A JAVA TOOL SET FOR MONITORING LAB AT FLORIDA STATE UNIVERSITY

By

Esra Erimez

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Computer Science Department at Florida State University has been validating access to its computer labs maintaining log sheets manually. The difficulties experienced in tracking information and monitoring the collected data has lead to the development of a new utility facilitating fast, accurate and organized collection of user data. This report details the design, implementation and maintenance of a Java Tool suite designed to automate card validation of users entering computer labs and to support analysis and query of the collected data. The system implements Java threads and takes advantage of log files created during validation and an Oracle Database Server 8i available on one of the department's servers. The tool suite consists of Reader Interface, Configuration Tool, Database Manager Interface and Report-Query Tool. Reader Interface transfers data from the card reader hardware to log files. Configuration Tool allows GUI configuration of settings for the Reader Interface. Database Manager, a multi-client server transfers data to an Oracle database. Report-Query Tool creates graphs and performs queries to analyze collected data. The tools have been built in Java to provide a solid basis to maintain platform-independence. Currently, the Reader Interface is being used in majors lab in Computer Science Department.

1. INTRODUCTION

Computer Science Department at Florida State University offers its students the ability to use several computer labs, with majors lab hosting about 50 computers**.** Lab assistants monitor computer labs to assist in user support and to validate access to the labs. In the past, the validation process consisted of manually recording student name, time in and out on paper. The collected data was easily lost, unorganized and difficult to analyze.

Scope of this project addresses the issues of automating the card validation process and therefore ensuring efficient retrieval and subsequent analysis of student access to the labs by recording data in a reliable format. The nature of the environment in which the system is implemented in brings the following specifications to the project:

Ease of use: Interface should be very simple to allow fast validation by acquiring the information from the student's FSU ID and giving instant visual feedback to the user. The interfaces that provide analysis should give quick responses. Simplicity reinforces the tendency to use the application.

Robustness: Computer labs are open more than 8 hours a day. The application must run consistently with very little maintenance requirements.

Accuracy: Recorded data is adopted for statistical analysis taking into consideration the capacity of the lab to provide guidelines in determining future revisions of the labs. Therefore collected data should be reliable.

Security: The components sensitive to data collection or data privacy should be protected.

Support to monitor Security: Dedicated personnel to investigate cases of theft or damaged equipment can use the collected data. Student's ID is required to enter the lab and must be recorded by the program.

Ease of Installation: The program should be configurable to accommodate usage in different locations.

Platform-independence: Computer Science Department maintains different operating systems: WindowsNT Server 4.0, Unix Solaris, Red Hat Linux. The system should support platform independence and provide a basis to easily upgrade the components to work in different platforms.

Supervision from a remote location: A feature to monitor lab usage in computer labs.

2. SYSTEM DESIGN AND IMPLEMENTATION

Implemented as a Java tool suite, the system constitutes the following modules (Figure 2.1):

- Reader Interface
- Configuration Tool
- initDB Tool
- DBManager Interface
- Report Tool
- Query Tool

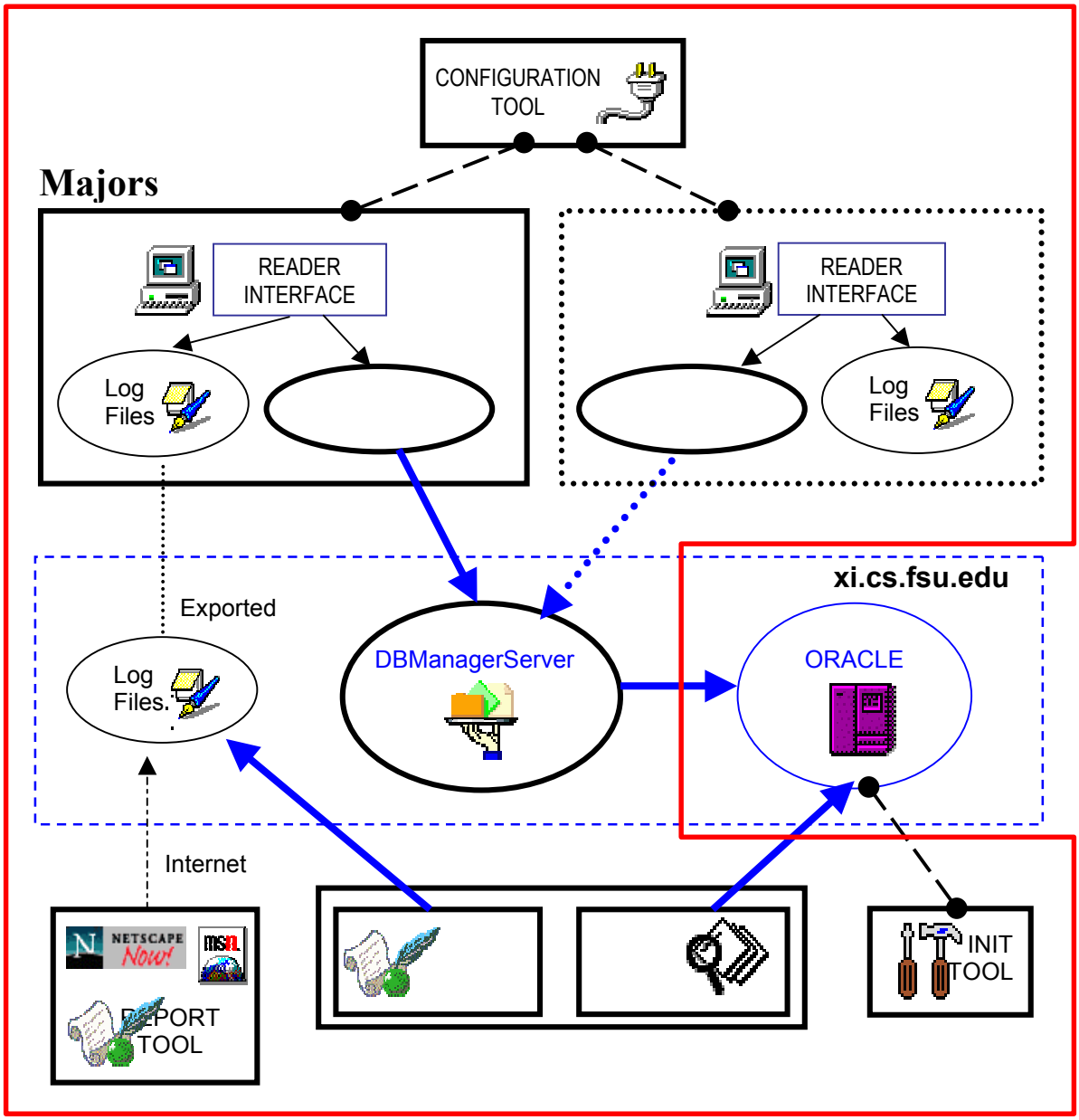

Figure 2.1: System Structure

The system is supported by several helper components.
Student provides student object structure. The supported fields are:
cardnum: student's card number,
name: student's name,
lname: student's last name,
datein: the date and time the student came in,
dateout: the date and time the student left,
labname: the lab the student was in,
intervalStr: the length of time the student stayed in hour:minutes:seconds format.

ReaderWriter handles the reading and writing of log files and temporary files.

DateIO implements useful date manipulation functions.

2.1. Reader Interface

Reader Interface transfers data from the card reader hardware to the computer's local hard-drive, to the screen and if database enabled, to the *DBManagerClient*. The interface is installed as a stand-alone running application (Figure 2.1.1).

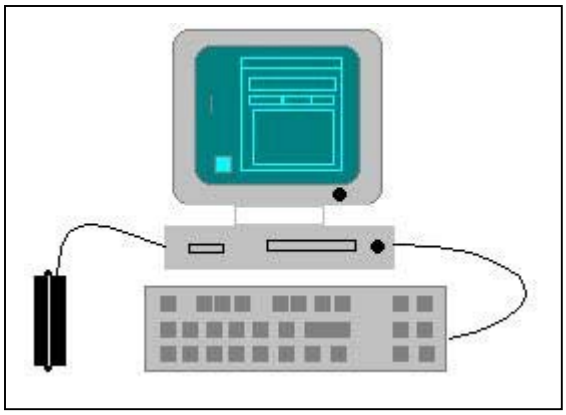

Figure 2.1.1: Lab-assistant's machine

Reader Interface maintains two types of log files.

