** button. This starts the server and the server starts to listen for clients at a particular port.

REFERENCES

- [1] Alamo, Jesús A.del., “*I-Lab*”, MIT Engineering Council, April 9, 2001.
- [2] Alamo, Jesús A.del., “*Proposal to I-Campus*”, MIT, December 10, 1999.
- [3] Armfield Limited, “*Instruction Manual: Computer Controlled Heat Exchanger Service Unit HT30XC (HT30X-Remote-AL11036)*”, July 2001.
- [4] Bartlett, Dean A., “*The Fundamentals of Heat Exchangers*”, American Institute of Physics, 1996, <http://www.aip.org/tip/INPHFA/vol-2/iss-4/p18.pdf>
- [5] Burrell, B., Wiggins, R. J. N., SonWalkar, N., Kutney, M. C., Dalzell, W., Colton, C. K., “*A Comparison of Web-Based and Laboratory Learning Environments*”, American Society for Engineering Education, June 2000, <http://www.asee.org/conferences/search/20535.pdf>
- [6] Colton, Clark K., “*I-Lab Proposal*”, MIT, February 2002.
- [7] Incropera, Frank P. and Dewitt, David P., “*Fundamentals of Heat and Mass Transfer : 5th Edition*”, John Wiley and Sons, August 9, 2001.
- [8] Microsoft, “*SQL server 2000 for JDBC, User’s Guide and Reference*”, April 2002.
- [9] Travis, Jeffrey, “*LabView for Everyone, 2nd Edition*”, Prentice Hall.
- [10] National Instruments, “*National Instruments DataSocket Java Bean Documentation*”, National Instruments Corporation, Austin, Texas.
- [11] National Instruments, “*Integrating the Internet into Your Measurement System: DataSocket Technical Overview*”, National Instruments White Paper No. 1680, National Instruments Corporations, Austin, TX.
- [12] Sudarshan, R., “*A Web-Based Virtual Laboratory for Monitoring Physical Infrastructure*”, SM Thesis, Department of Civil and Environment Engineering, MIT, 2002.
- [13] Sun Microsystems, JavaTM 2 Platform, Standard Edition, (J2SETM), v1.4.0 API Specification, Sun Microsystems Inc., CA, <http://java.sun.com/j2se/1.4/docs/api/>