- *labname.log*: A text file containing all the entries representing users who have been validated entering the lab "labname". For example, for majors lab, this file is majors.log.
- *labnamedate.log*: A text file containing all the entries representing users who have been validated leaving the lab. The corresponding entry is removed from labname.log, and a time stamp is added to the entry when the user leaves.

The fields in the log files are separated by a "|" character for future parsing.
The fields in labname.log are:
ID|LASTNAME|FIRSTNAME|DATEIN|LABNAME
The fields in labnamedate.log are:
ID|LASTNAME|FIRSTNAME|DATEIN|DATEOUT|INTERVAL|LABNAME

Below is a partial listing of a log directory.
majors.log
majors01102000.log
majors01112000.log
majors01122000.log
majors01132000.log

The following is a sample entry from majors.log:
5894371000479397|JOHNSON|KURTIS F|Thu Mar 30 11:26:39 EST 2000|majors

The following is a sample entry from majors03302000.log:
5894371001611550|STEEDMAN|RONALD J|Thu Mar 30 09:26:14 EST 2000|Thu Mar 30 09:59:45 EST 2000|2011|0:33:31|majors

The interface is a combination of threaded and helper components working together in coordination (Figure 2.1.2):

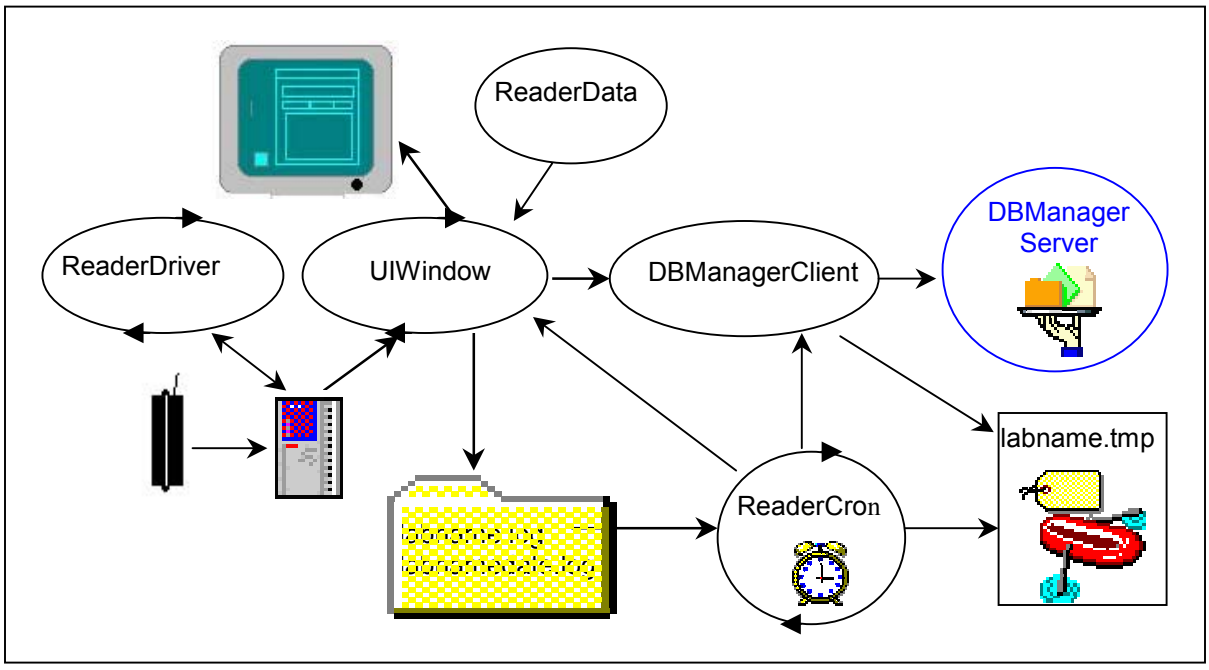

Figure 2.1.2: Reader Interface Structure

ReaderData contains the global settings for the reader interface. Two types of settings are implemented: Settings that need to be determined before compilation and settings that are given default values and can be configured later with Configuration Tool or by changing the configuration file. Pre-compilation settings are:
INSTALLDIR: Absolute path to the directory the application is installed in. The default value is /usr/local/reader
CONFFILE: Absolute path to the file that contains the configuration settings. The default value is INSTALLDIR+"portconf.txt".

DaySchedule provides the structure for day schedule. It consists of the following fields:
Name: name of the day,
Opentime: lab opening time,
Closetime: lab closing time,
AMCloseFlag: At or after midnight closing flag. It is set to TRUE if the lab closes at and after midnight.

Schedule provides the structure to contain day schedules for a week.

ReaderDriver, implemented as a thread, continuously waits for data from the port the card reader hardware is connected to. After the card is swiped, the card reader hardware sends the data read from the card as two fields separated by a termination character of carriage return. The first field contains the card id, the second field holds the name in lastname/firstname format. ReaderDriver parses this input to obtain user's card id, last name, first name, and sends the information to ReaderMonitor.

ReaderMonitor synchronizes the transfer of data from ReaderDriver to UIWindow.

UIWindow implements the graphical interface with the help of AddJPanel and UsersJPanel and continuously running as a thread takes the new data from the ReaderMonitor and checks if the entry is a special entry sent by ReaderCron to signal log out of all the entries from *labname.log* to *labnamedate.log*. If so, all the entries are removed as described below. If not, UIWindow checks if the entry already exists in an array that mirrors the contents of labname.log. Then the following operations are performed:

- If the entry does not exist, the datein field is set to the current time stamp and the data is added to labname.log. The display is updated to depict the change.
- If the entry exists, the dateout field is set to the current time stamp and the entry is removed from labname.log and appended to labnamedate.log. The display is updated to depict the change. UIWindow. If database connectivity is enabled, ReaderWriter's functions are used to pass the entry to DBManagerClient.

AddJPanel provides the graphical interface to collect the information input by the user.

UsersJPanel provides the graphical interface that displays the entries validated by the user on a table and gives the total count.

ReaderCron wakes up periodically at 2:00 A.M. and sends a special message to UIWindow by putting a student instance with a name field set to "CRONSTD" to ReaderMonitor . This is a signal for UIWindow to log out the remaining entries. This way the next day an empty file is ready to be used by the interface and the logs for the day are closed. Another duty of ReaderCron is to check *labname.tmp*. If this file is not empty and database connectivity is enabled ReaderCron forwards the entries to DBManagerClient.

HelpWindow implements the help information display.

The files are located on the local hard drive and the use of the interface is therefore not affected by failures on the network and/or of the database server.

2.2. Configuration Tool

Reader Interface reads its settings from the configuration file *ReaderData.CONFFILE*. Configuration Tool implements a GUI to allow configuration of settings known to the Reader Interface. The keywords that are assumed by the application and the mapping of the keywords to corresponding configurable values are:
LOCATION: ReaderData.LABNAME
PORT: ReaderData.PORT
LOGDIR: ReaderData.LOGDIR
DBENABLE: ReaderData.DBENABLE
SERVERHOST: ReaderData.SERVERHOST
SERVERPORT: ReaderData.SERVERPORT
SCHEDULE: ReaderData.SCHEDULE
ReaderWriter after reading the keywords expects the setting for that keyword is given on the next line. Everything else is disregarded.

Settings manages the reading and writing of the configuration values in *ReaderData.CONFFILE.*

SettingsWindow implements the GUI interface to set the parameters. When the OK button is clicked *ReaderData.CONFFILE* containing the configured settings is created.

2.3. DBManager Interface

DBManager Interface handles the collection and transfer of user data to the Oracle database. A *clients.allow* file contains the names and IP addresses of the clients allowed to make requests from the server. The entries should be added to this file in the following format:
Hostname/IP address
e.g: lov5bcard/128.100.100.100

The interface consists of the following (Figure 2.3.1):
DBManagerClient contacts DBManagerServer listening on a known TCP port and sends the user data. If DBMnagerServer does not respond, the data is written to a temporary file *ReaderData.LOGDIR+FILESEPARATOR+ReaderData.LABNAME+.tmp* on the client host.

DBManagerServer, a multi-client server, opens a socket and keeps listening on an pre-assigned TCP port for requests from DBManagerClients. When DBManagerServer receives a request, it spawns a DBManagerServerThread and forwards the request to this newly created thread to process the incoming data.

DBManagerServerThread tries to connect to the Oracle server and send the received data to the database server with OraConnector. If a connection is not established then DBManagerServerThread puts the data to WriterMonitor and exits freeing back resources to the system.

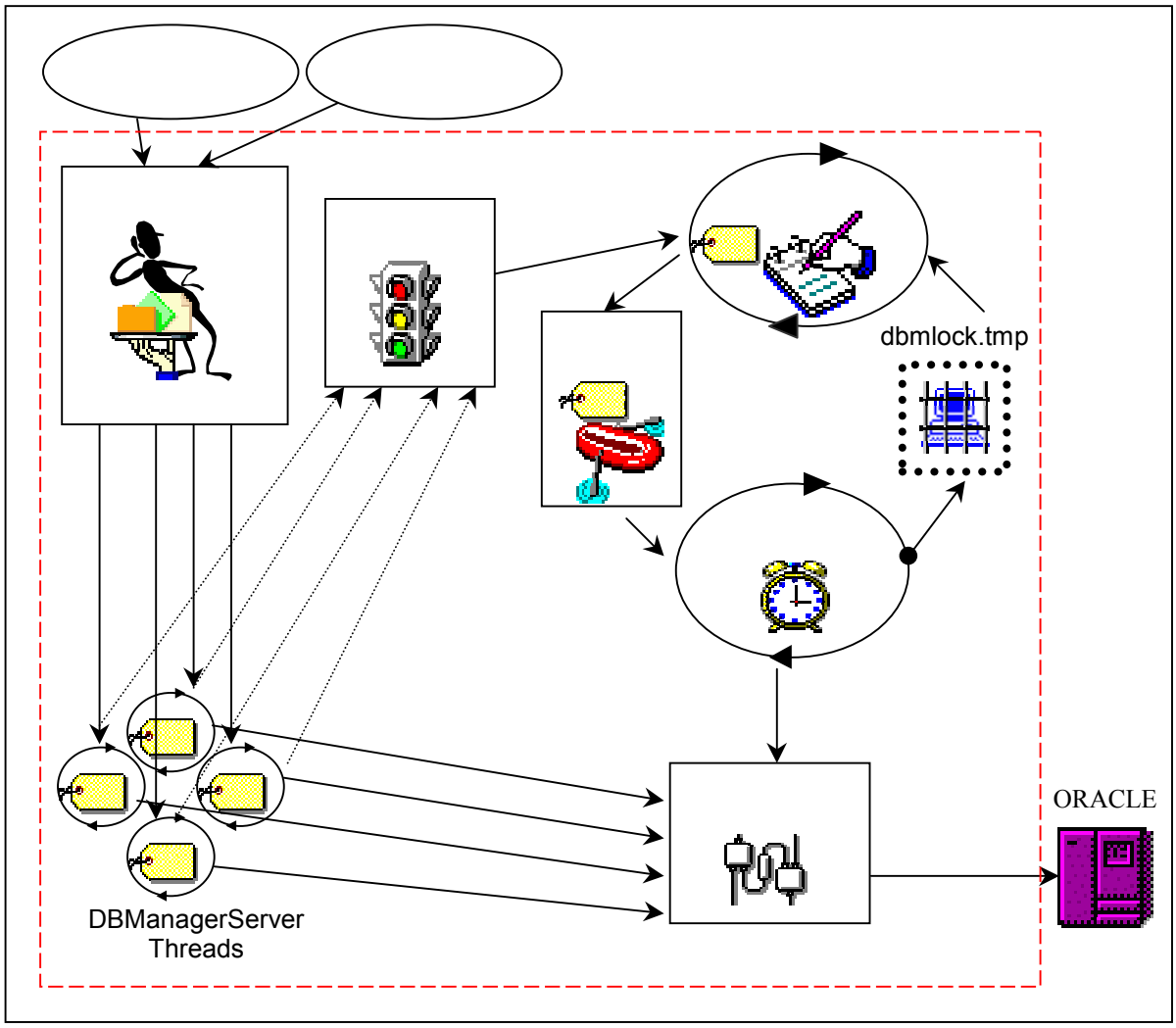

Figure 2.3.1: DBManager Interface

WriterMonitor synchronizes the transfer of data from DBManagerServerThread to WriterThread.

WriterThread is a continuosly running thread waiting for data from WriterMonitor and handles the writing of temporary *dbm.tmp* file in cases when the Oracle server did not respond. When the WriterThread gets data, it checks if the data is coming from DBCron. If the received data is not from DBCron it calls ReaderWriter to write the entry to dbm.tmp. Once the record is written, WriterThread is free to accept new data. If the received data is from DBCron, WriterThread waits until the lock created by DBCron is removed. After the lock is released it continues to wait for new data.

DBCron periodically wakes up at 2:00 A.M., creates a lock file *dbmlock.tmp*, puts a special message "CRON" to the WriterMonitor to signal it will be accessing dbm.tmp and then checks dbm.tmp. If the file is not empty, DBCron creates a string array with all the information included in dbm.tmp and then tries to connect to Oracle with OraConnector. If DBCron can connect to Oracle, it sends the records to the database and removes all the entries that are sent from dbm.tmp. If DBCron can not

establish a connection to the database, then it puts the entries back in dbm.tmp. After updating dbm.tmp, DBCron removes the lock file and goes back to sleep.

OraConnector implements the functions to establish a connection to the Oracle database, to add records to the database and to perform queries on the database. OraConnector adds records executing an SQL statement constructed as the following:
INSERT INTO tablename VALUES (id, firstname, lastname, TO_DATE('datein timein', 'MM/DD/YYYY HH24:MI:SS), TO_DATE('dateout timeout', 'MM/DD/YYYY HH24:MI:SS')).

Use of DBManagerServer prevents the slow down of the client system performance, handling database-related transactions away from the client. The unavailability of both the Oracle server and DBManagerServer do not influence the Reader Interface.

2.4. initDB Tool

initDB tool is built to initialize and update Oracle table *lablog* to contain the logged user data as needed. InitDB uses *DBInit* class. Once started, DBInit reads all the files contained in the directory pointed to at the command line, opens each file one by one, and uses OraConnector to connect to the Oracle database, to send a series of SQL INSERT statements to populate the database and then to close the connection. DBInit does not delete any records from the database. To avoid unwilling errors, a SQL DELETE statement must be specifically run to remove the records from the table.

2.5. Oracle Server 8i

An Oracle database is used to host the lablog table. lablog is made up of the following columns:
ID (varchar2 – (17))
FIRSTNAME (varchar2 – (30))
LASTNAME (varchar2 (30))
DATEIN (date)
DATEOUT (date)
INTERVAL (varchar2 (15))
LOCATION (varchar2 (15))
Oracle statement executed to create this table with Oracle's sqlplus utility is:
create table lablog (ID varchar2(17), FIRSTNAME varchar2(30), LASTNAME varchar2(30), DATEIN date, DATEOUT date, INTERVAL varchar2(15), LOCATION varchar2(15));

A unique composite index made of ID and datein fields has been created to avoid the presence of duplicated entries. Oracle statement executed to create the index with Oracle's sqlplus utility is:
create unique index IdDate on lablog (ID, datein);
This way, if the index is violated, Oracle does not accept the record.

The server listens at the address 128.186.121.41 on port 1521 and an Oracle user is created to access the table. Oracle by default accepts maximum number of open cursors equal to 50 for a process or user. The Administrator can increase this number in the Ora.ini file. Currently Oracle server is set up with this value and when SQL statements are executed this value is taken into consideration.

2.6. Report Tool

Report Tool is designed to provide analysis of data collected by Reader Interfaces. Reader Interface writes on a text file system and the Report Tool reads from these same files exported to the host it resides on. Report Tool looks for the log files in a subdirectory structure of logs+System.FILESEPERATOR+location. Therefore, either the directory containing the log files on the client should be exported under the directory logs as *location* or in the logs directory a softlink named *location* pointing to the directory where the files are exported must be created.

Locations holds the location names for the labs. When reports for other locations are available they can easily be added to this file. Once the program is compiled again, the added locations will be available in the combo boxes on the graphical user interface.

Week, a helper class to construct graphs, holds statistical totals for the weekdays.

ReportUIP displays the graphical user interface for the tool and detects the requests made by the user. ReportUIP works with ReportPanel to display the graphs and statistics.

ReportPanel handles data collection, calculation and display of three types of reports. ReportPanel also supplies statistics.

- Real-Time Report: When a timer set to wake up periodically is activated, ReportPanel reads labname.log to count the number of entries validated as log-ins and uses GraphCanvas to plot the count on the screen. This is continued as time progresses for 30 minutes. After 30 minutes elapses the graph is refreshed to 0 minutes. ReportPanel keeps track of the minimum number and maximum number of users recorded up until that time and calculates a weighted average to determine the average number of users that were present.
- Daily Report: ReportPanel counts the entries corresponding to each hour, both for log in and log out times, by looking at the recorded datein and dateout timestamps in the labnamedate.log file for the requested date. The total number of users at the end of each hour is also calculated. GraphCanvas is used to plot the three totals. The total number of analyzed entries are counted and displayed as part of the statistics.
- Yearly Report: This type of report demands a more time-consuming process of opening the log files for the requested year and counting the entries in these files. The counts are coded based on the day and the month they were generated for. After the averages per month per day are calculated, GraphCanvas draws bar charts of the calculated information.

UsersJPanel is used to display the contents of entries for both the Real-Time Report and Daily Report on a JTable.

GraphCanvas is developed to implement and perform painting tasks of graphs supported by ReportPanel. The types of graphs drawn are bar charts and line graphs.

ReportWindow displays ReportPanel on a separate window.

2.7. Query Tool

QueryForm implements the interface to execute queries. When requested to perform the query, the information input by the user is used to construct the SQL statement depicting the query. LIKE operator is used in the query to overpass some possible data entry errors. OraConnector is called to establish a connection to the Oracle database. If a connection is established, a call to execute the query is made and the results returned by Oracle are displayed on a table. QueryForm implements a Print function to create the pages containing the input query and the results for printing. This function is built as a thread allowing the user to continue with other query requests. *SwingWorker* class provided by Sun is used to execute the actions performed on the interface on a thread, therefore allowing better rendering of the graphical interface. The form also provides a text area to permit an administrator familiar with the table structure and SQL to construct individual queries.

ProgressClip, a class implementing an animated progress bar by painting itself periodically, is used on the form's interface as a visual guide to inform the user that the program is busy performing the request.

3. USER AND MAINTENANCE DOCUMENTATION

3.1. Hardware Specifications

Reader Interface Hardware:

- A card reader hardware configured to return the account number field and the name field and to send a carriage return at the end of each field. Currently available card reader hardware is Model 1500 Microscanner decoder and Magnetic Stripe Reader from American Microsystems. A 9male/9female RS-232 cable is required to connect the decoder to the PC. The 9-pin male end is connected into connector labeled TERMINAL on the decoder and the other end is connected to the serial port either COM1 or COM2 on the PC. Magnetic Stripe Reader is connected into the circular connector labeled MSR on the decoder. This decoder is configured with the following switchboard settings:

	1	2	3	4	5	6	7	8
SW1	ON	ON	ON	ON	ON	ON	OFF	OFF
SW2	ON	ON	OFF	OFF	ON	OFF	ON	ON
SW3	ON	OFF	ON	ON	ON	ON	OFF	ON

Table 3.1: Card Reader Encoder Switchboard Setup

- A Pentium 90 or higher PC with 32MB RAM, 1GB of hard disk space for Red Hat and 10MB of free disk space for the Reader Interface and the log files, a serial port for the connection to the card reader hardware, and a network card.
- A monitor capable of a minimum resolution of 640x480 pixels.

DBManager Interface and Report-Query Tool:

- The application should be installed on a host inside the Computer Science Network to run Oracle queries.

3.2. Software Specifications

- Operating System: Reader Interface currently runs on Red Hat 6.1. The tools can be installed on NT, Solaris or Linux platform. If browser bound report tool is going to be used jdk1.2.2 plug-in is required.
- Java interpreter jdk1.2.2
- Database Server Oracle8i version 8.1.5. Minimum value of the number of maximum open cursors for a process or user is 50.
- Oracle's JDBC driver classess111.zip is required to support Java Oracle database connectivity and is included with the developed system.
- xntpd should be installed on the linux-based system that hosts Reader Interface to synchronize the time with a trustworthy server.

3.3. Reader Interface Environment Set-Up on Red Hat Linux

- Creating the users and groups:
  - user reader group reader
  - user monitor group monitors

  Following are example entries from the passwd and group files.
  /etc/passwd:
  monitor:xxxxxxxxxxxx.:500:501:Lab Monitor:/home/monitor:/bin/bash
  system:xxxxxxxxxxxxx:503:100::/home/system:/bin/bash
  reader:*:502:500:Card Reader:/home/reader
  /etc/group:
  reader::500:
  monitors:x:501:
- Setting the permissions on the port the card reader hardware is connected to:
  Change the group ownership to monitors and give read write permissions to the group.
  ```
  chgrp monitors /dev/ttyS0
  chmod g+rw ttyS0
  crw-rw----   1 root     monitors      4,  64 Jan 20 16:48 ttyS0
  ```
- Setting the clock:
  Make sure xntpd is installed and running. Configure the host's clock to synchronize with a trusted server such as xi.
- Install URW fonts if not installed.
  ```
  cd /usr/X11R6/lib/X11/fonts
  tar xvzf urw-fonts.tar.gz
  chkfontpath --add /usr/X11R6/lib/X11/fonts/URW
  ```
- Creating the directories:
  - /readerlog

    create /readerlog directory and set the ownership and permissions on the directory.
    ```
    mkdir /readerlog
    drwxrwx--x   2 root     monitors     2048 Apr 30 07:38 /readerlog
    ```
    /readerlog contents will look as the following:
    ```
    -rw-r--r--   1 monitor  monitors        0 Apr 30 14:48 majors.log
    -rw-r--r--   1 monitor  monitors    12716 Mar  2 00:04 majors03011999.log
    ```

- assuming the application is going to be installed in /usr/local/reader, create /usr/local/reader or link /usr/local/reader to the newest version of the software and set the ownership and permissions on /usr/local/reader. /usr/local should look as follows:

  ```
  lrwxrwxrwx   1 root     root          13 Feb  9  1999 java -> ./jdk117_v1a/
  drwxr-sr-x   5 505      505         1024 Feb  9  1999 jdk117_v1a

  lrwxrwxrwx   1 reader   reader         7 Mar 23 12:46 reader -> reader2
  drwxr-xr-x   3 reader   reader      1024 Mar 22  1999 reader1
  drwxr-xr-x   2 reader   reader      1024 May  5  1999 reader2
  ```

- Setting up a secure environment:
  Comment out all the entries in /etc/inetd.conf that are not needed on the host computer, e.g. finger, telnet etc.
- Set up the startup file:
  1. cd /etc
  2. Edit /etc/inittab

  Below is a sample inittab:

```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

```
# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
#1:12345:respawn:/sbin/mingetty tty1
#2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
## x:5:respawn:/usr/bin/X11/xdm -nodaemon
x:5:respawn:/root/startReader
```

- Create /root/startReader:
  -rwxr-xr-x 1 root root 155 Feb 23 17:02 startReader
  Edit /root/startReader to contain the following:
  ```
  #!/bin/sh
  PATH="$PATH:/usr/X11R6/bin"
  export PATH
  su monitor -c "/usr/X11R6/bin/xinit /etc/mon-sh"
  ```
- Create /etc/mon-sh. Assuming the interface is installed in /usr/local/reader:
  1. cd /etc
  2. ln -s /usr/local/reader/run ./mon-sh
  lrwxrwxrwx 1 root root 21 Feb 16 15:48 /etc/mon-sh -> /usr/local/reader/run
- Set the x windows environment to allow only Reader Interface to be displayed on the screen:
  Edit window manager's startup files to disable xwindow menus and taskbars.
- To allow access from a remote host, e.g. for backup purposes and report tools, make the logs available exporting /readerlog to *.cs.fsu.edu. To do this edit /etc/exports to add the following:
  /readerlog *.cs.fsu.edu(ro) backup(no_root_squash)
  exportfs –a
  Then configure the mounting of the directory.

3.4. Reader Interface Installation on Red Hat Linux

- copy source files to the installation directory.
- if different from the default, edit ReaderData.java to set the location the program is installed in. The default is /usr/local/reader
- make
- Setup the Reader Interface configuration file. The default configuration file is portconf.txt
- make sure the default /usr/local/reader/run is pointing to the right locations.
  -rwxr-xr-x 1 reader reader 1474 Mar 23 12:03 /usr/local/reader/run

  The following is the default run file included with the reader interface:

```
#!/bin/sh
PATH="$PATH:/usr/X11R6/bin"
export PATH
xset s off
/usr/X11R6/bin/RunWM --Fvwm95 &
JAVA_HOME=/usr/local/java
echo "JAVA_HOME : $JAVA_HOME"
CLASSPATH=${JAVA_HOME}/lib/classes.zip:/usr/local/reader
echo "CLASSPATH : $CLASSPATH"
echo "Starting the reader ....."
CMD="${JAVA_HOME}/bin/java -classpath ${CLASSPATH} UIWindow"
${CMD}
```

3.5. DBManager Interface and Report Interface Installation

- Install JDK 1.2.2 for Win 95/98/NT, Solaris, or Linux depending on the host operating system used.
- Copy the software to a directory.

3.6. Reader Graphical User Interface

No login procedure is required for the lab assistants (Figure 3.6.1). Lab assistants use the card reader hardware to swipe the FSU ID card of the student. The application obtains the ID, last name and first name of the student. It internally detects if the user is leaving or entering the lab derived from the number of times the same card is swiped. If the user is validated to come in, the date and time of validation is added as a combined field to the entry. The entry is displayed in green color in the "[Scanned]" text box and the recorded information is added to the table on the user window. If the user is validated leaving, the entry is displayed in red color in the "[Scanned]" text box and removed from the table. The interface displays the current number of users in the lab.

If the card reader hardware can not read from the card, the lab assistant can manually enter the information in the log entry boxes and press the *LOG* button to complete the validation. The ID text box has to be filled for the entry to be accepted. The user can also click on any entry to copy the contents to the id, name and last name text boxes. This is useful in removing entries that were added manually, relieving the user from typing in the id again.

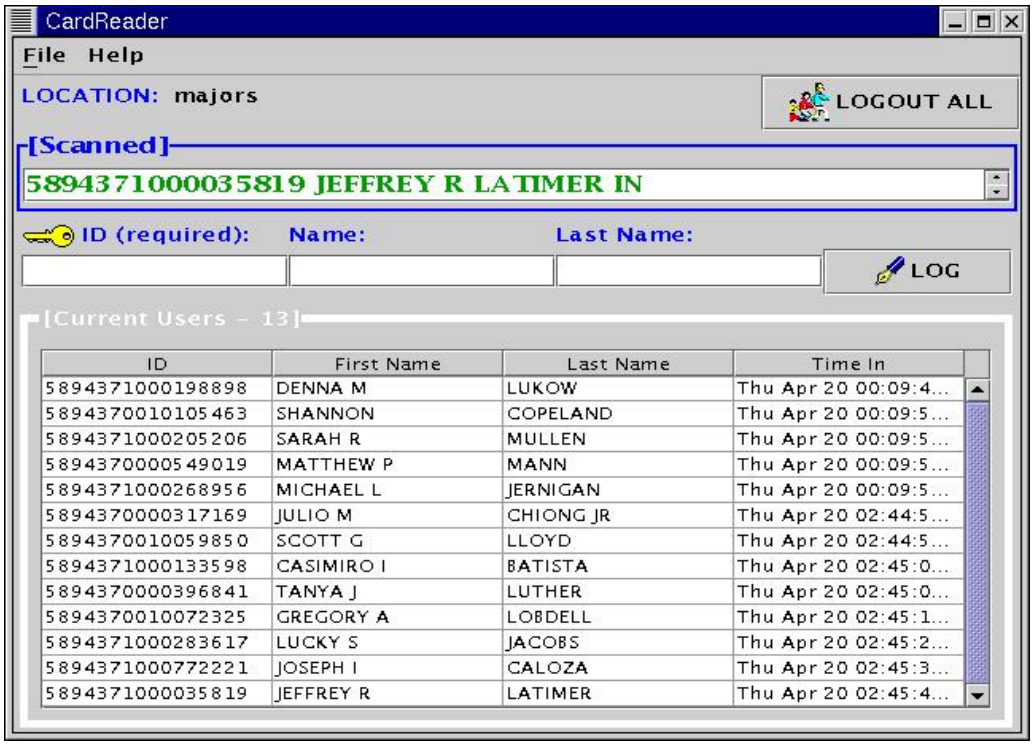

Figure 3.6.1: Reader Graphical User Interface

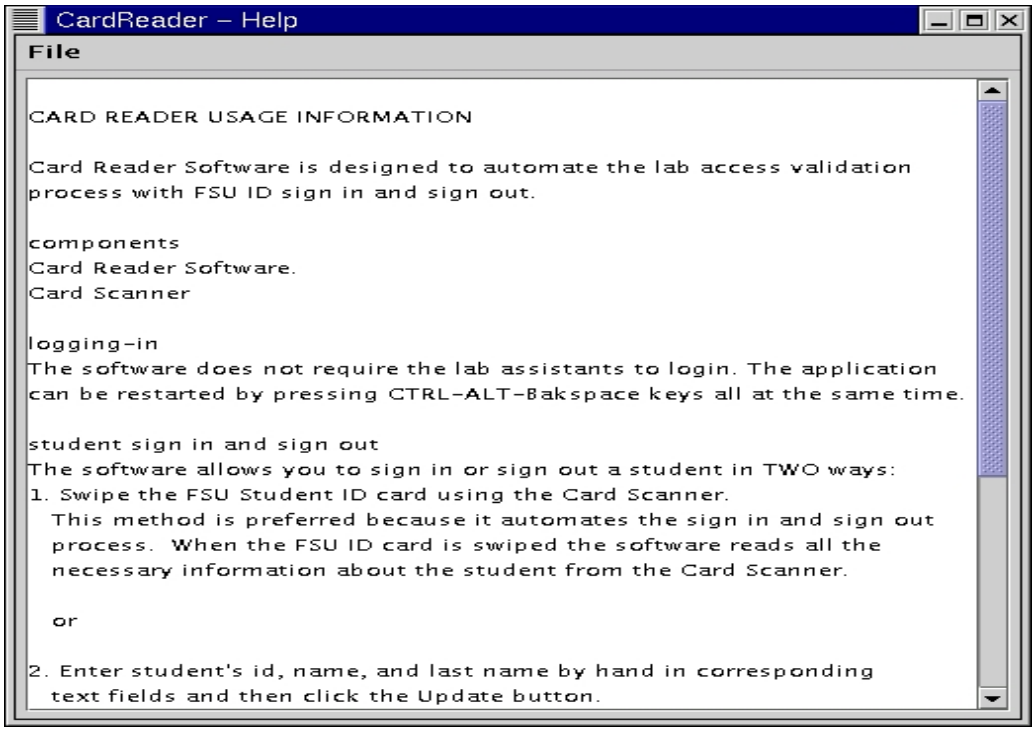

Figure 3.6.2: Help Window

The interface includes a *LOGOUT ALL* button to allow the lab assistant to remove all the entries. This feature supports fast log-off procedure at closing time when several students leave the lab at the same time.

An info page describing the validation instructions can be viewed choosing the Help option from the menu. (Figure 3.6.2).

The application is easily restarted with CTRL-ALT-←

3.7. Configuration Tool

Configuration Tool is run executing readercfgtool at the prompt (Figure 3.7.1) and allows System Administrators to configure Reader Interfaces for each location by setting:

- the location the interface is installed for, e.g. majors, literacy,
- the port the card reader is physically connected to,
- the directory where the log files will be kept. A browser interface assists in picking a directory (Figure 3.7.2),
- the operational schedule of the lab,
- DBEnable-DBDisable option list to allow database connectivity, Serverhost IP for DBManagerServer host IP and Port to point to the port DBManager is listening to.

Instead, if the configuration file is to be edited, the following must be fulfilled.

- The values for the parameters must be set on the line following the keyword.
- The format of schedule for the days is:

Day	OpenTime	CloseTime
	hour:minutes:seconds|+	hour:minutes:seconds|+

  The range of values for the hour, minute and second fields are: (00-23):(00-59):(00-59)
  Anything at midnight (00:00:00) or after midnight should be specified with |+ flag. As a special case anything between 24:00:00 and 24:59:59 is treated as after midnight and does not require the use of + flag.
- The # sign is assumed to be introducing comment on that line.

Below is a complete sample configuration file:

```
#This should be the lab name the card reader is collecting information for
LOCATION
majors
LOGDIR
/readerlog
# This is the port the card reader is physically connected to.
PORT
/dev/ttyS1
DBENABLE
true
SERVERHOST
128.186.121.41
SERVERPORT
```

5555
SCHEDULE
Mon 08:00:00 00:30:00|+
Tue 08:00:00 00:30:00|+
Wed 08:00:00 00:30:00|+
Thu 08:00:00 00:30:01|+
Fri 08:00:00 18:30:00|
Sat 08:00:00 18:30:00|
Sun 08:00:00 00:30:00|+

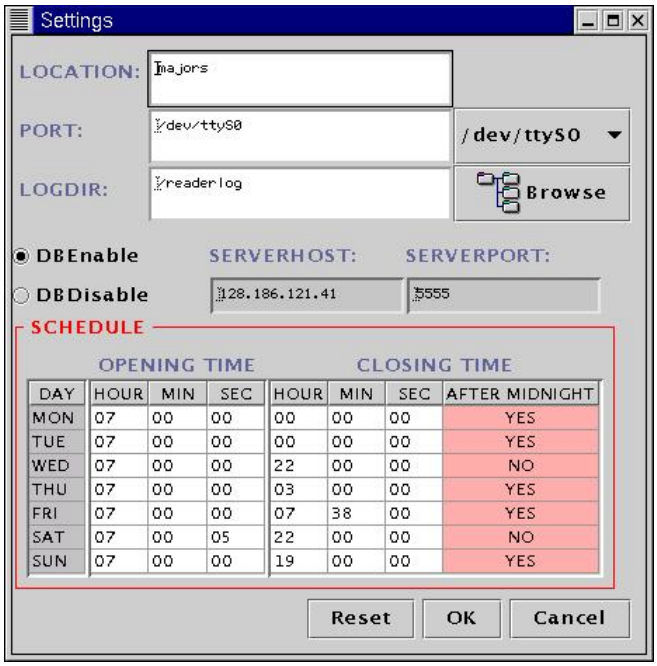

Figure 3.7.1: Configuration Tool

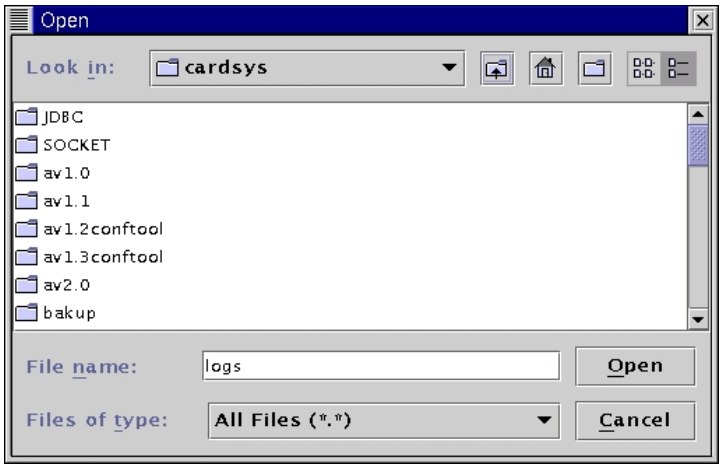

Figure 3.7.2: Browse window

3.8. DBManagerServer

DBManagerServer should be started if the information collected by the Reader Interface is going to be sent to the Oracle server. startServer command starts the DBManager server on a unix or linux based platform and if desired the environment should be set up to start the server at boot time.

3.9. InitDB Tool

initDB allows the addition of new records to the Oracle table to contain the data collected in the log files for the computer labs. It is executed with initDB command.
initDB accepts the following options:
-ffilename: file to insert to the table.
-ddirectory: directory where the log files are located.
-llocation: location the log files are for.

To insert one file only:
initDB –ffilename
To insert logs for majors found in somelogs directory:
initDB –lmajors -dsomelogs

3.10. Report Tool

Report Tool is platform independent and allows the analysis of lab usage. Report Tool can be run as an application or an applet in Java appletviewer or in a Java enabled browser. *runrep.bat* (NT) or *runrep* (Solaris, Linux) starts the Report Tool as an application. After choosing the lab to observe, the type of report to view and clicking the S*how* button, the report is displayed on the current window. Multiple reports can be viewed at the same time with the help of the *Compare* button. In this case, the resulting report is displayed in a new window.

Real-Time Report shows a continuously updated current view of lab usage, displaying the number of students currently in the lab with a refresh rate of six seconds. When 30 minutes is reached, the graph refreshed updating the reference time on the x-axis (Figure 3.10.1).

Daily Report plots a diagram based on the information read from the log file available for the requested day (Figure 3.10.2). Three diagrams are displayed over a 24-hour range:

- total number of students *coming IN* every hour (yellow bar)
- total number of students going OUT every hour (red bar)
- total number of students who were in the lab at the end of the hour (interpolating cyan line)

When the Users tab is selected, Real-Time Report displays the students currently using the lab (Figure 3.10.3). Daily Report shows all the entries for that day.

Yearly Report plots the *average request for the lab service* for the selected year. Two types of averages are plotted, a monthly average (white dots) and weekday of the month averages for each month (colored bar charts) (Figure 3.10.4). The averages are based on the record count for each day and month.

Query Tool can be accessed from the Report Tool clicking the *Query* button.

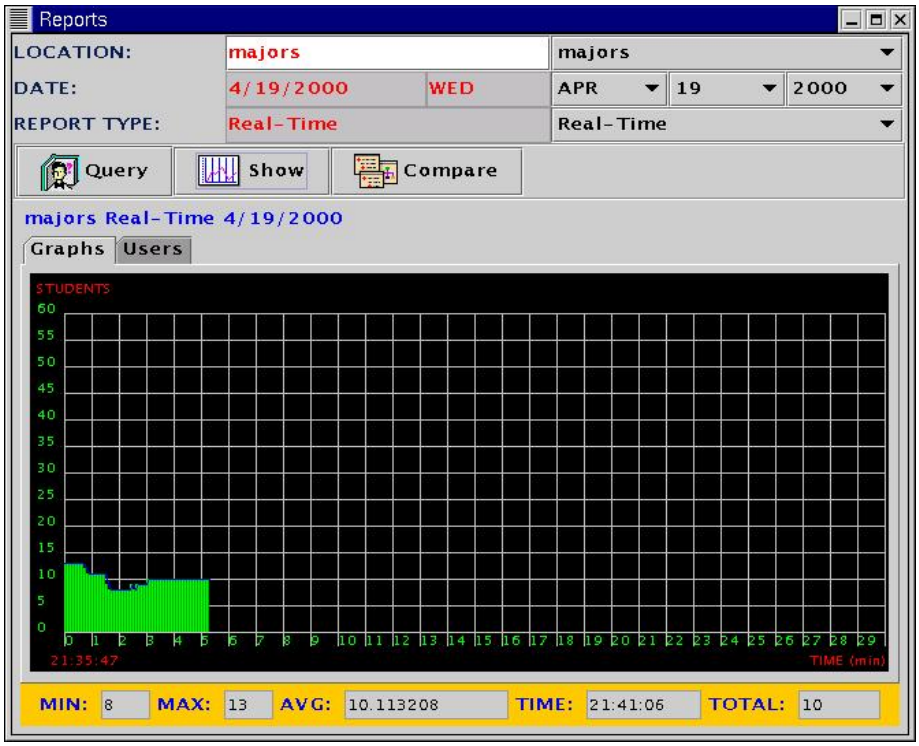

Figure 3.10.1: Report Tool Real-Time Report

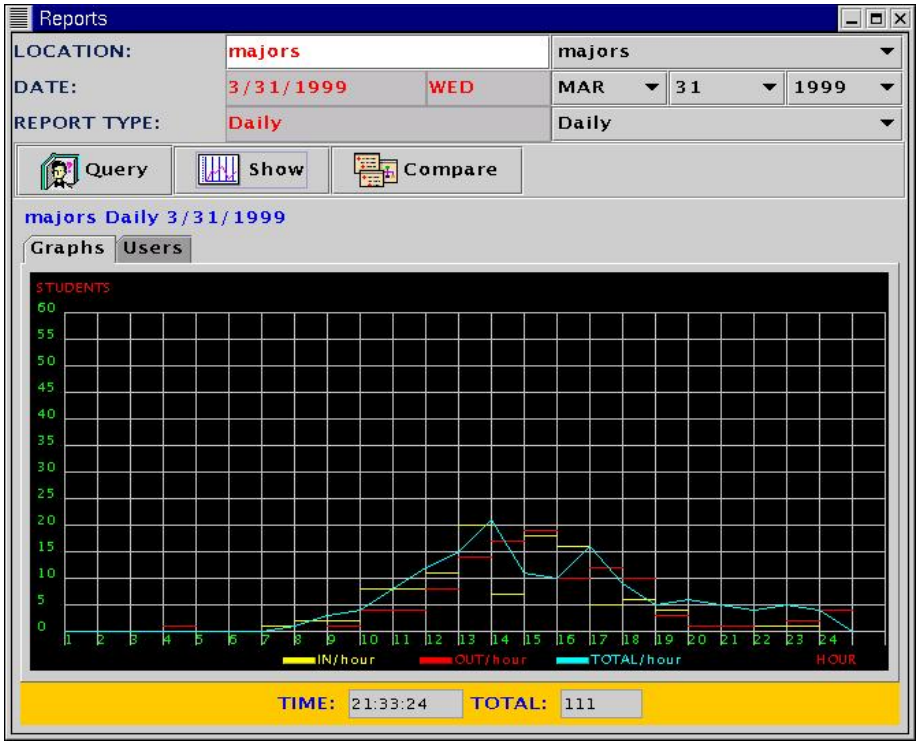

Figure 3.10.2: Report Tool Daily Report

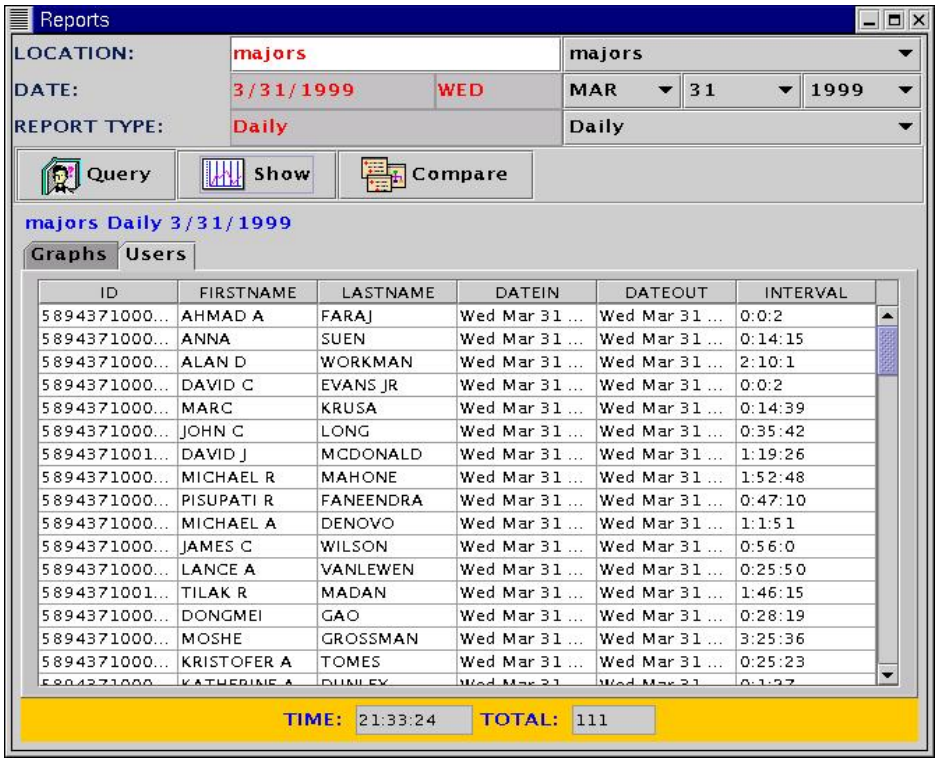

Figure 3.10.3: Report Tool Users table

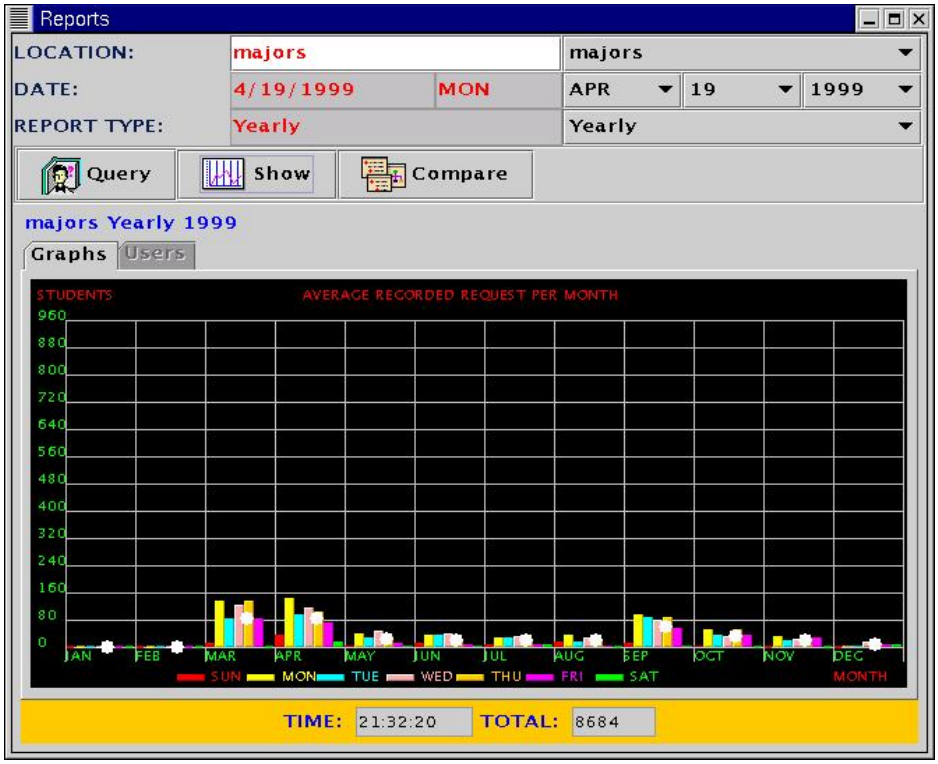

Figure 3.10.4: Report Tool Yearly Report

3.11. Query Tool

Query Tool allows the user to retrieve records that match the given criteria (Figure 3.11.1). Only users with the appropriate permissions should use this interface to protect data privacy.

The user can take advantage of the pre-structured dynamic query that allows the specification of the location, student ID, name, and time interval in any combination or build an SQL statement and press the *Run* button. The pre-structured dynamic query allows the user to enter a partial value of a location, ID or name. When fields are left blank all the entries with zero or more characters match that field. The query is built to include all the records that contain dates that match values greater than or equal to FROM DATE field and less than or equal to TO DATE field. If any of the date fields are left blank by clicking the Reset button then all the entries matching the remaining date criteria as mentioned above are displayed. Therefore, if both date fields are left blank, all the records matching any date will be considered. When Query or QueryAppend buttons are clicked the records are displayed in a table format. *Query* button refreshes the table each time a query is sent to the database. *QuaryAppend* button appends the new set of records received to the ones obtained on a prior query. The user can interrupt a query at any time clicking the *Stop* button.

Once the user has collected the requested information on the table, the query results can be printed clicking the *Print* button. The contents can be printed to a file or sent to a printer (Figure 3.11.2).

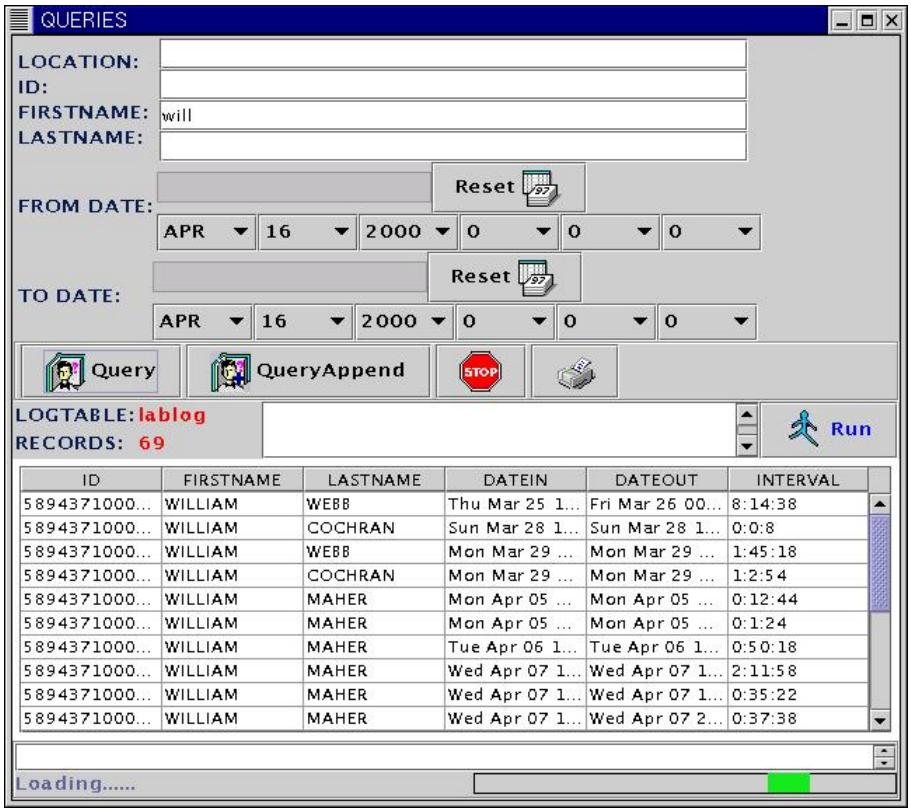

Figure 3.11.1: Query Tool

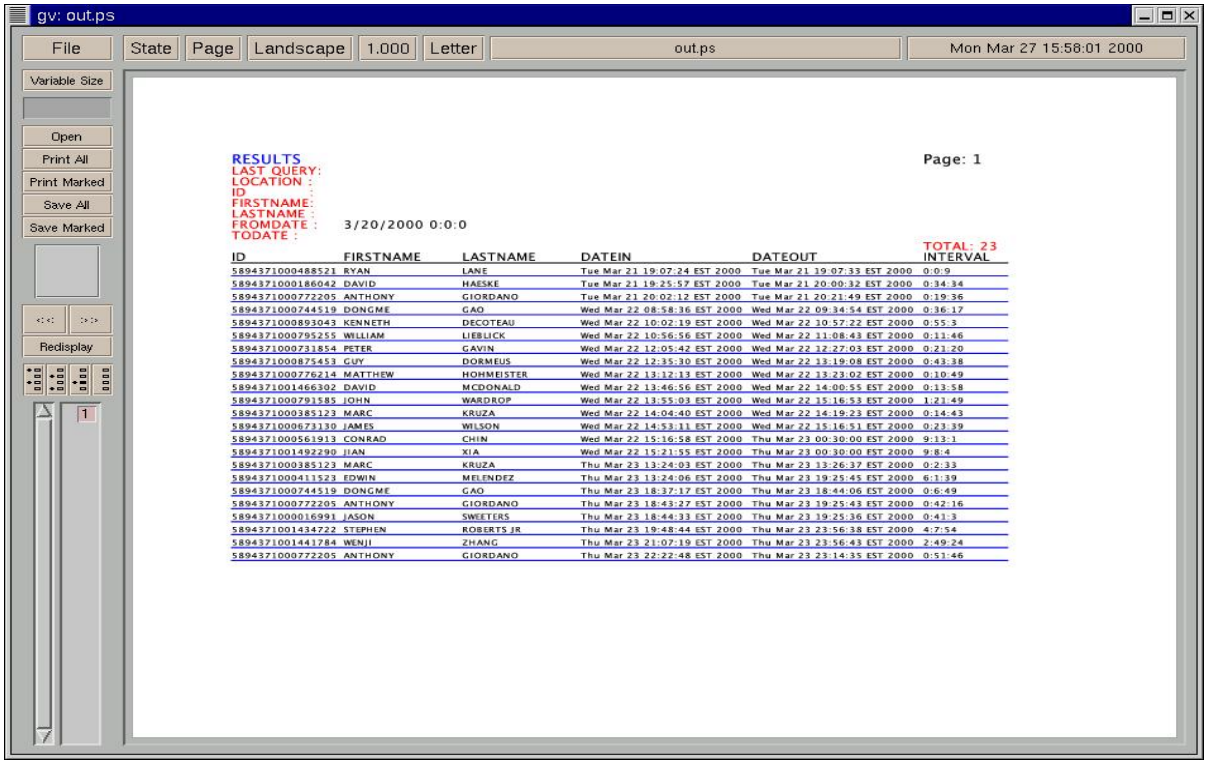

Figure 3.11.2: Ghostview look of the printed output.

4. VALIDATION TESTING AND PERFORMANCE ANALYSIS

Reader interface has been running for a year without interruption.

Magnetic stripe reader is sensitive to physical impact and at times requires replacement.

More than 9000 records were collected in one year. The hard drive space used by one year of data collected in majors lab is 1.2 MB (305 files). The average size of the daily log files is 4 KB.

Initialization of the Oracle database with 9000 records takes about 3 hours with a modem connection of 26000bps.

When query access with modem connection is observed, queries that return record counts less than 100 provide much faster responses.

Host Machine	Connection Speed	9382 records	less than 100
PentiumIII 500 MHz, 128 RAM, WindowsNT Server 4.0	24,000 bps	7 min	2-4 sec
PentiumI 166 MHz, 64 RAM, Windows 98	26,400 bps	22 min	5-7 sec

Table 4.1: Query response time with modem connection.

5. CONCLUSIONS AND FUTURE ENHANCEMENTS

Design of a ReaderDriver module that runs on Windows Operating System would enhance the platform-independence of the Reader Interface on platforms that are commonly used in Computer Science Department.

A validation that verifies the user is a member of Computer Science Department community by connecting to the main database at FSU can support the verification process. This however would tie the application to an outside source that is not maintained by Computer Science Department.

The implementation of this system which serves as a platform for both data collection and analysis simplifies the task of staff to monitor the usage of computer labs and allows better interpretation of the collected information.

6. REFERENCES

1. Mary Campione et al., The Java Tutorial, Second Edition, Sun Microsystems, Addison-Wesley, 1999.
2. Dr. Satyaraj Pantham, Pure JFC Swing, SAMS, 1999.
3. Kevin Loney et al, Oracle8I DBA Handbook, Oracle Press, Osbore/McGraw-Hill, 2000.
4. James R. Groff et al., The Complete Reference SQL, Osborne/McGraw-Hill, 1999.
5. Red Hat Official Web site, http://www.redhat.com.
6. User's Manual, Magnetic Stripe Reader Model 150, American Microsystems.
7. IconBazaar Computer Icons, http://www.iconbazaar.com